UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

DEPARTMENT OF INDUSTRIAL ENGINEERING

Linear Regression in R

Instructor: Christian Capezza
Course leaders: Professors Biagio Palumbo, Antonio Lepore
Course: Statistics for Technology, a.y. 2019/2020
MSc in Mechanical Engineering for Design and Production

29 November 2019

Contents

Linear regression in R

Esempio 13.1

Disponiamo dell'età e della pressione arteriosa di 36 uomini ritenuti in buone condizioni di salute. I dati, ordinati per età crescente, sono disponibili nel file "pressione_arteriosa.csv", che possiamo importare in una variabile della classe *data frame*

```
pressione_df <- read.csv("pressione_arteriosa.csv")
class(pressione_df)
```

```
## [1] "data.frame"
```

Un data frame è una tabella di dati, come una matrice, in cui però ogni colonna può essere di un tipo diverso. In questo caso, abbiamo solo due colonne numeriche. visualizziamo il data frame

```
pressione_df
```

```
##    id_paziente eta pressione_arteriosa_mmHg
```

```
## 1  paziente_01  33                        127
## 2  paziente_02  37                        126
## 3  paziente_03  38                        118
## 4  paziente_04  39                        113
## 5  paziente_05  39                        126
## 6  paziente_06  40                        120
## 7  paziente_07  42                        132
## 8  paziente_08  43                        122
## 9  paziente_09  44                        134
## 10 paziente_10  47                        147
## 11 paziente_11  47                        132
## 12 paziente_12  47                        131
## 13 paziente_13  48                        120
## 14 paziente_14  48                        130
## 15 paziente_15  49                        144
## 16 paziente_16  50                        136
## 17 paziente_17  50                        125
## 18 paziente_18  51                        128
## 19 paziente_19  52                        152
## 20 paziente_20  53                        142
## 21 paziente_21  53                        148
## 22 paziente_22  54                        122
## 23 paziente_23  55                        161
## 24 paziente_24  56                        132
## 25 paziente_25  56                        142
## 26 paziente_26  57                        135
## 27 paziente_27  57                        146
## 28 paziente_28  58                        158
## 29 paziente_29  61                        152
## 30 paziente_30  62                        145
## 31 paziente_31  63                        148
## 32 paziente_32  65                        162
## 33 paziente_33  65                        148
## 34 paziente_34  66                        145
## 35 paziente_35  68                        151
## 36 paziente_36  75                        172
```

Possiamo ottenere alcune informazioni sul tipo di variabili per ogni colonna del data frame con la funzione str

```r
str(pressione_df)
```

```
## 'data.frame':    36 obs. of  3 variables:
##  $ id_paziente             : Factor w/ 36 levels "paziente_01",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ eta                     : int  33 37 38 39 39 40 42 43 44 47 ...
##  $ pressione_arteriosa_mmHg: int  127 126 118 113 126 120 132 122 134 147 ...
```

e alcune statistiche sintetiche sulle usando la funzione summary

```r
summary(pressione_df)
```

```
##       id_paziente      eta        pressione_arteriosa_mmHg
##  paziente_01: 1   Min.   :33.00   Min.   :113.0
##  paziente_02: 1   1st Qu.:46.25   1st Qu.:126.8
##  paziente_03: 1   Median :51.50   Median :135.5
##  paziente_04: 1   Mean   :51.89   Mean   :138.1
##  paziente_05: 1   3rd Qu.:57.25   3rd Qu.:148.0
```

```
##  paziente_06: 1   Max.   :75.00   Max.   :172.0
##  (Other)    :30
```

Si può analizzare il numero di righe o colonne di un data frame attraverso

```
nrow(pressione_df)
```

```
## [1] 36
```

```
ncol(pressione_df)
```

```
## [1] 3
```

Inoltre spesso è utile avere accesso al nome delle colonne e delle righe del dataframe

```
# Nomi delle colonne
colnames(pressione_df)
```

```
## [1] "id_paziente"              "eta"
## [3] "pressione_arteriosa_mmHg"
```

```
# Nomi delle righe
rownames(pressione_df)
```

```
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "15"
## [16] "16" "17" "18" "19" "20" "21" "22" "23" "24" "25" "26" "27" "28" "29" "30"
## [31] "31" "32" "33" "34" "35" "36"
```

Infatti è possibile selezionare determinate colonne o righe del dataframe

```
pressione_df[1,] # Prima riga del data frame
```

```
##   id_paziente eta pressione_arteriosa_mmHg
## 1 paziente_01  33                      127
```

```
pressione_df[1:10,] # Prime dieci righe del data frame
```

```
##    id_paziente eta pressione_arteriosa_mmHg
## 1  paziente_01  33                      127
## 2  paziente_02  37                      126
## 3  paziente_03  38                      118
## 4  paziente_04  39                      113
## 5  paziente_05  39                      126
## 6  paziente_06  40                      120
## 7  paziente_07  42                      132
## 8  paziente_08  43                      122
## 9  paziente_09  44                      134
## 10 paziente_10  47                      147
```

```
pressione_df[,2] # Seconda colonna del data frame
```

```
##  [1] 33 37 38 39 39 40 42 43 44 47 47 47 48 48 49 50 50 51 52 53 53 54 55 56 56
## [26] 57 57 58 61 62 63 65 65 66 68 75
```

```
pressione_df$eta # Equivalentemente, colonna del data frame richiamata attraverso il suo nome
```

```
##  [1] 33 37 38 39 39 40 42 43 44 47 47 47 48 48 49 50 50 51 52 53 53 54 55 56 56
## [26] 57 57 58 61 62 63 65 65 66 68 75
```

```
# Nomi delle righe
rownames(pressione_df)
```

```
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "15"
## [16] "16" "17" "18" "19" "20" "21" "22" "23" "24" "25" "26" "27" "28" "29" "30"
## [31] "31" "32" "33" "34" "35" "36"
```

Analisi di regressione manualmente

Per effettuare l'analisi di regressione, dobbiamo estrarre da un data frame generico, che può contenere numeri così come testo, degli oggetti matematici che è possibile maneggiare, come matrici e vettori.

```
x_eta <- pressione_df$eta
y_pressione <- pressione_df$pressione_arteriosa_mmHg
```

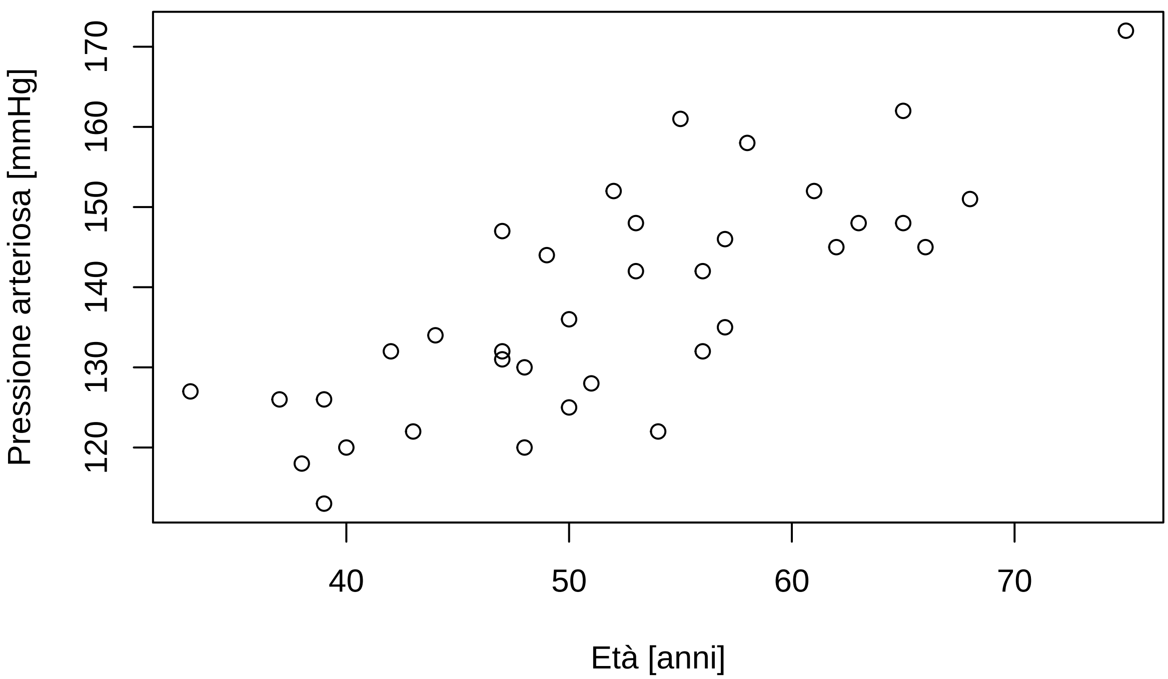
Il coefficiente di correlazione tra la pressione Y e l'età X, benché non elevato:

```
rho <- cov(x_eta,y_pressione) / (var(x_eta) * var(y_pressione))^0.5
rho
```

```
## [1] 0.7777252
```

E' comunque indice dell'esistenza di una dipendenza lineare. Infatti, possiamo visualizzare un grafico delle osservazioni delle due variabili

```
plot(x_eta, y_pressione, xlab="Età [anni]", ylab="Pressione arteriosa [mmHg]")
```



che mostra un certo grado di allineamento. Si può quindi procedere alla stima dei parametri col metodo dei minimi quadrati

Il modello di regressione è $y = ax + b + \varepsilon$, e le stime $\hat{a}$ di a e $\hat{b}$ di b sono date da

$\hat{a} = \frac{\widehat{\text{Cov}}(x,y)}{\widehat{Var}(x)}$

$\hat{b} = \overline{y} - \hat{a}\overline{x}$

Dunque:

```
a_hat <- cov(x_eta,y_pressione) / var(x_eta)
a_hat
```

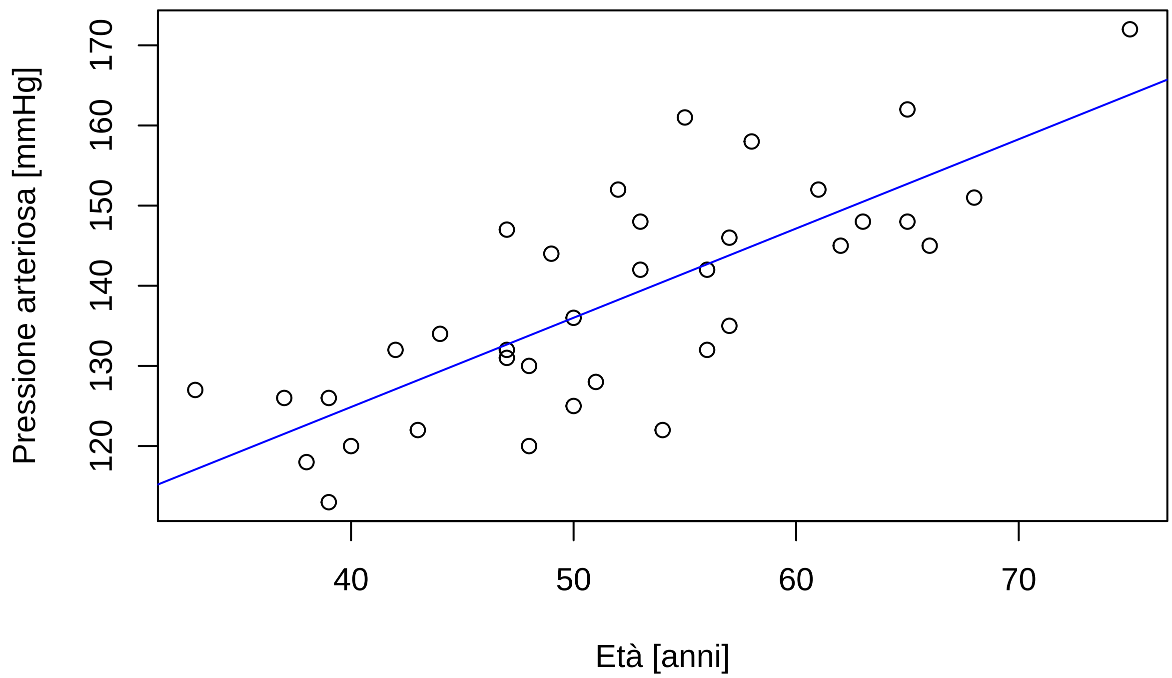
```
## [1] 1.113618
```

```
b_hat <- mean(y_pressione) - a_hat * mean(x_eta)
b_hat
```

```
## [1] 80.32669
```

Possiamo disegnare la retta di regressione

```
plot(x_eta, y_pressione, xlab="Età [anni]", ylab="Pressione arteriosa [mmHg]")
abline(a = b_hat, b = a_hat, col = "blue")
```



La dispersione dei dati intorno alla retta non è trascurabile. Tuttavia, essendo $\rho^2 = 0.78^2 = 0.61$, grazie a questa regressione si riesce a spiegare con l'età circa il 60% della varianza della pressione arteriosa dei 36 uomini.

Analisi di regressione con la funzione lm (linear model)

R consente di agevolare l'analisi di regressione, fornendo una funzione `lm` che include la maggior parte dei calcoli richiesti. Inoltre, consente di agire direttamente sui data frame, senza dover estrarre a mano le colonne relative alle variabili interessate (noi però gli diamo le colonne già estratte nelle variabili x_eta e y_pressione).

Per effettuare l'analisi di regressione, basta scrivere in questo modo:

`lm(variabile_di_risposta ~ variabile indipendente)`

Ossia, nell'esempio

```
modello_regressione <- lm(y_pressione ~ x_eta)
class(modello_regressione)
```

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

In una singola variabile, sono conservate tutte le informazioni desiderate. Per una sintetica analisi

```
summary(modello_regressione)
```

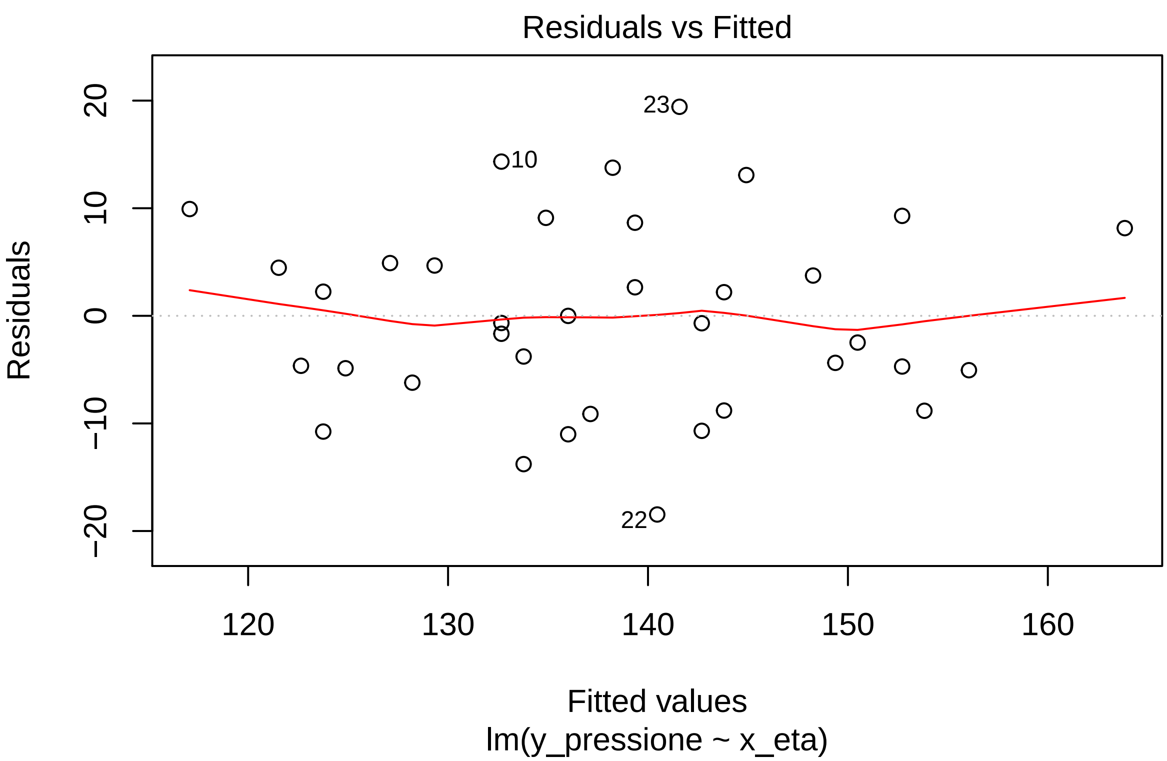
```
##
```

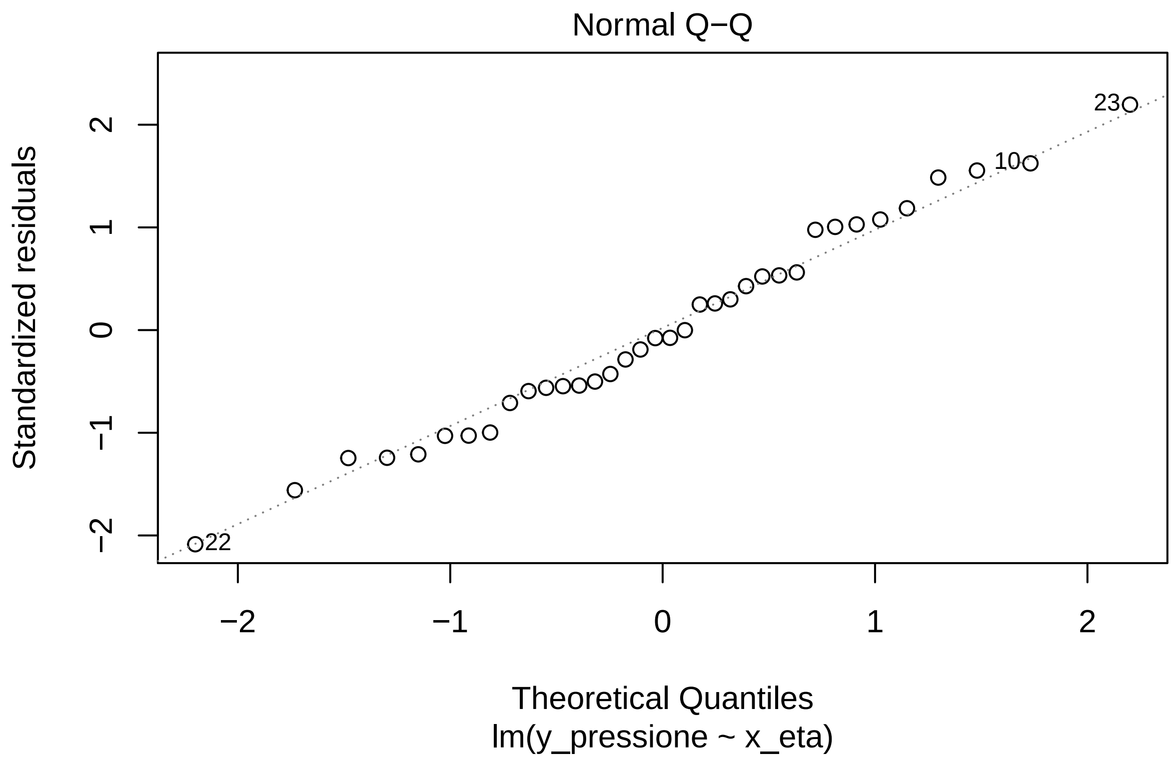
```
## Call:
## lm(formula = y_pressione ~ x_eta)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.462  -5.343  -0.678   5.714  19.424
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  80.3267     8.1486   9.858 1.68e-11 ***
## x_eta         1.1136     0.1544   7.214 2.40e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.984 on 34 degrees of freedom
## Multiple R-squared:  0.6049, Adjusted R-squared:  0.5932
## F-statistic: 52.04 on 1 and 34 DF,  p-value: 2.396e-08
```

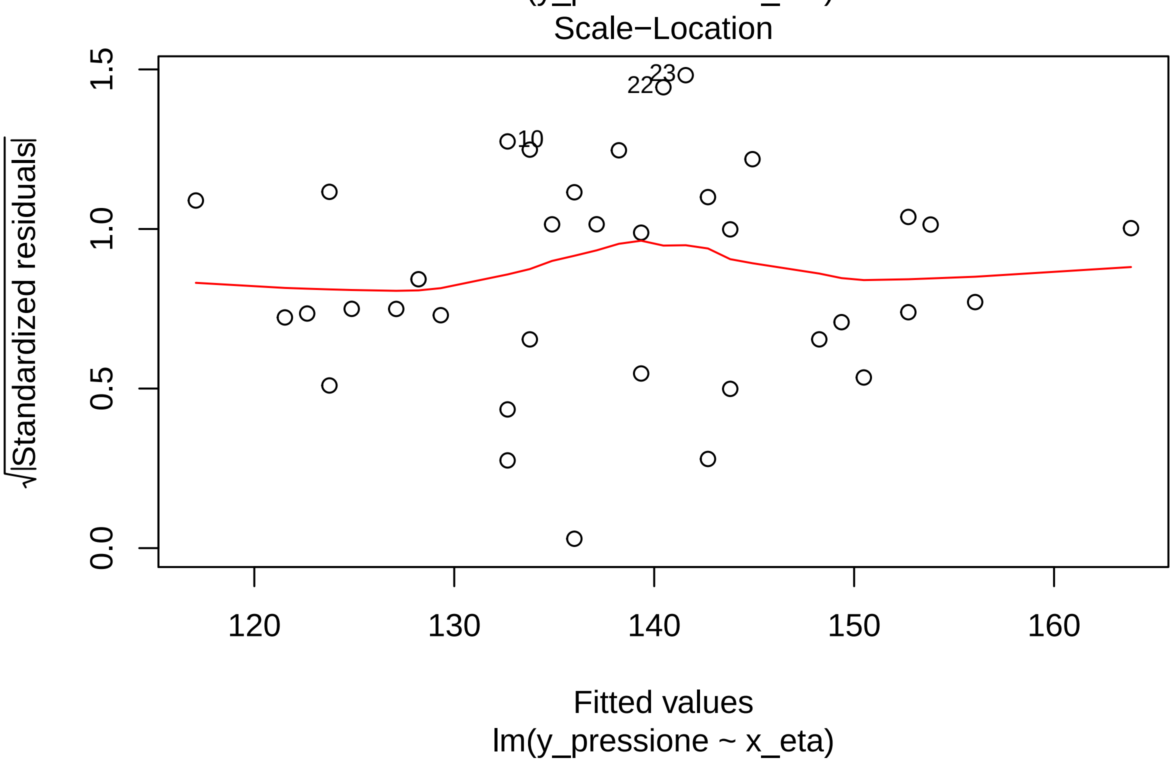
Viene fornito anche il coefficiente di determinazione, indicato come R-squared.

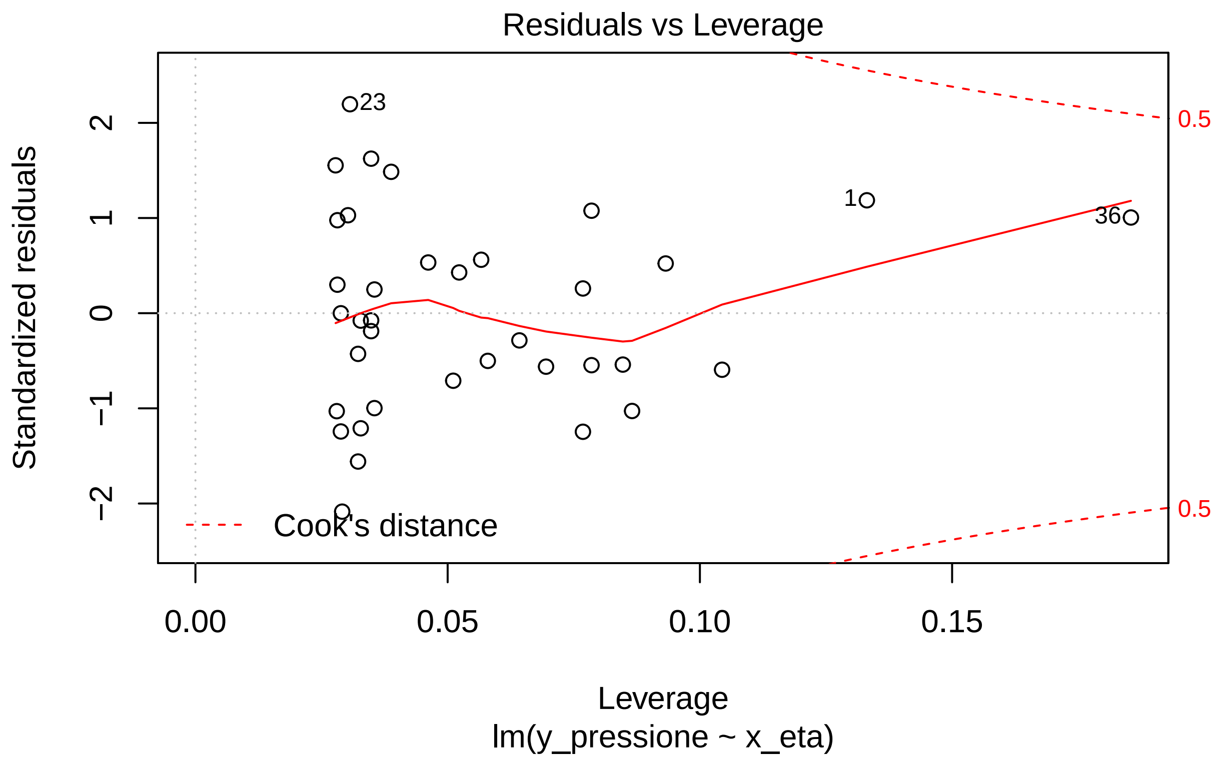
Check dei residui

```
plot(modello_regressione)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(y_pressione ~ x_eta)
23
10
22
−2
−1
0
1
2


Scale−Location
√|Standardized residuals|
Fitted values
lm(y_pressione ~ x_eta)
22
23
10
0.0
0.5
1.0
1.5
120
130
140
150
160



Esempio 13.2

Si applichi il test di dipendenza lineare ai dati dell'esempio 13.1. Si effettui nuovamente la stima dei parametri del modello di regressione.

```
pressione_df <- read.csv("pressione_arteriosa.csv")

x_eta <- pressione_df$eta
y_pressione <- pressione_df$pressione_arteriosa_mmHg
a_hat <- cov(x_eta,y_pressione) / var(x_eta)
b_hat <- mean(y_pressione) - a_hat * mean(x_eta)
```

Ora, per effettuare il test bisogna valutare se è vera l'ipotesi nulla che $a = 0$. In tal caso, il rapporto: $\frac{\Sigma_f/1}{\Sigma_e/(n-2)}$,

dove $\Sigma_f = \sum_{i=1}^{n}(\hat{y}_i - \overline{y})^2$ e $\Sigma_e = \sum_{i=1}^{n} e_i^2$, con $e_i = y_i - \hat{y}_i$, si distribuisce come una variabile aleatoria di Fisher, con 1 e $n-2$ gradi di libertà. Per effettuare il test a una coda, si calcoli la statistica suddetta.

```
y_hat <- a_hat * x_eta + b_hat
e <- y_pressione - y_hat
Sigma_e <- sum(e^2)
Sigma_e
```

```
## [1] 2744.492
```

```
Sigma_f <- sum( (y_hat - mean(y_pressione))^2 )
Sigma_f
```

```
## [1] 4201.064
```

```
n <- length(y_pressione)
Z <- Sigma_f / (Sigma_e / (n-2) )
```

```
Z
```

```
## [1] 52.04467
```

Se il valore ottenuto di Z è maggiore del limite superiore dell'intervallo di accettazione dell'ipotesi nulla, con un livello di significatività $1 - \alpha = 0.95$, allora il test è rigettato, dimostrando la dipendenza lineare tra le variabili.

```
qf(0.95, 1, n-2)
```

```
## [1] 4.130018
```

In alternativa, si può equivalentemente considerare il p-value corrispondente al valore della statistica calcolato

```
1 - pf(Z, 1, n-2) # L'area della coda destra è 1 - cdf
```

```
## [1] 2.395771e-08
```

Confrontando i risultati ottenuti con la funzione lm di R

```
modello_regressione <- lm(y_pressione ~ x_eta)
summary(modello_regressione)
```

```
##
## Call:
## lm(formula = y_pressione ~ x_eta)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -18.462  -5.343  -0.678   5.714  19.424
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  80.3267     8.1486   9.858 1.68e-11 ***
## x_eta         1.1136     0.1544   7.214 2.40e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.984 on 34 degrees of freedom
## Multiple R-squared:  0.6049, Adjusted R-squared:  0.5932
## F-statistic: 52.04 on 1 and 34 DF,  p-value: 2.396e-08
```

si può osservare che il valore del p-value ottenuto coincide con quello corrispondente al test sul coefficiente di regressione relativo alla variabile età.

Esempio 13.3

Si parta dagli esempi precedenti

```
pressione_df <- read.csv("pressione_arteriosa.csv")

x_eta <- pressione_df$eta
y_pressione <- pressione_df$pressione_arteriosa_mmHg
a_hat <- cov(x_eta,y_pressione) / var(x_eta)
b_hat <- mean(y_pressione) - a_hat * mean(x_eta)
y_hat <- a_hat * x_eta + b_hat
e <- y_pressione - y_hat
n <- length(y_pressione)
```

Si calcolino gli intervalli di confidenza dei parametri a e b.

Sapendo che

- $\hat{a} \sim N(a, \sigma_a^2)$, con $\sigma_a^2 = \frac{\sigma^2}{\sum_{i=1}^n (x_i - \overline{x})^2}$
- $\hat{b} \sim N\left(b, \sigma_b^2\right)$, con $\sigma_b^2 = \sigma^2 \left(\frac{1}{n} + \frac{\overline{x}^2}{\sum_{i=1}^n (x_i - \overline{x})^2}\right)$

si calcoli prima di tutto una stima corretta di $\sigma^2 = \mathrm{Var}(z_i)$, che è $\hat{\sigma}^2 = \frac{1}{n-2} \sum_{i=1}^n (y_i - \hat{y}_i)^2$ (vedi formula (13.23) del libro di Erto)

```
sigma_hat_squared <- 1 / (n - 2) * sum(e ^ 2)
sigma_hat_squared
```

```
## [1] 80.72035
```

Si noti che la sua radice quadrata, ossia la stima della deviazione standard dei residui, coincide con l'output "Residual standard error" di `lm`:

```
sqrt(sigma_hat_squared)
```

```
## [1] 8.98445
```

Adesso si possono calcolare le stime delle varianze e delle deviazioni standard di $\hat{a}$ (`sigma_hat_a_squared` e `sigma_hat_a`, rispettivamente) e $\hat{b}$ (`sigma_hat_b_squared` e `sigma_hat_b`, rispettivamente). Chiamiamo con `devianza_x` il termine $\sum_{i=1}^n (x_i - \overline{x})^2$,

```
devianza_x <- sum( (x_eta - mean(x_eta))^2 )
sigma_hat_a_squared <- sigma_hat_squared / devianza_x
sigma_hat_a <- sigma_hat_a_squared ^ 0.5
sigma_hat_a
```

```
## [1] 0.1543648
```

```
sigma_hat_b_squared <- sigma_hat_squared * ( 1/n + mean(x_eta)^2 / devianza_x)
sigma_hat_b <- sigma_hat_b_squared ^ 0.5
sigma_hat_b
```

```
## [1] 8.148584
```

Ora è possibile calcolare gli intervalli di confidenza dei parametri, con livello di confidenza 0.95 (vedi formula (13.26) del libro di Erto). Si ricordi che gli intervalli di confidenza sono quegli intervalli $I_a, I_b \subseteq \mathbb{R}$, dove

- $I_a = (\hat{a} - t_{0.975}\hat{\sigma}_a, \hat{a} + t_{0.975}\hat{\sigma}_a)$
- $I_b = \left(\hat{b} - t_{0.975}\hat{\sigma}_b, \hat{b} + t_{0.975}\hat{\sigma}_b\right)$

tali che $\Pr\left(a \in I_a\right) = 0.95$ e $\Pr\left(b \in I_b\right) = 0.95$.

```
t_0975 <- qt(0.975, n-1)
I_a_sup <- a_hat + t_0975 * sigma_hat_a
I_a_inf <- a_hat - t_0975 * sigma_hat_a
c(I_a_inf,I_a_sup)
```

```
## [1] 0.8002413 1.4269957
```

```
I_b_sup <- b_hat + t_0975 * sigma_hat_b
I_b_inf <- b_hat - t_0975 * sigma_hat_b
c(I_b_inf,I_b_sup)
```

```
## [1] 63.78418 96.86919
```

Si noti che le deviazioni standard calcolate `sigma_hat_a` e `sigma_hat_b` coincidono con le voci Std. Error nell'output di `lm`, e possono essere richiamate direttamente con

```
summary(modello_regressione)$coefficients[, "Std. Error"]
```

```
## (Intercept)       x_eta
##   8.1485840   0.1543648
```

```
sigma_hat_a # Check slopes are equal
```

```
## [1] 0.1543648
```

```
sigma_hat_b # Check intercepts are equal
```

```
## [1] 8.148584
```

Test d'ipotesi sui parametri

In generale gli intervalli di confidenza come quelli appena calcolati possono essere utilizzati anche per effettuare test d'ipotesi sui parametri a e b. Infatti basta ricordare che se uno specifico valore di un parametro è incluso nel relativo intervallo di confidenza, l'ipotesi ad esso legata non è rigettabile, e ciò può essere affermato allo stesso livello di confidenza in base al quale è stato costruito l'intervallo.

In ogni caso, volendo effettuare un test d'ipotesi sui parametri, senza voler calcolare gli intervalli di confidenza, si considera prima l'ipotesi nulla, e la si confronta con le ipotesi alternative. In genere, nel caso dei coefficienti di regressione si vuole testare l'ipotesi nulla che il valore vero del coefficiente sia zero. L'ipotesi alternativa di solito è che il valore vero del parametro sia diverso da zero, ma sono possibili anche test a una coda.

Se l'ipotesi nulla è che $a = 0$, o che $b = 0$, allora, se il livello di significatività del test è $1 - \alpha = 0.95$, allora deve verificarsi che $\Pr\left(\hat{a} \in (-t_{0.975}\hat{\sigma}_a, t_{0.975}\hat{\sigma}_a)\right) = 0.95$, o che $\Pr\left(\hat{b} \in (-t_{0.975}\hat{\sigma}_b, t_{0.975}\hat{\sigma}_b)\right) = 0.95$.

Gli intervalli di accettazione dell'ipotesi nulla $a = 0$, e $b = 0$, contro le ipotesi alternative $a \neq 0$, e $b \neq 0$, sono

```
H0_a_sup <- + t_0975 * sigma_hat_a
H0_a_inf <- - t_0975 * sigma_hat_a
c(H0_a_inf,H0_a_sup) # Intervallo di accettazione di H0
```

```
## [1] -0.3133772  0.3133772
```

```
a_hat # Ricordiamo valore di a_hat
```

```
## [1] 1.113618
```

```
a_hat > H0_a_inf & a_hat < H0_a_sup # Se FALSE H0 rigettata
```

```
## [1] FALSE
```

```
H0_b_sup <- + t_0975 * sigma_hat_b
H0_b_inf <- - t_0975 * sigma_hat_b
c(H0_b_inf,H0_b_sup) # Intervallo di accettazione di H0
```

```
## [1] -16.5425  16.5425
```

```
b_hat # Ricordiamo valore di b_hat
```

```
## [1] 80.32669
```

```
b_hat > H0_b_inf & b_hat < H0_b_sup # Se FALSE H0 rigettata
```

```
## [1] FALSE
```

Ancora, in alternativa si può valutare il p-value delle statistiche calcolate rispetto alla distribuzione di Student nell'ipotesi H_0 che i parametri siano nulli. Il p-value è la probabilità di osservare valori più anomali di quello osservato per la statistica utilizzata per il test d'ipotesi mentre è vera l'ipotesi nulla H_0. Nel caso della statistica t di Student, se t^* è il valore osservato, il p-value è la probabilità che $T_{n-2} \in (-\infty, -|t^*|) \cup (|t^*|, \infty)$

```
p_value_a <- (1 - pt(a_hat / sigma_hat_a, n-2)) * 2
p_value_a
```

```
## [1] 2.395771e-08
```

```
p_value_b <- (1 - pt(b_hat / sigma_hat_b, n-2)) * 2
p_value_b
```

```
## [1] 1.681477e-11
```

Si noti che i p-value ottenuti coincidono con quelli forniti dall'output di `lm`

```
summary(modello_regressione)$coefficients[, "Pr(>|t|)"]
```

```
##  (Intercept)        x_eta
## 1.681472e-11 2.395771e-08
```

Esempio 13.4

Si parta dagli esempi precedenti

```
pressione_df <- read.csv("pressione_arteriosa.csv")

x_eta <- pressione_df$eta
y_pressione <- pressione_df$pressione_arteriosa_mmHg
a_hat <- cov(x_eta,y_pressione) / var(x_eta)
b_hat <- mean(y_pressione) - a_hat * mean(x_eta)
y_hat <- a_hat * x_eta + b_hat
e <- y_pressione - y_hat
n <- length(y_pressione)
sigma_hat_squared <- 1 / (n - 2) * sum(e ^ 2)
sigma_hat <- sigma_hat_squared ^ 0.5
devianza_x <- sum( (x_eta - mean(x_eta))^2 )
t_0975 <- qt(0.975, n - 2)
```

Intervallo di confidenza della risposta media

Si valuti l'intervallo di confidenza della risposta media, in corrispondenza dei 70 anni di età.

```
x_eta_new <- 70
y_hat_new <- a_hat * x_eta_new + b_hat

# delta risposta media è la semiampiezza dell'intervallo
delta_risposta_media <- t_0975 * sigma_hat *
  ( 1 / n + (x_eta_new - mean(x_eta)) ^ 2 / devianza_x) ^ 0.5
y_sup_new_media <- y_hat_new + delta_risposta_media
y_inf_new_media <- y_hat_new - delta_risposta_media
c(y_inf_new_media, y_sup_new_media)
```

```
## [1] 151.8348 164.7252
```

La stessa cosa si può ottenere agevolmente in R

```
mod <- lm(pressione_arteriosa_mmHg ~ eta, data = pressione_df)
predict(mod, interval = "confidence") # in corrispondenza dei training data
```

```
##         fit      lwr      upr
## 1  117.0761 110.4148 123.7374
## 2  121.5306 115.9560 127.1052
## 3  122.6442 117.3297 127.9587
## 4  123.7578 118.6973 128.8183
## 5  123.7578 118.6973 128.8183
## 6  124.8714 120.0578 129.6850
## 7  127.0987 122.7531 131.4443
## 8  128.2123 124.0848 132.3398
## 9  129.3259 125.4035 133.2483
## 10 132.6668 129.2590 136.0745
## 11 132.6668 129.2590 136.0745
## 12 132.6668 129.2590 136.0745
## 13 133.7804 130.5018 137.0589
## 14 133.7804 130.5018 137.0589
## 15 134.8940 131.7188 138.0692
## 16 136.0076 132.9074 139.1079
## 17 136.0076 132.9074 139.1079
## 18 137.1212 134.0654 140.1771
## 19 138.2348 135.1915 141.2781
## 20 139.3485 136.2855 142.4115
## 21 139.3485 136.2855 142.4115
## 22 140.4621 137.3478 143.5764
## 23 141.5757 138.3799 144.7715
## 24 142.6893 139.3842 145.9944
## 25 142.6893 139.3842 145.9944
## 26 143.8029 140.3633 147.2426
## 27 143.8029 140.3633 147.2426
## 28 144.9166 141.3199 148.5132
## 29 148.2574 144.0825 152.4323
## 30 149.3710 144.9754 153.7667
## 31 150.4846 145.8575 155.1118
## 32 152.7119 147.5955 157.8283
## 33 152.7119 147.5955 157.8283
## 34 153.8255 148.4537 159.1973
## 35 156.0527 150.1532 161.9523
## 36 163.8481 155.9852 171.7109
```

```r
predict(mod, interval = "confidence",
        newdata = data.frame(eta = 70)) # in corrispondenza di eta = 70
```

```
##      fit      lwr      upr
## 1 158.28 151.8348 164.7252
```

Intervallo di confidenza della risposta singola

Si valuti l'intervallo di confidenza della risposta singola, in corrispondenza dei 70 anni di età.

```r
x_eta_new <- 70
y_hat_new <- a_hat * x_eta_new + b_hat

# delta risposta singola differisce per un 1 in più sotto radice
delta_risposta_singola <- t_0975 * sigma_hat *
  ( 1 + 1/n + (x_eta_new - mean(x_eta)) ^ 2 / devianza_x) ^ 0.5
y_sup_new_singola <- y_hat_new + delta_risposta_singola
```

```
y_inf_new_singola <- y_hat_new - delta_risposta_singola
c(y_inf_new_singola, y_sup_new_singola)
```

```
## [1] 138.9172 177.6428
```

La stessa cosa si può ottenere agevolmente in R

```
predict(mod, interval = "prediction") # in corrispondenza dei training data
```

```
## Warning in predict.lm(mod, interval = "prediction"): predictions on current data refer to _future_ re
```

```
##         fit       lwr      upr
## 1  117.0761  97.64032 136.5119
## 2  121.5306 102.43992 140.6212
## 3  122.6442 103.62786 141.6605
## 4  123.7578 104.81089 142.7047
## 5  123.7578 104.81089 142.7047
## 6  124.8714 105.98897 143.7539
## 7  127.0987 108.33005 145.8673
## 8  128.2123 109.49297 146.9316
## 9  129.3259 110.65074 148.0011
## 10 132.6668 114.09287 151.2406
## 11 132.6668 114.09287 151.2406
## 12 132.6668 114.09287 151.2406
## 13 133.7804 115.22976 152.3310
## 14 133.7804 115.22976 152.3310
## 15 134.8940 116.36136 153.4266
## 16 136.0076 117.48767 154.5275
## 17 136.0076 117.48767 154.5275
## 18 137.1212 118.60867 155.6338
## 19 138.2348 119.72436 156.7453
## 20 139.3485 120.83473 157.8622
## 21 139.3485 120.83473 157.8622
## 22 140.4621 121.93979 158.9844
## 23 141.5757 123.03954 160.1119
## 24 142.6893 124.13399 161.2446
## 25 142.6893 124.13399 161.2446
## 26 143.8029 125.22317 162.3827
## 27 143.8029 125.22317 162.3827
## 28 144.9166 126.30709 163.5260
## 29 148.2574 129.52759 166.9872
## 30 149.3710 130.59077 168.1513
## 31 150.4846 131.64887 169.3204
## 32 152.7119 133.74998 171.6738
## 33 152.7119 133.74998 171.6738
## 34 153.8255 134.79308 172.8579
## 35 156.0527 136.86468 175.2408
## 36 163.8481 143.96841 183.7277
```

```
predict(mod, interval = "prediction",
        newdata = data.frame(eta = 70)) # in corrispondenza di eta = 70
```

```
##      fit      lwr      upr
## 1 158.28 138.9172 177.6428
```

Bande di confidenza

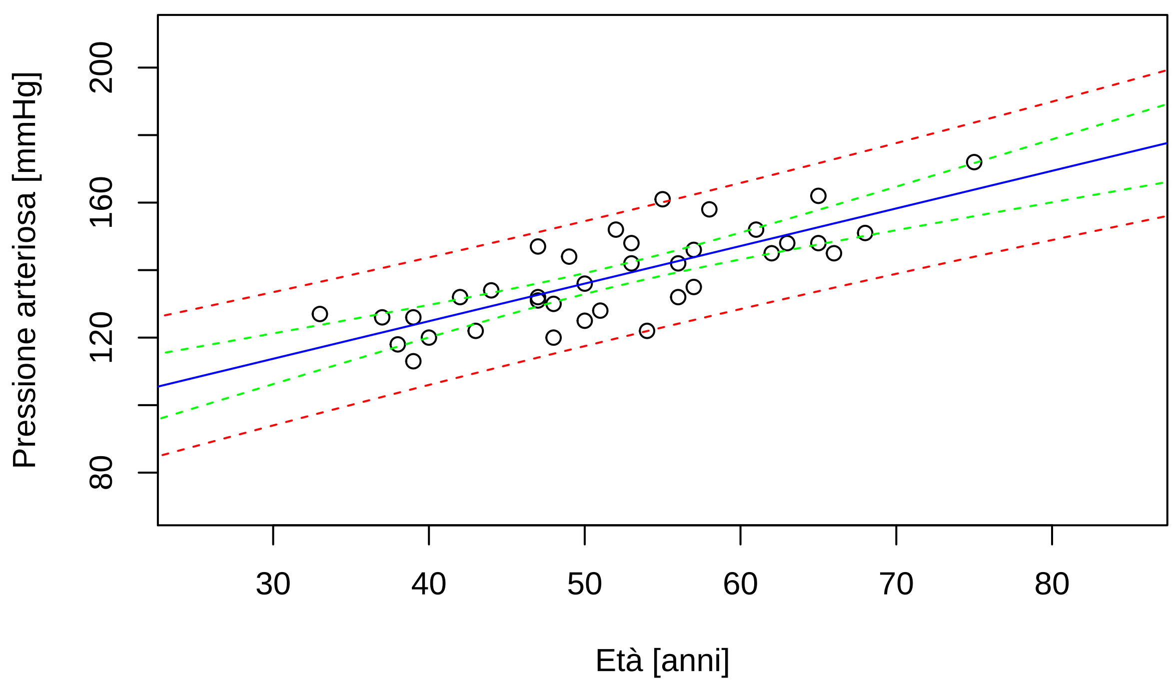
Più in generale, si può calcolare gli intervalli per ogni possibile nuova osservazione di x, invece che soltanto in corrispondenza di 70, e costruire le cosiddette bande di confidenza.

Si ricalcolino gli intervalli per ogni x, sostituendo al valore 70 assegnato alla variabile `x_eta_new`, un vettore di tutti possibili valori. Poi si tratta solo di ricopiare codice già scritto.

```
x_eta_new <- seq(from = min(x_eta) - 20, to = max(x_eta) + 20, length = 100)
y_hat_new <- a_hat * x_eta_new + b_hat

# Risposta media, il codice resta uguale
delta_risposta_media <- t_0975 * sigma_hat *
  ( 1/n + (x_eta_new - mean(x_eta))^2 / devianza_x)^0.5
y_sup_new_media <- y_hat_new + delta_risposta_media
y_inf_new_media <- y_hat_new - delta_risposta_media

# Risposta singola, il codice resta uguale
delta_risposta_singola <- t_0975 * sigma_hat *
  ( 1 + 1/n + (x_eta_new - mean(x_eta))^2 / devianza_x)^0.5
y_sup_new_singola <- y_hat_new + delta_risposta_singola
y_inf_new_singola <- y_hat_new - delta_risposta_singola
```

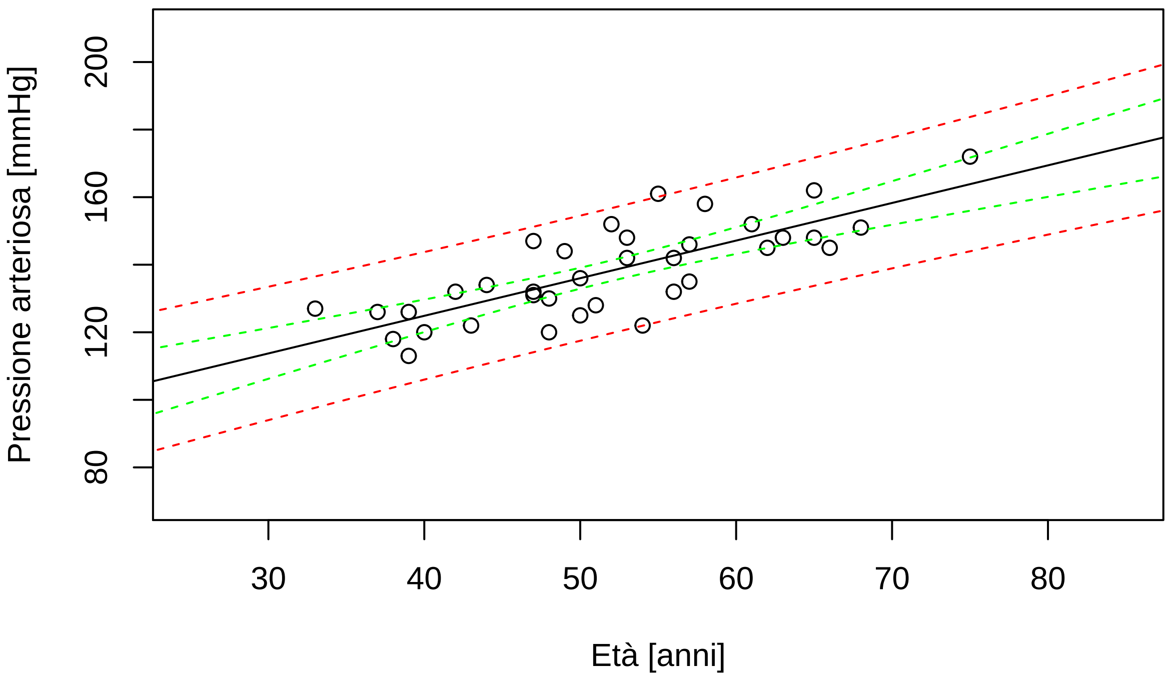
Ora possiamo mostrare i grafici

```
plot(x_eta, y_pressione,
     xlab = "Età [anni]", ylab = "Pressione arteriosa [mmHg]",
     ylim = c(70, 210), xlim = c(25, 85))
lines(x_eta_new, y_hat_new, col="blue")
lines(x_eta_new, y_sup_new_media, col="green", lty = 2)
lines(x_eta_new, y_inf_new_media, col="green", lty = 2)
lines(x_eta_new, y_sup_new_singola, col="red", lty = 2)
lines(x_eta_new, y_inf_new_singola, col="red", lty = 2)
```



La stessa cosa si può ottenere agevolmente in R

```
plot(x_eta, y_pressione,
     xlab = "Età [anni]", ylab = "Pressione arteriosa [mmHg]",
     ylim = c(70, 210), xlim = c(25, 85))
abline(a = mod$coefficients[1], # intercept
       b = mod$coefficients[2]) # slope
conf_interval <- predict(mod, interval = "confidence",
                         newdata = data.frame(eta = x_eta_new))
lines(x_eta_new, conf_interval[, 2], lty = 2, col = "green")
lines(x_eta_new, conf_interval[, 3], lty = 2, col = "green")
pred_interval <- predict(mod, interval = "prediction",
                         newdata = data.frame(eta = x_eta_new))
lines(x_eta_new, pred_interval[, 2], lty = 2, col = "red")
lines(x_eta_new, pred_interval[, 3], lty = 2, col = "red")
```

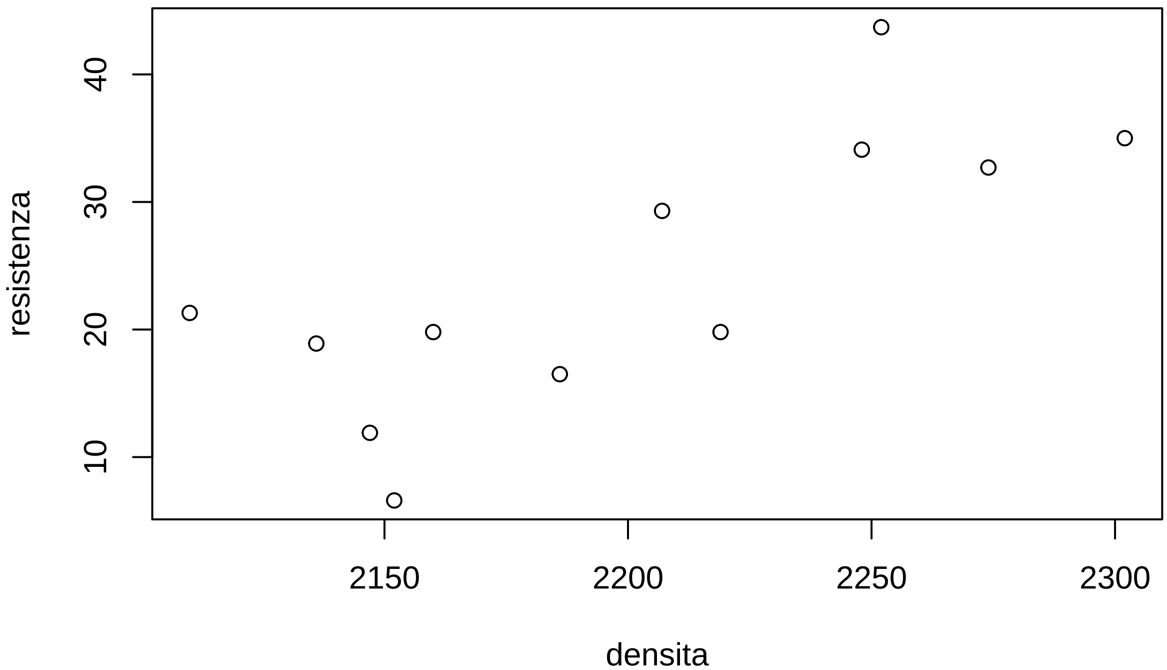


Esempio 13.5

Questo è un nuovo esempio. È noto che la densita a rottura $\sigma(N/mm^2)$ del calcestruzzo dipende dalla sua densità $\delta(kg/m^3)$. Tant'è che per realizzare buoni calcestruzzi vengono impiegati additivi fluidificanti che, riducendo del 20-30% la quantità d'acqua necessaria per l'impasto, ne riducono la porosità migliorandone la densita. Pur esistendo, questo legame funzionale non è completamente noto. Tuttavia, allorché non sia possibile estrarre provini idonei per le prove meccaniche, esso costituisce l'unico strumento disponibile per esprimere un parere sull'idoneità di un determinato calcestruzzo già da tempo in opera. Ciò premesso, siamo riusciti a procurarci ben 12 (ed è tanto!) coppie di misure sperimentali (di σ e δ) su altrettanti provini di calcestruzzo di un fabbricato per civili abitazioni crollato a tre anni dalla sua sopraelevazione di ulteriori due piani.

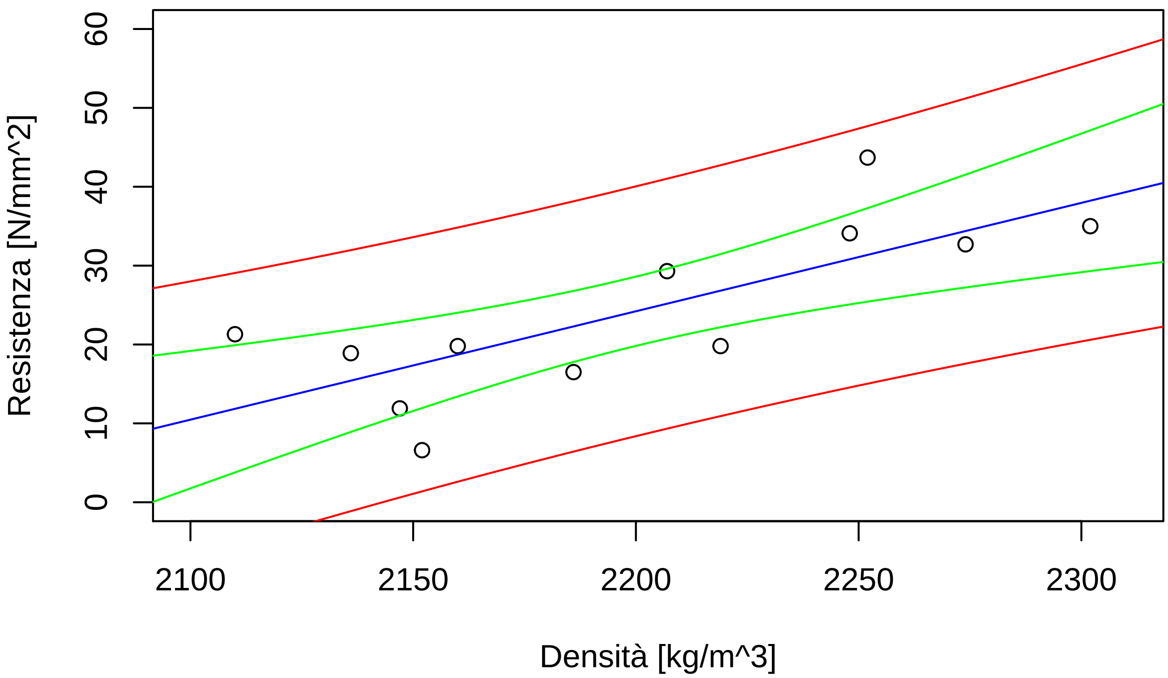
Carichiamo i nuovi dati e ripetiamo i calcoli già svolti per gli esempi precedenti, copiando il codice già scritto e adattando i nomi alle nuove variabili in gioco.

```
# Lettura dati
calcestruzzo_df <- read.csv("calcestruzzo.csv")
plot(calcestruzzo_df)
```




```
# Stime
x_densita <- calcestruzzo_df$densita
y_resistenza <- calcestruzzo_df$resistenza
a_hat <- cov(x_densita,y_resistenza) / var(x_densita)
b_hat <- mean(y_resistenza) - a_hat * mean(x_densita)
y_hat <- a_hat * x_densita + b_hat
e <- y_resistenza - y_hat
n <- length(y_resistenza)
sigma_hat_squared <- 1/(n-1) * sum(e^2)
sigma_hat <- sigma_hat_squared^0.5
devianza_x <- sum( (x_densita - mean(x_densita))^2 )

# Nuove osservazioni
x_densita_new <- seq(from = min(x_densita) - 20,
                     to = max(x_densita) + 20, length = 100)
y_hat_new <- a_hat * x_densita_new + b_hat

# Risposta media
t_0975 <- qt(0.975, n - 1)
delta_risposta_media <- t_0975 * sigma_hat *
  ( 1/n + (x_densita_new - mean(x_densita))^2 / devianza_x)^0.5
y_sup_new_media <- y_hat_new + delta_risposta_media
y_inf_new_media <- y_hat_new - delta_risposta_media

# Risposta singola
delta_risposta_singola <- t_0975 * sigma_hat *
  ( 1 + 1/n + (x_densita_new - mean(x_densita))^2 / devianza_x)^0.5
y_sup_new_singola <- y_hat_new + delta_risposta_singola
y_inf_new_singola <- y_hat_new - delta_risposta_singola

# Grafici
plot(x_densita, y_resistenza,
```

```
     xlab="Densità [kg/m^3]", ylab="Resistenza [N/mm^2]",
     xlim = c(2100, 2310), ylim=c(0, 60))
lines(x_densita_new, y_hat_new, col="blue")
lines(x_densita_new, y_sup_new_media, col="green")
lines(x_densita_new, y_inf_new_media, col="green")
lines(x_densita_new, y_sup_new_singola, col="red")
lines(x_densita_new, y_inf_new_singola, col="red")
```



Cosa fare se i dati sono sparsi ed il Giudice ha posto, tra i numerosi altri, il seguente specifico quesito? "... Accerti il CTU (consulente tecnico d'ufficio, N.d.R.) se vi siano concause diverse dalla mancanza di tenuta del calcestruzzo e determini la fondatezza del valore di resistenza di 20 N/mm^2, assunto dal progettista come dato di progetto per la sopraelevazione, sulla base del valore di densità di 2202 kg/m^3 misurato sull'opera preesistente."

La stima del coefficiente di correlazione è $\hat{\rho} = 0.77$ ed è in accordo con quanto già noto in letteratura. Il valore (medio) di risposta che valutiamo in base alla regressione dei nostri dati è $\hat{y}_{new} = 0.14 \cdot 2202 - 278.31 = 30$ ed è nettamente superiore a quello utilizzato nel progettare la sopraelevazione. D'altro canto, però, la prassi professionale vorrebbe che fosse assunto come carico di sicurezza (valido per la progettazione) almeno la metà di quello misurato sui provini, cioè 15 N/mm^2. Come uscirne? Quale parere esprimere in tutta coscienza ed in una forma comprensibile al magistrato? L'unica maniera è valutare il rischio d'errore connesso alla valutazione fatta dal progettista. Abbiamo:

$t = \frac{20-30}{7.253\sqrt{1.084}} = -1.3, \quad \nu = 10$

cui corrisponde una coda sinistra di probabilità pari a 0.11, cioè in 11 casi su 100 l'assunzione risulta errata verificandosi una resistenza minore di 20 N/mm^2. Quindi, in base alle 12 coppie di dati in nostro possesso, possiamo affermare che alla scelta del progettista (di assumere il valore di resistenza di 20 N/mm^2) è connesso un rischio d'errore pari all'11%. Viceversa se il progettista avesse seguito la prassi professionale, assumendo un valore pari a 15 N/mm^2, la statistica sarebbe risultata pari a -1.95 ed il rischio sarebbe stato circa del 4%. A verifica di quanto affermato, in figura notiamo che a 2202 kg/m^3 corrisponde un intervallo di confidenza al livello 0.90 uguale a (10.8, 38.2).

Esempio su regressione multipla (dati dal libro di Montgomery, Introduction to Linear Regression)

Un venditore di bibite vuole analizzare il servizio di fornitura delle macchinette automatiche. E' interessato alla previsione del tempo necessario al corriere per servire le macchinette distributrici in un outlet. Il servizio consiste nel rifornire le macchinette con le bibite e in attività di manutenzione minore. L'ingegnere industriale responsabile di questo studio suggerisce che le due varaibili più importanti che influenzano il tempo di fornitura (y) sono il numero di scatole di prodotti da stoccare (x_1) e la distanza percorsa dal corriere (x_2). L'ingegnere ha raccolto 25 osservazioni sul tempo di fornitura. Il modello di regressione ipotizzato è

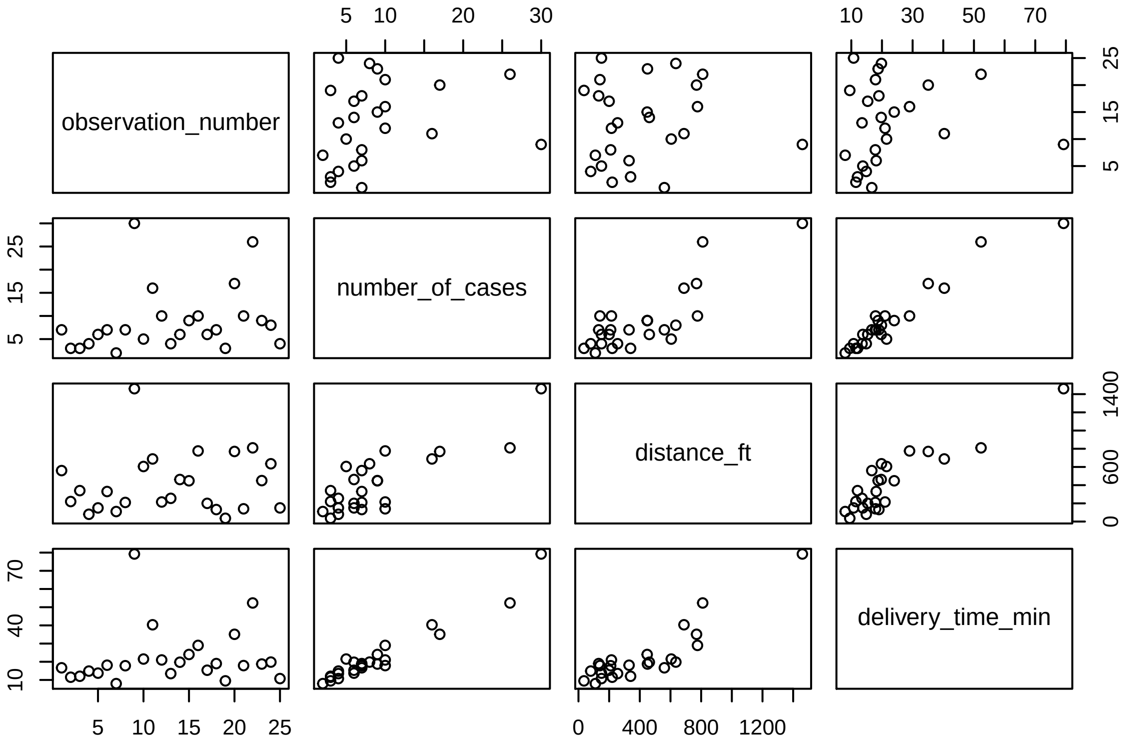
$y_i = a_0 + a_1 x_{1i} + a_2 x_{2i} + \varepsilon_i$, $i = 1, 2, \ldots, 25$.

I dati sono disponibili nel file delivery_time_data.csv. Possiamo leggerli attraverso la funzione read.csv

```
delivery_data <- read.csv("delivery_time_data.csv")
delivery_data
```

```
##    observation_number number_of_cases distance_ft delivery_time_min
## 1                   1               7         560             16.68
## 2                   2               3         220             11.50
## 3                   3               3         340             12.03
## 4                   4               4          80             14.88
## 5                   5               6         150             13.75
## 6                   6               7         330             18.11
## 7                   7               2         110              8.00
## 8                   8               7         210             17.83
## 9                   9              30        1460             79.24
## 10                 10               5         605             21.50
## 11                 11              16         688             40.33
## 12                 12              10         215             21.00
## 13                 13               4         255             13.50
## 14                 14               6         462             19.75
## 15                 15               9         448             24.00
## 16                 16              10         776             29.00
## 17                 17               6         200             15.35
## 18                 18               7         132             19.00
## 19                 19               3          36              9.50
## 20                 20              17         770             35.10
## 21                 21              10         140             17.90
## 22                 22              26         810             52.32
## 23                 23               9         450             18.75
## 24                 24               8         635             19.83
## 25                 25               4         150             10.75
```

```
plot(delivery_data)
```



La variabile di risposta y è la colonna delivery_time_min, mentre le due variabili indipendenti, o regressori, o covariate, x_1 e x_2 sono rispettivamente le colonne number_of_cases e distance_ft.

Il comando per applicare il metodo di regressione ai dati è la funzione lm. Per prima cosa, però, costruiamo un modello di regressione semplice, considerando come unica variabile indipendente x_1:

```
mod_simple <- lm(delivery_time_min ~ number_of_cases, data = delivery_data)
summary(mod_simple)
```

```
##
## Call:
## lm(formula = delivery_time_min ~ number_of_cases, data = delivery_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.5811 -1.8739 -0.3493  2.1807 10.6342
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)        3.321      1.371   2.422   0.0237 *
## number_of_cases    2.176      0.124  17.546 8.22e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.181 on 23 degrees of freedom
## Multiple R-squared:  0.9305, Adjusted R-squared:  0.9275
## F-statistic: 307.8 on 1 and 23 DF,  p-value: 8.22e-15
```

Questo lo facciamo per confrontare il risultato della regressione semplice, con una sola variabile, con quello di regressione multipla in cui aggiungiamo una variabile indipendente a quella già considerata.

```
mod_multiple <- lm(delivery_time_min ~ number_of_cases + distance_ft,
                   data = delivery_data)
summary(mod_multiple)
```

```
##
## Call:
## lm(formula = delivery_time_min ~ number_of_cases + distance_ft,
##     data = delivery_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7880 -0.6629  0.4364  1.1566  7.4197
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     2.341231   1.096730   2.135 0.044170 *
## number_of_cases 1.615907   0.170735   9.464 3.25e-09 ***
## distance_ft     0.014385   0.003613   3.981 0.000631 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.259 on 22 degrees of freedom
## Multiple R-squared:  0.9596, Adjusted R-squared:  0.9559
## F-statistic: 261.2 on 2 and 22 DF,  p-value: 4.687e-16
```

Infatti, com'è possibile vedere, la stima del coefficiente relativo alla variabile numero di scatole, nel modello di regressione semplice è pari a 2.176, mentre in quello di regressione multipla cambia e diventa pari a 1.615907, perché la stima viene influenzata dall'inserimento della nuova variabile nel modello.

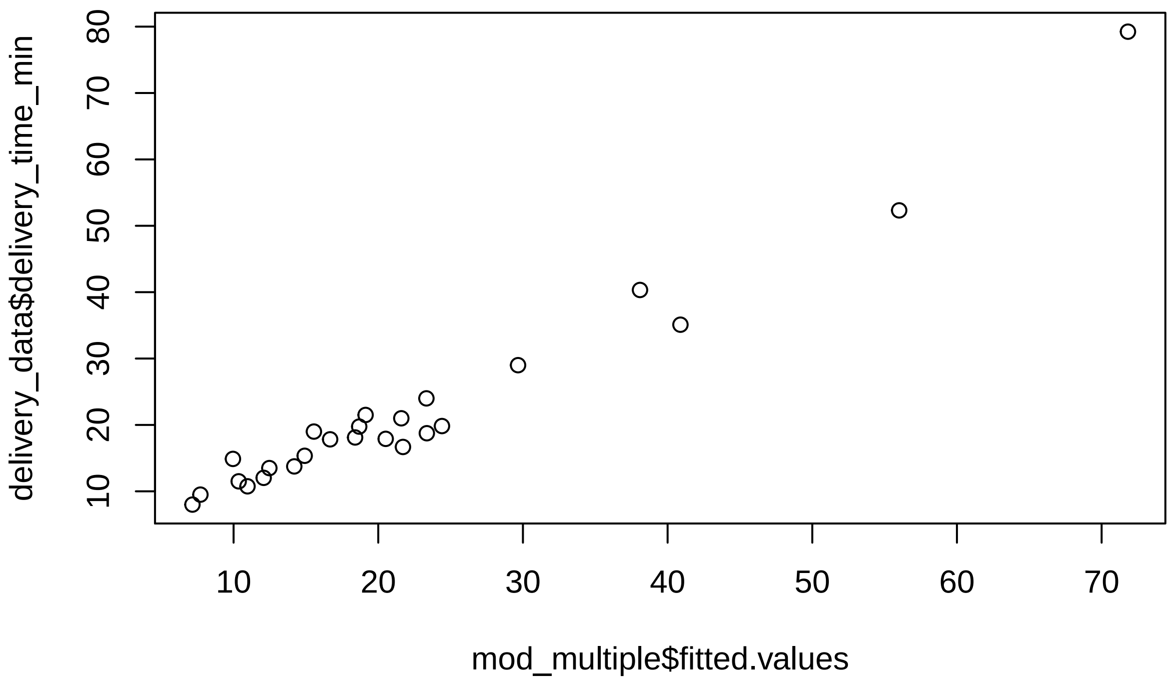
Variabili ortogonali

Nota bene che ciò avviene perché i regressori non sono ortogonali tra loro. Infatti, se i regressori fossero ortogonali, le stime dei coefficienti relativi non sarebbero influenzate dalla presenza delle altre variabili. In altre parole, aggiungendo o sottraendo regressori ortogonali a un modello di regressione, la stima dei coefficienti di regressione relativi alle variabili presenti nel modello non cambia. Questo è il caso dell'analisi della varianza nella progettazione degli esperimenti, in cui l'aggiunta di ulteriori fattori non cambia la stima dell'effetto dei fattori già considerati. Questo avviene perché i fattori sono sempre presenti in maniera bilanciata (infatti si ha il piano ortogonale).

Grafici

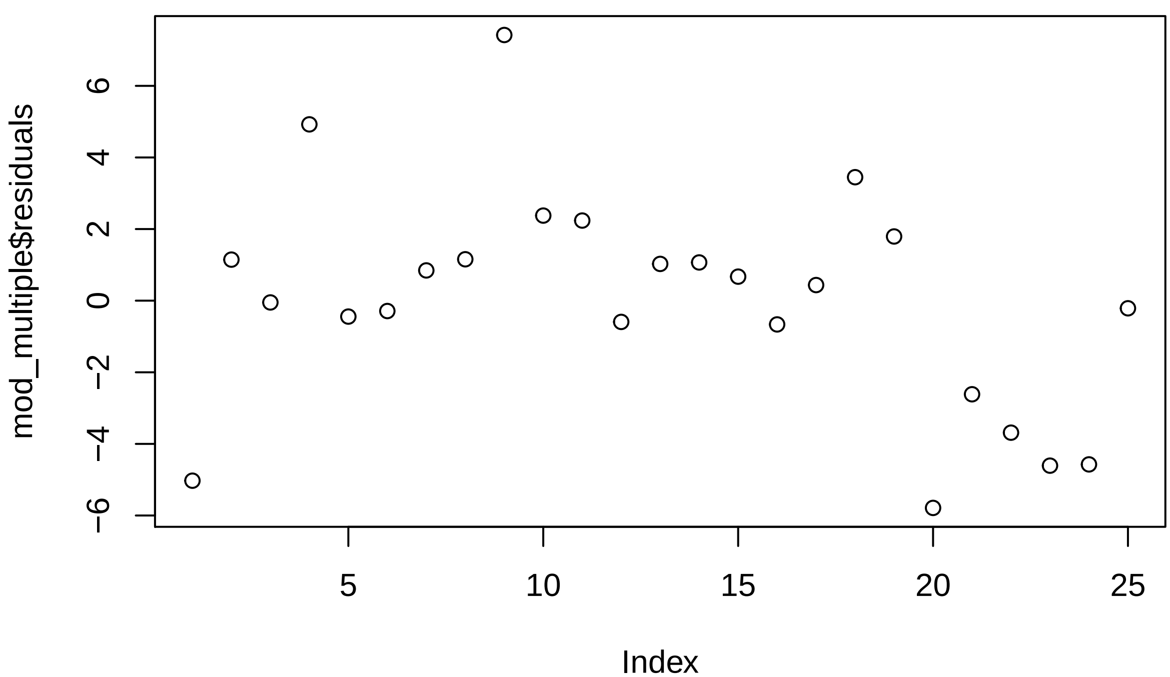
Valori previsti della variabile di risposta ($\hat{y}$) vs valori osservati (y)

```
plot(mod_multiple$fitted.values, delivery_data$delivery_time_min)
```



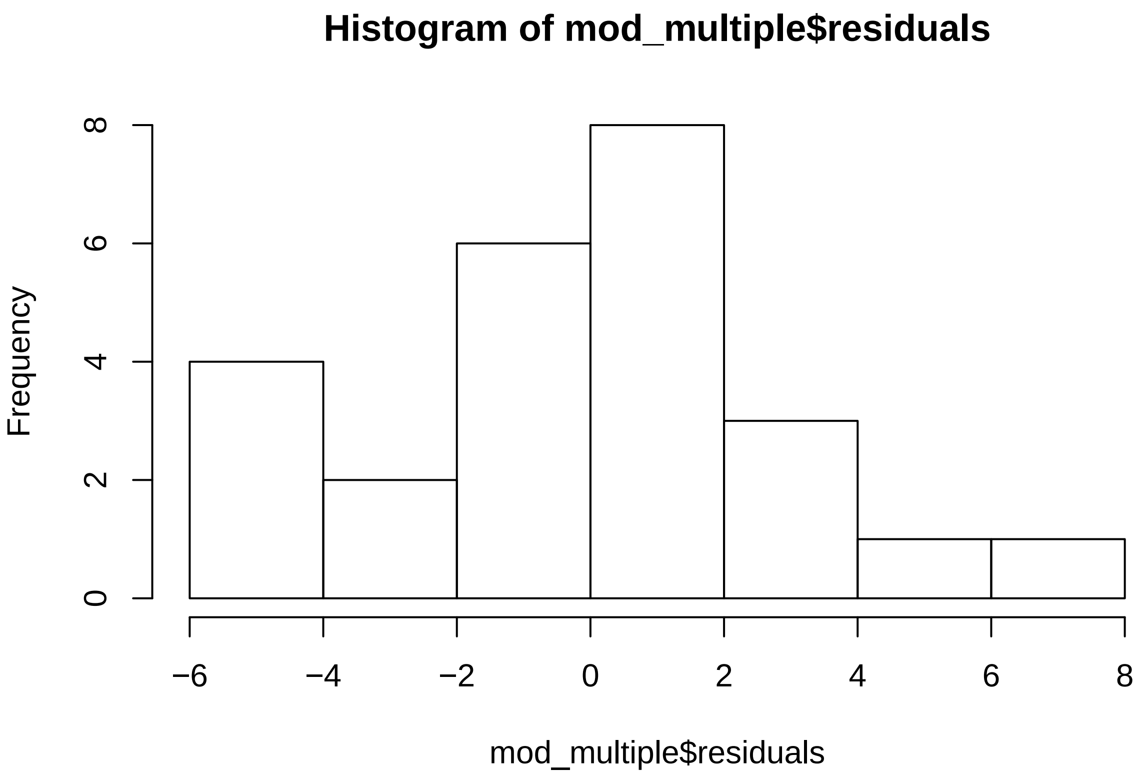
Residui vs numero osservazioni

```r
plot(mod_multiple$residuals)
```



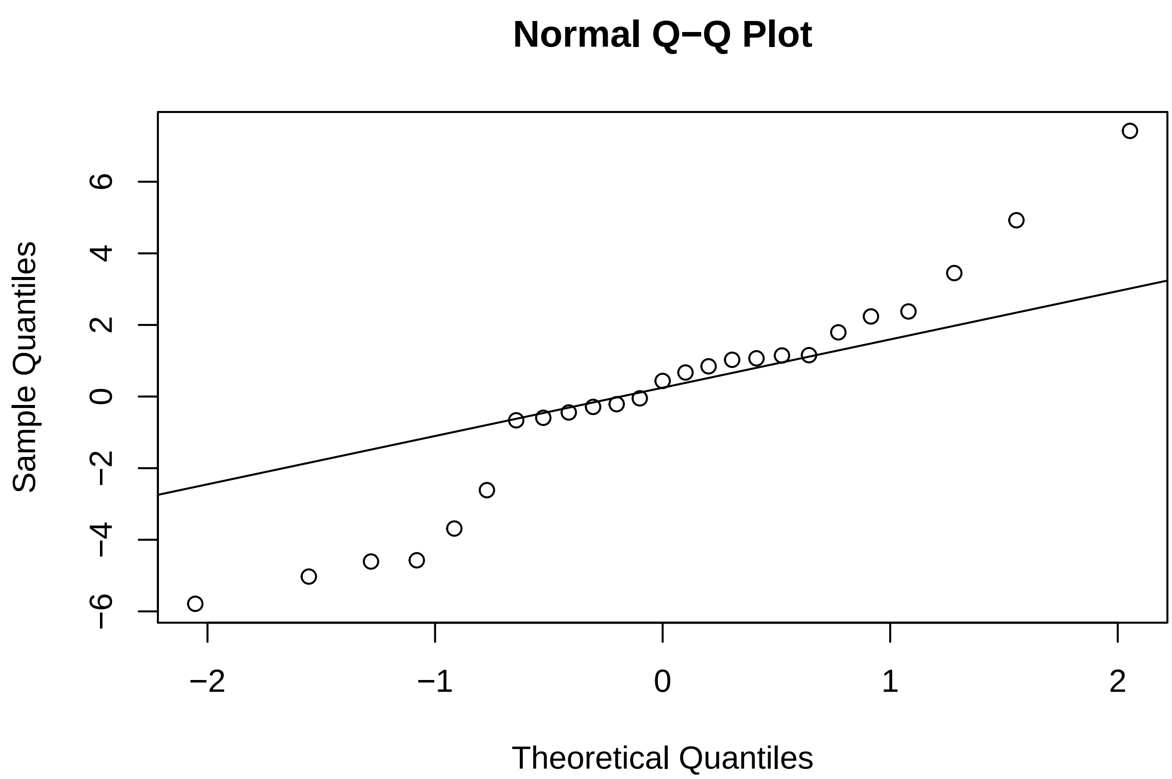
Istogramma dei residui

```r
hist(mod_multiple$residuals)
```



Carta di probabilità normale dei residui

```
qqnorm(mod_multiple$residuals)
qqline(mod_multiple$residuals)
```



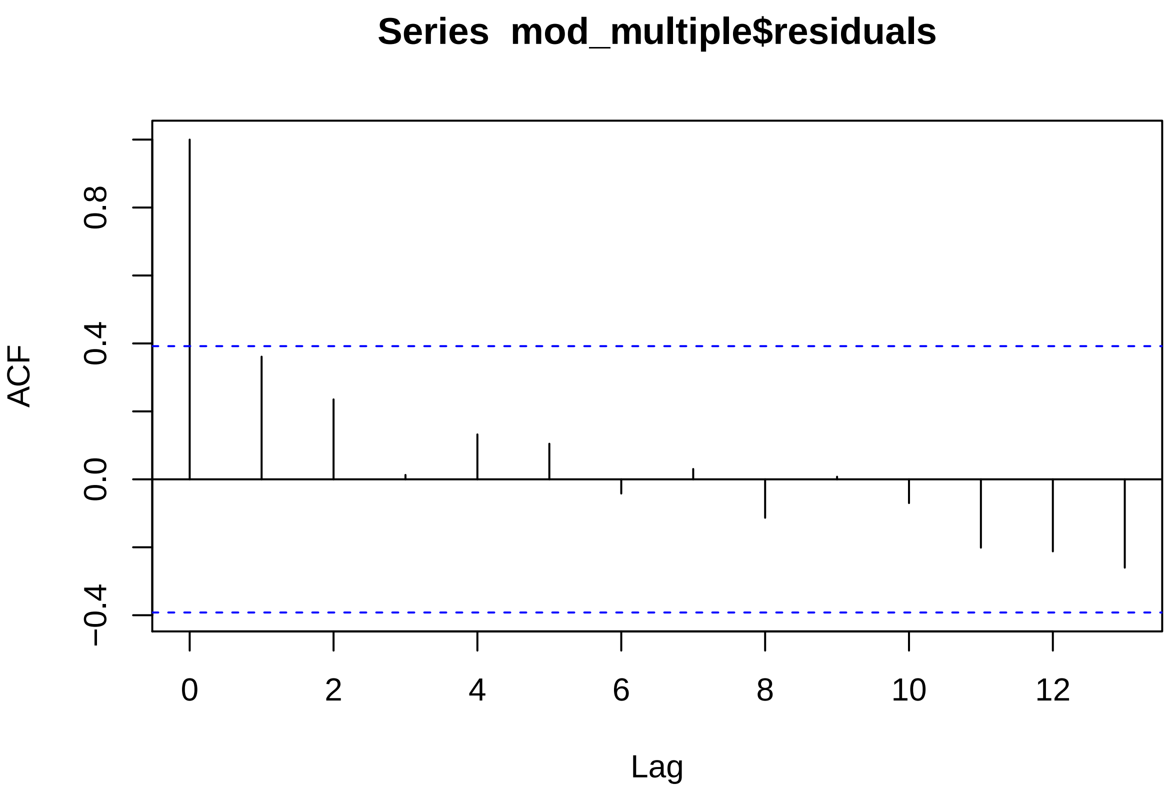
Test di normalità di Shapiro-Wilk

```
shapiro.test(mod_multiple$residuals)
```

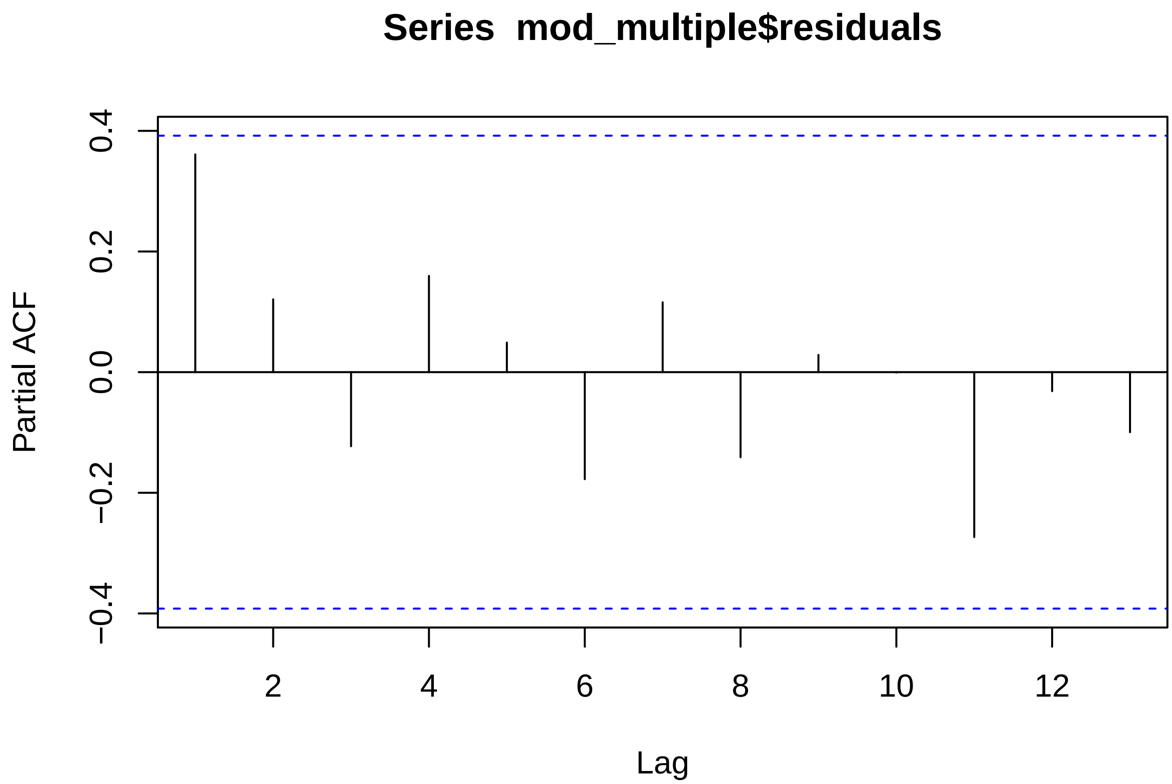
```
##
##  Shapiro-Wilk normality test
##
## data:  mod_multiple$residuals
## W = 0.95151, p-value = 0.2711
```

Autocorrelazione dei residui

```
acf(mod_multiple$residuals)
```




```
pacf(mod_multiple$residuals)
```



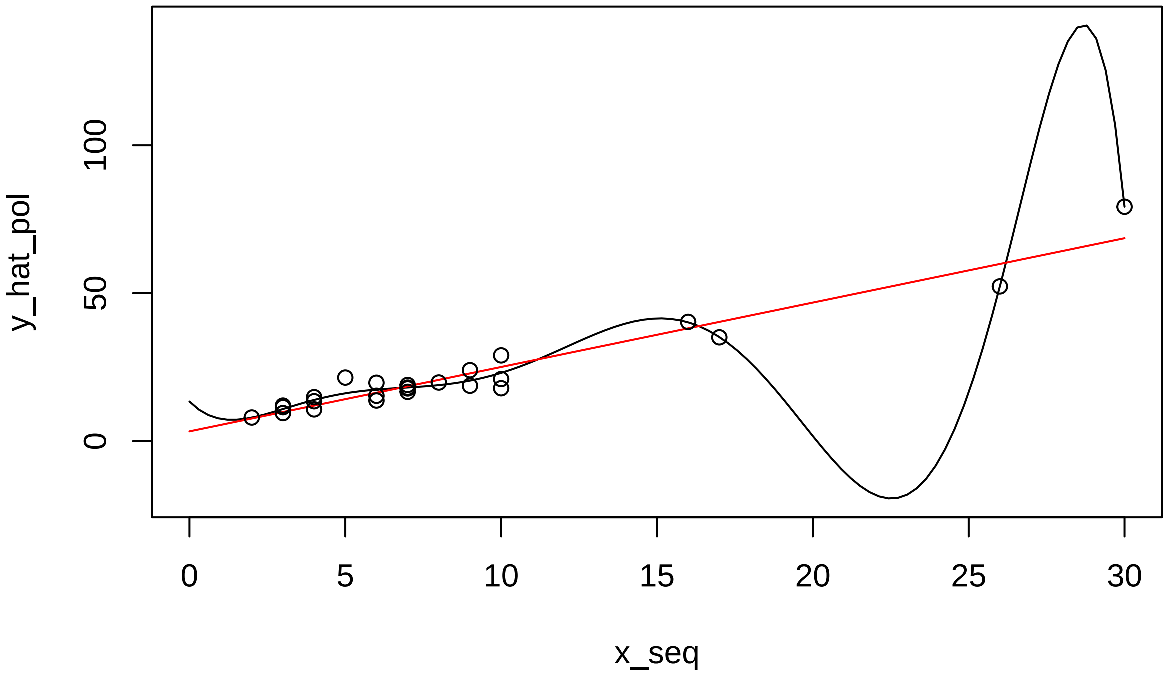
Overfitting

```
delivery_data$x <- delivery_data$number_of_cases
mod_polynomial <- lm(delivery_time_min ~ x + I(x ^ 2) + I(x ^ 3) +
                       I(x ^ 4) + I(x ^ 5) + I(x ^ 6) + I(x ^ 7) +
                       I(x ^ 8), data = delivery_data)
summary(mod_polynomial)
```

```
##
## Call:
## lm(formula = delivery_time_min ~ x + I(x^2) + I(x^3) + I(x^4) +
##     I(x^5) + I(x^6) + I(x^7) + I(x^8), data = delivery_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.1852 -1.4733 -0.0001  0.8806  5.9148
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.341e+01  6.981e+01   0.192    0.850
## x           -1.071e+01  8.004e+01  -0.134    0.895
## I(x^2)       5.873e+00  3.565e+01   0.165    0.871
## I(x^3)      -1.097e+00  8.087e+00  -0.136    0.894
## I(x^4)       8.223e-02  1.031e+00   0.080    0.937
## I(x^5)      -6.888e-04  7.651e-02  -0.009    0.993
## I(x^6)      -2.127e-04  3.253e-03  -0.065    0.949
## I(x^7)       9.888e-06  7.310e-05   0.135    0.894
## I(x^8)      -1.316e-07  6.695e-07  -0.197    0.847
##
```

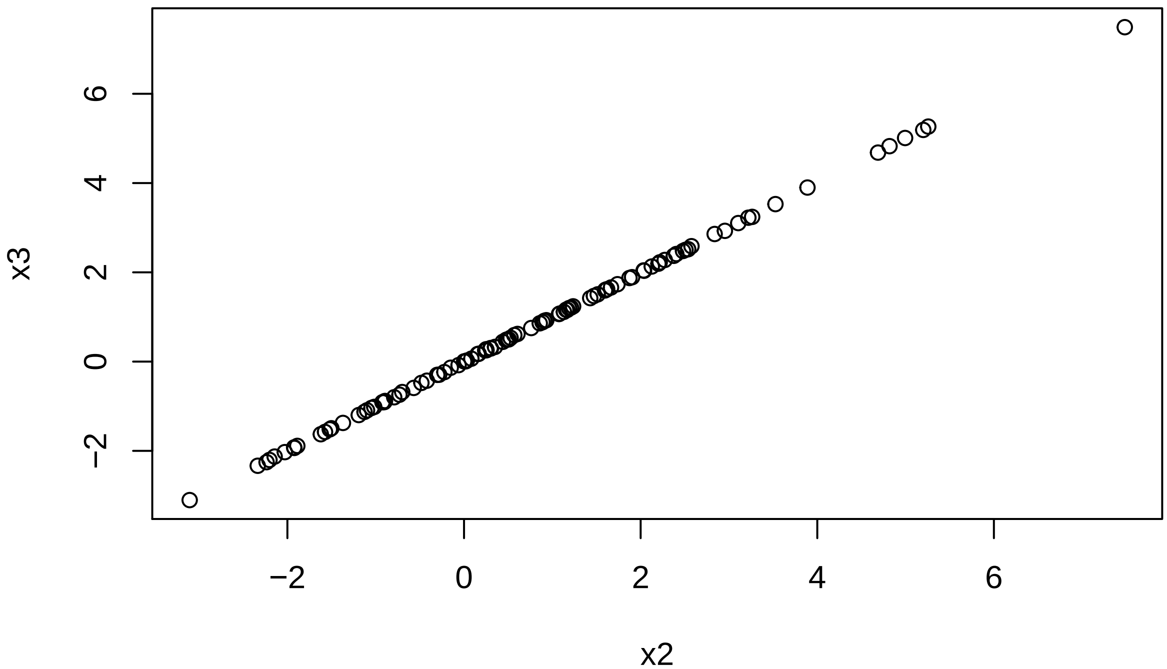
```
## Residual standard error: 3.082 on 16 degrees of freedom
## Multiple R-squared:  0.9737, Adjusted R-squared:  0.9606
## F-statistic:  74.1 on 8 and 16 DF,  p-value: 3.497e-11
```

```
x_seq <- seq(from = 0, to = 30, length = 100)
y_hat_pol <- predict(mod_polynomial, newdata = data.frame(x = x_seq))
y_hat_lin <- predict(mod_simple, newdata = data.frame(number_of_cases = x_seq))
plot(x = x_seq, y = y_hat_pol, type = "l")
lines(x = x_seq, y = y_hat_lin, col = "red")
points(delivery_data$x, delivery_data$delivery_time_min)
```



Multicollinearity

```
## Simulate some data
# Simulate data
n <- 100
a1 <- 2
a0 <- 1
a2 <- 3
set.seed(123)
x1 <- rnorm(n)
x2 <- rnorm(n, mean = 1, sd = 2)
eps <- rnorm(n, sd = 0.6)
y <- a0 + a1 * x1 + a2 * x2 + eps

# Multicollinearity
# This can happen when variables are strongly correlated
x3 <- x2 + rnorm(n, sd = 0.01) # not exact linear dependence
plot(x2, x3)
```




```r
a3 <- 1
y <- a0 + a1 * x1 + a2 * x2 + a3 * x3 + eps
df <- data.frame(y, x1, x2, x3)
mod <- lm(y ~ ., df)
X <- model.matrix(mod)
qr(X)$rank              # X is still full rank!
```

```
## [1] 4
```

```r
summary(mod)            # see the high standard deviation of estimates!
```

```
##
## Call:
## lm(formula = y ~ ., data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.14920 -0.39388 -0.09137  0.38386  1.23819
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.07249    0.06232  17.209   <2e-16 ***
## x1           1.91874    0.06322  30.351   <2e-16 ***
## x2           5.84758    5.55508   1.053    0.295
## x3          -1.83980    5.55376  -0.331    0.741
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.573 on 96 degrees of freedom
## Multiple R-squared:  0.9949, Adjusted R-squared:  0.9947
## F-statistic:  6211 on 3 and 96 DF,  p-value: < 2.2e-16
```

```r
                        # p-values say variables are not significant!
                        # but we have simulated from a model!
```

```
# Variance inflation factor 1 / (1- R2_j) - one covariate against others
library(car)
```

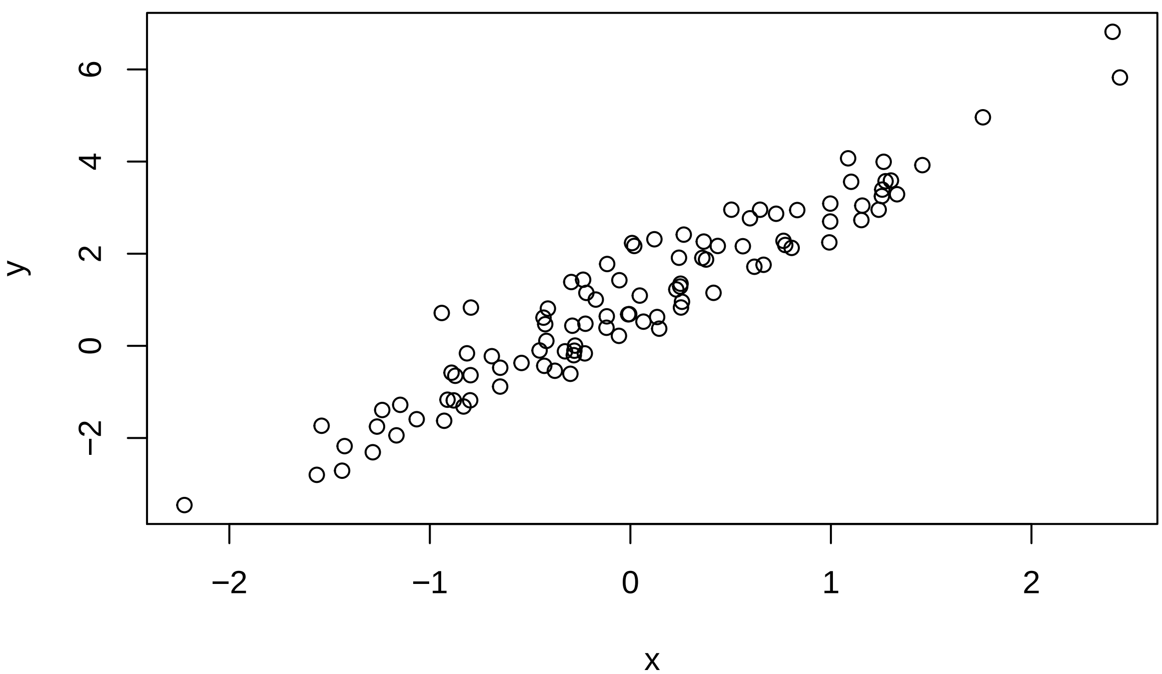
```
## Loading required package: carData
```

```
vif(mod)      # Here you see that x2 and x3 are strongly correlated
```

```
##           x1           x2           x3
##     1.004231 34806.630281 34807.407460
```

Leverage Influence

```
# Leverage and influence: helps identifying outliers
# See also this useful link:
# http://omaymas.github.io/InfluenceAnalysis/

# Consider the simple linear regression simulated example
n <- 100
a <- 2
b <- 1
set.seed(0)
x <- rnorm(n)
eps <- rnorm(n, sd = 0.6)
y <- b + a * x + eps
df <- data.frame(x, y)
plot(x, y)
```

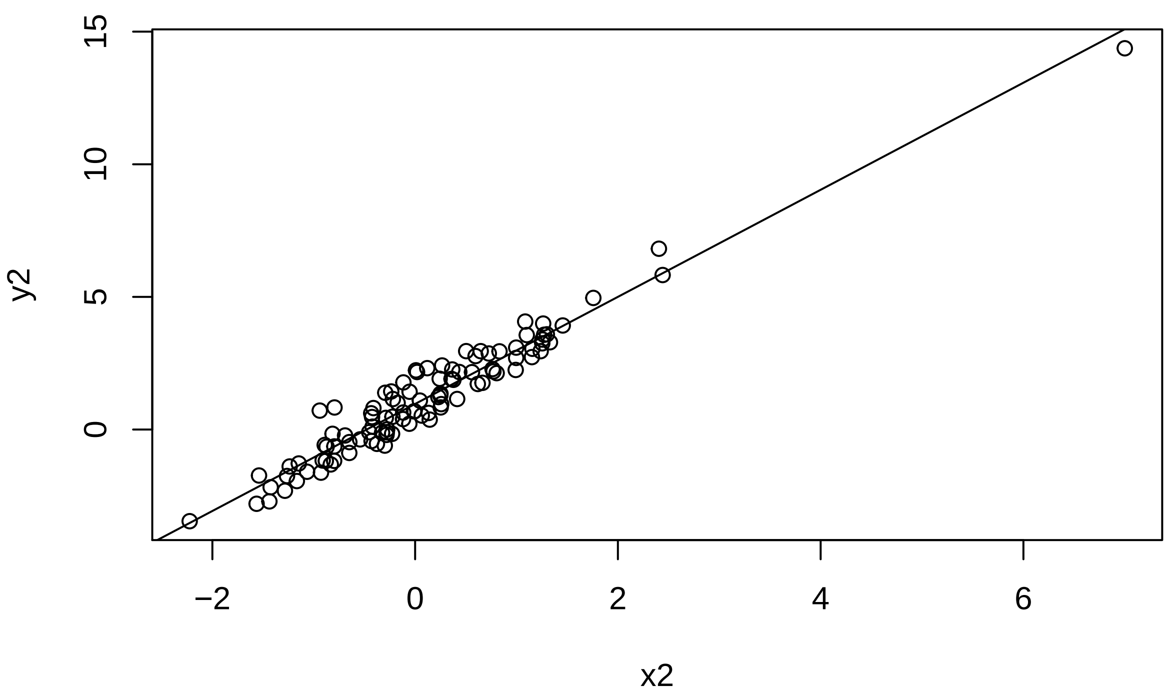



```
mod <- lm(y ~ x, df)
summary(mod)
```

```
##
## Call:
## lm(formula = y ~ x, data = df)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.95397 -0.48919 -0.09186  0.38275  1.70275
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.97078    0.05777   16.80   <2e-16 ***
## x            2.08328    0.06576   31.68   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5775 on 98 degrees of freedom
## Multiple R-squared:  0.911,  Adjusted R-squared:  0.9101
## F-statistic:  1004 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
# Now add a new point with high leverage, with no influence
x2 <- c(x, 7)
# and generate according to the model
eps2 <- c(eps, rnorm(1, sd = 0.6)) # add a new error
y2 <- b + a * x2 + eps2             # and generate y
plot(x2, y2)                        # See the leverage point!
df2 <- data.frame(x2, y2)
mod_lev <- lm(y2 ~ x2, df2)
abline(mod_lev)
```




```
hatvalues(mod_lev)
```

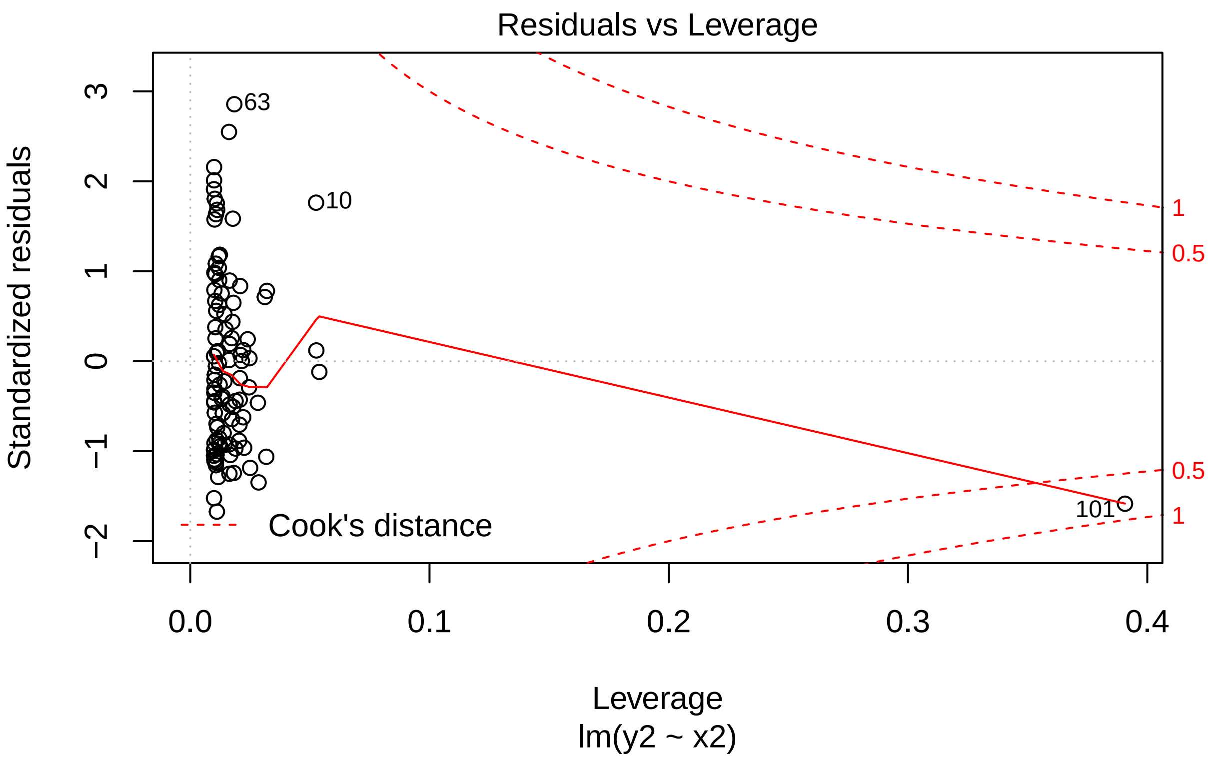
```
##           1           2           3           4           5           6
## 0.020845903 0.011295006 0.022130890 0.021023708 0.010732865 0.031144626
##           7           8           9          10          11          12
## 0.018207504 0.011092733 0.009976869 0.052584717 0.013502484 0.016231946
##          13          14          15          16          17          18
```

```
## 0.022157766 0.011060520 0.011120614 0.011921847 0.010106460 0.017621543
##          19          20          21          22          23          24
## 0.010844820 0.023999940 0.010697835 0.010552019 0.009914789 0.013950874
##          25          26          27          28          29          30
## 0.010077793 0.011254435 0.017784808 0.014789131 0.025015903 0.009917165
##          31          32          33          34          35          36
## 0.010756564 0.013114662 0.012100711 0.014284732 0.013118315 0.018868899
##          37          38          39          40          41          42
## 0.016369850 0.012069010 0.020390039 0.010999801 0.032051158 0.011656020
##          43          44          45          46          47          48
## 0.012266905 0.016710223 0.022534701 0.020588358 0.031769711 0.018947320
##          49          50          51          52          53          54
## 0.014273781 0.010713345 0.010143636 0.011651969 0.053950454 0.016179884
##          55          56          57          58          59          60
## 0.010072537 0.010101163 0.012112719 0.010458672 0.052686232 0.024558508
##          61          62          63          64          65          66
## 0.010469710 0.009985305 0.018405392 0.010244788 0.016460838 0.010081746
##          67          68          69          70          71          72
## 0.028259296 0.010500852 0.010096817 0.009906578 0.009943039 0.010119768
##          73          74          75          76          77          78
## 0.014279274 0.010255952 0.012515100 0.018027757 0.009922582 0.010251205
##          79          80          81          82          83          84
## 0.017941041 0.028562819 0.016204687 0.020680728 0.013594720 0.010674049
##          85          86          87          88          89          90
## 0.012030067 0.011982278 0.016439387 0.010978772 0.020716663 0.012358039
##          91          92          93          94          95          96
## 0.021535983 0.017331422 0.009956462 0.017449070 0.011931869 0.009907231
##          97          98          99         100         101
## 0.011016610 0.024751031 0.010051336 0.016432990 0.390689358
```

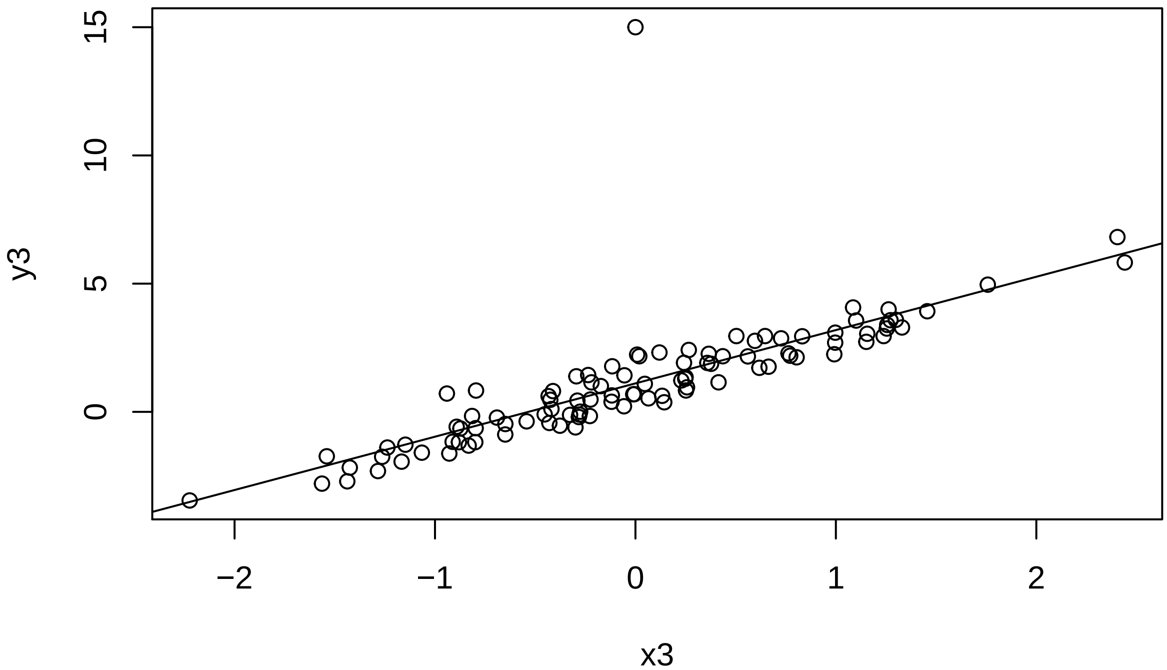
```r
summary(mod_lev)
```

```
##
## Call:
## lm(formula = y2 ~ x2, data = df2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9677 -0.5064 -0.1087  0.3876  1.6472
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.96506    0.05811   16.61   <2e-16 ***
## x2           2.01817    0.05199   38.82   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.582 on 99 degrees of freedom
## Multiple R-squared:  0.9384, Adjusted R-squared:  0.9377
## F-statistic:  1507 on 1 and 99 DF,  p-value: < 2.2e-16
```

```r
plot(mod_lev, which = 5)                    # see the leverage point in last plot
```




```
# Note: here the leverage point is according to the model. It is not an outlier

# Now add a new point with far from others, but with no leverage
x3 <- c(x, 0)
y3 <- c(y, 15)
plot(x3, y3)
df3 <- data.frame(x3, y3)
mod_inf <- lm(y3 ~ x3, df3)
abline(mod_inf)
```




```r
hatvalues(mod_inf)
```

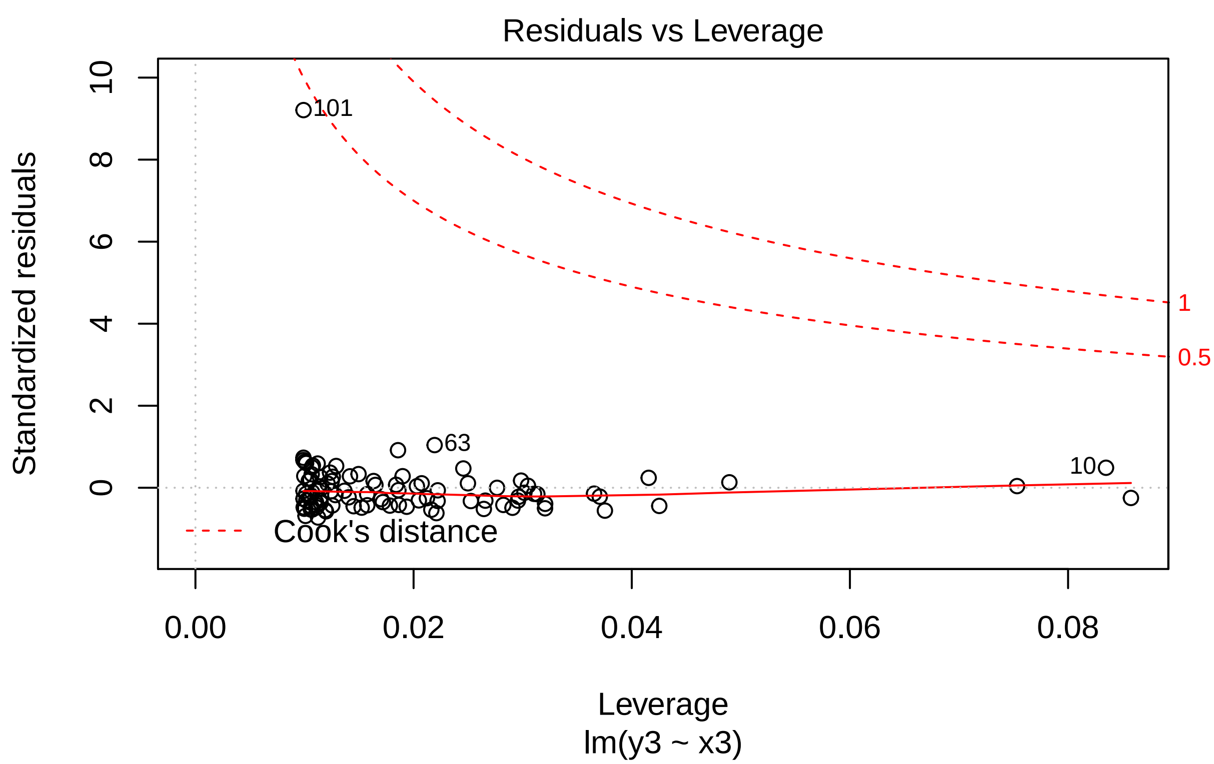
```
##           1           2           3           4           5           6 
## 0.029852944 0.011477266 0.032061101 0.030158895 0.011895307 0.041550417 
##           7           8           9          10          11          12 
## 0.021627150 0.011205219 0.009911309 0.083478411 0.017022897 0.018649829 
##          13          14          15          16          17          18 
## 0.027652341 0.011162327 0.011242446 0.012342586 0.010585543 0.020740859 
##          19          20          21          22          23          24 
## 0.012115043 0.030484227 0.010690150 0.011534505 0.010060427 0.017824461 
##          25          26          27          28          29          30 
## 0.009983039 0.012902715 0.024560418 0.016499537 0.032050529 0.009908635 
##          31          32          33          34          35          36 
## 0.010765024 0.014044728 0.012594052 0.015754471 0.016332433 0.026440870 
##          37          38          39          40          41          42 
## 0.022092967 0.012549363 0.029067891 0.011081841 0.048950308 0.013657952 
##          43          44          45          46          47          48 
## 0.012829109 0.019367632 0.028230846 0.025249637 0.042523311 0.026576612 
##          49          50          51          52          53          54 
## 0.018399231 0.010709852 0.010670957 0.011966607 0.085763627 0.018571826 
##          55          56          57          58          59          60 
## 0.009978505 0.010573193 0.014503393 0.010394339 0.075325027 0.031345004 
##          61          62          63          64          65          66 
## 0.011367211 0.009915531 0.021926993 0.010148867 0.018993082 0.010527484 
##          67          68          69          70          71          72 
## 0.037068341 0.011430777 0.010563025 0.009924790 0.009901130 0.010616359 
##          73          74          75          76          77          78 
## 0.015746432 0.010161000 0.015239715 0.024982513 0.010091845 0.010155829 
##          79          80          81          82          83          84 
## 0.021223816 0.037539104 0.018608985 0.029568602 0.017188138 0.010660040 
##          85          86          87          88          89          90 
## 0.012494535 0.012427359 0.022214634 0.011054083 0.029630472 0.014953120 
##          91          92          93          94          95          96 
```

```
## 0.031039623 0.020302965 0.009903558 0.020480459 0.014170013 0.010023670
##          97          98          99         100         101
## 0.011104073 0.036545454 0.010454267 0.022203443 0.009907521
```

```
summary(mod_inf)
```

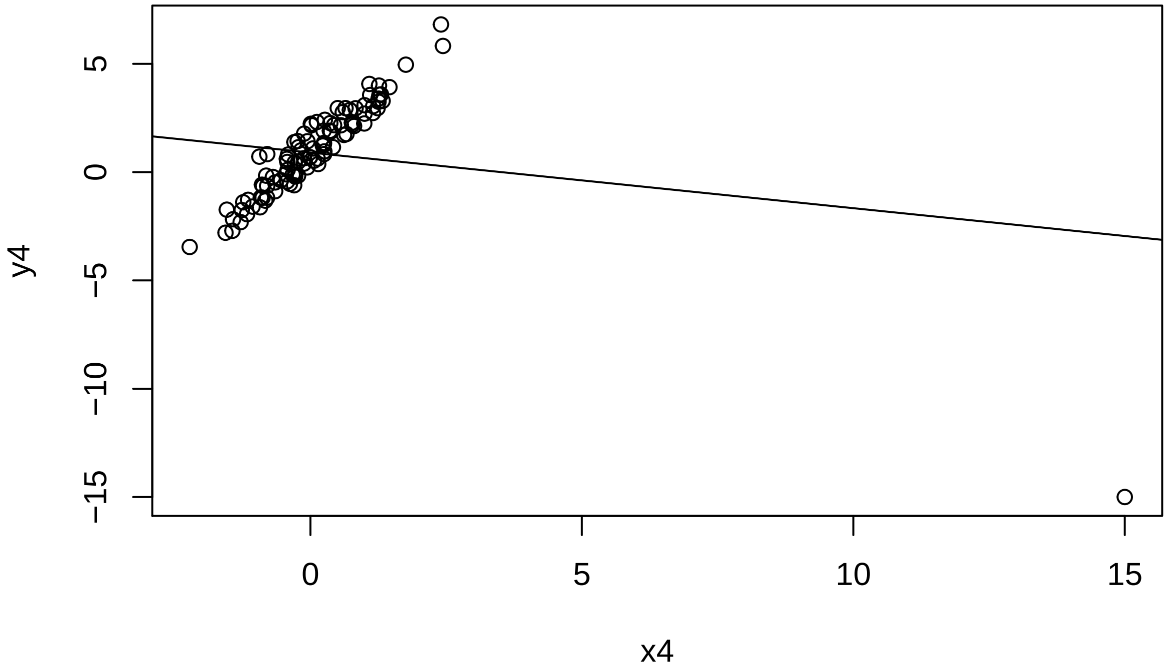
```
##
## Call:
## lm(formula = y3 ~ x3, data = df3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.0942 -0.6299 -0.2346  0.2463 13.8902
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.1098     0.1509   7.354 5.64e-11 ***
## x3            2.0792     0.1726  12.044  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.516 on 99 degrees of freedom
## Multiple R-squared:  0.5944, Adjusted R-squared:  0.5903
## F-statistic: 145.1 on 1 and 99 DF,  p-value: < 2.2e-16
```

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




```
# Now add a new point far from others, and with great influence
x4 <- c(x, 15)
y4 <- c(y, - 15)
plot(x4, y4)
```

```r
df4 <- data.frame(x4, y4)
mod_il <- lm(y4 ~ x4, df4)
abline(mod_il)
```




```r
hatvalues(mod_il)
```

```
##           1           2           3           4           5           6
## 0.013886098 0.010727118 0.014388916 0.013955554 0.010099438 0.019683543
##           7           8           9          10          11          12
## 0.013941251 0.010625714 0.010005366 0.026575229 0.011074732 0.013045213
##          13          14          15          16          17          18
## 0.015711778 0.010609438 0.010639771 0.011034813 0.009923060 0.013676424
##          19          20          21          22          23          24
## 0.010135190 0.016530944 0.010423015 0.010043411 0.009905720 0.011241044
##          25          26          27          28          29          30
## 0.010074817 0.010270794 0.012697786 0.012383694 0.016981387 0.009952569
##          31          32          33          34          35          36
## 0.010453669 0.011603967 0.011121270 0.012150470 0.010933330 0.013116832
##          37          38          39          40          41          42
## 0.012154700 0.011105981 0.013708213 0.010578651 0.018317301 0.010408744
##          43          44          45          46          47          48
## 0.011201188 0.013263025 0.015879668 0.015010996 0.019957973 0.013147230
##          49          50          51          52          53          54
## 0.011361543 0.010431132 0.009931265 0.010903348 0.027127822 0.013021464
##          55          56          57          58          59          60
## 0.010071436 0.009921944 0.010569590 0.010295502 0.029068178 0.016778701
##          61          62          63          64          65          66
## 0.010018819 0.010011694 0.014030531 0.010175848 0.013149532 0.009917982
##          67          68          69          70          71          72
## 0.018414240 0.010028044 0.009921039 0.009938307 0.009978002 0.009925926
##          73          74          75          76          77          78
## 0.012147939 0.010182295 0.010713828 0.012791496 0.009903460 0.010179557
##          79          80          81          82          83          84
```

```
## 0.013820910 0.018547980 0.013032780 0.013821612 0.011108838 0.010410537
##          85          86          87          88          89          90
## 0.011087181 0.011064078 0.012181269 0.010567954 0.013835638 0.010657288
##          91          92          93          94          95          96
## 0.014155884 0.013545031 0.009989334 0.013598336 0.010505481 0.009909765
##          97          98          99         100         101
## 0.010587188 0.015419536 0.009912256 0.012178824 0.744794803
```

```r
summary(mod_il)
```

```
##
## Call:
## lm(formula = y4 ~ x4, data = df4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.0508  -1.4139  -0.0059   1.4839   6.5306
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.9033     0.2465   3.664 0.000401 ***
## x4           -0.2568     0.1425  -1.802 0.074575 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.465 on 99 degrees of freedom
## Multiple R-squared:  0.03176,    Adjusted R-squared:  0.02198
## F-statistic: 3.248 on 1 and 99 DF,  p-value: 0.07457
```

```r
plot(mod_il, which = 5)
```