Laboratorio con R - 4 ANOVA

Metodi e Modelli per l'Inferenza Statistica - Ing. Matematica - a.a. 2023-24

Topics:

- Predittore categorico

- ANOVA (One-way, Two-ways)

0. Librerie

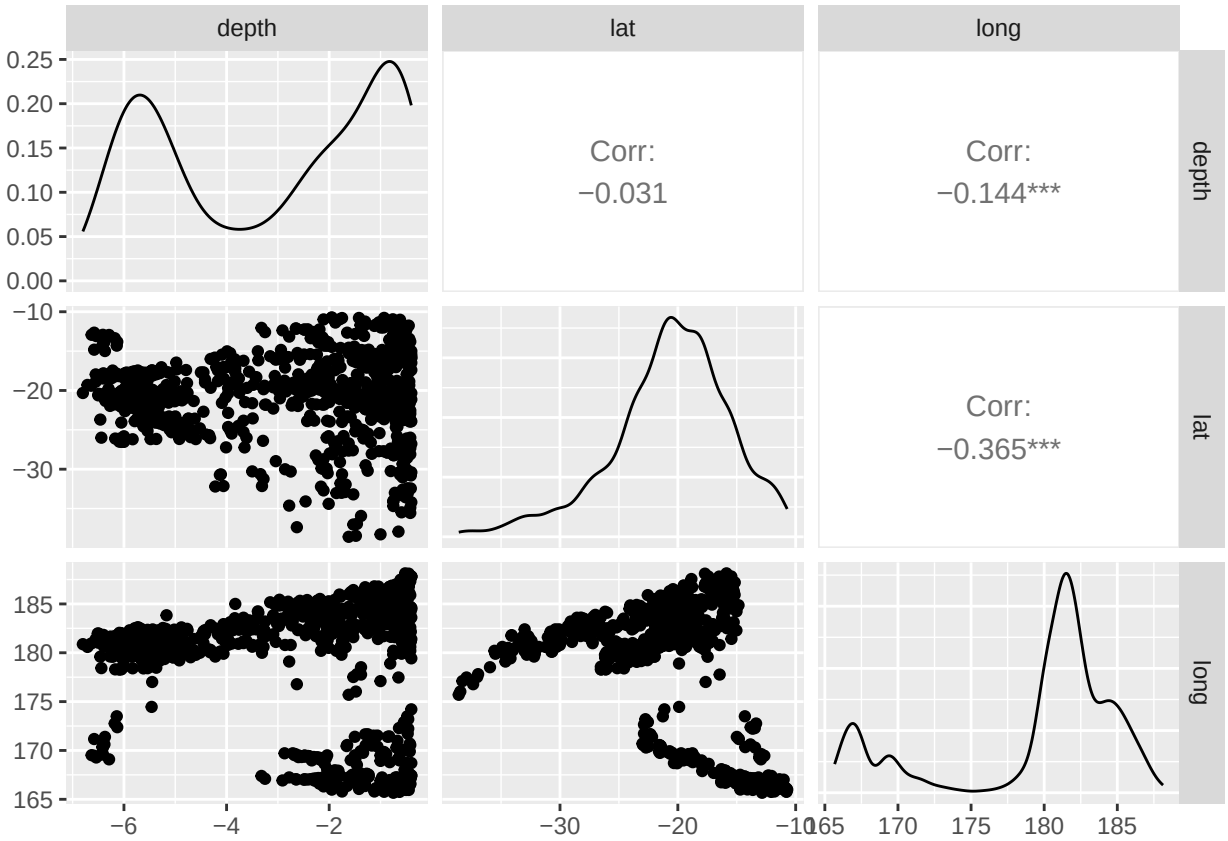
```
library( MASS )
library( car ) #for LEVENE TEST
library( faraway )
library(GGally)
library( Matrix )
library(rgl)
```

1. Modello lineare con predittore categorico

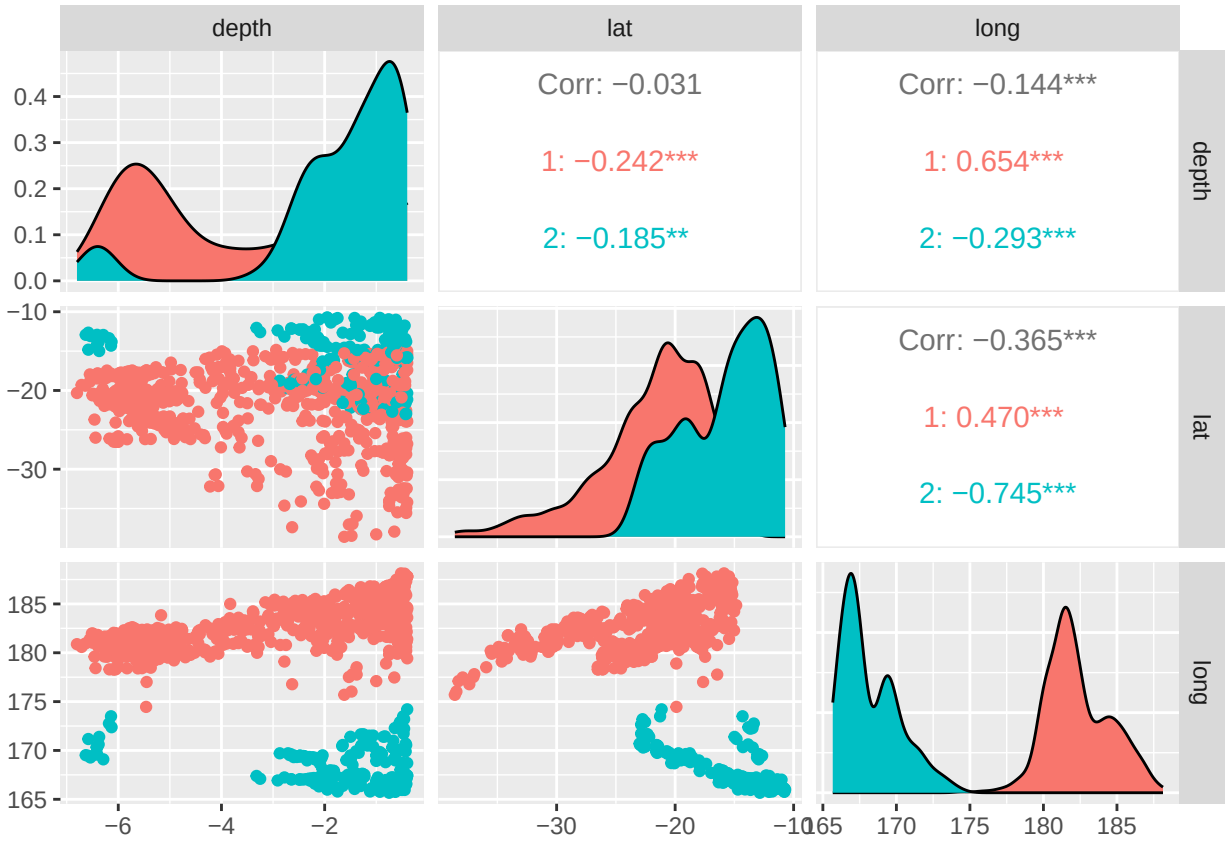
Importare il dataset Terremoti. Impostare un modello lineare per prevedere la profondità del terremoto in funzione di latitudine, longitudine e zona sismica. Il modello è buono? Il modello è valido?

2.1 Importiamo e visualizziamo i dati

```
terr = read.csv("Terremoti.csv")

# cosa contiene in dataset?
str(terr)
## 'data.frame':    1000 obs. of  7 variables:
##  $ X       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ lat     : num  -20.4 -20.6 -26 -18 -20.4 ...
##  $ long    : num  182 181 184 182 182 ...
##  $ depth   : num  -5.62 -6.5 -0.42 -6.26 -6.49 -1.95 -0.82 -1.94 -2.11 -6.22 ...
##  $ mag     : num  4.8 4.2 5.4 4.1 4 4 4.8 4.4 4.7 4.3 ...
##  $ stations: int  41 15 43 19 11 12 43 15 35 19 ...
##  $ zone    : int  1 1 1 1 1 1 2 1 1 1 ...

## visualizziamo i dati numerici
ggpairs(terr[,c('depth','lat','long')])
```




```
## coloriamo per zona
ggpairs(terr[,c('depth','lat','long')], aes(col=as.factor(terr$zone)))
```



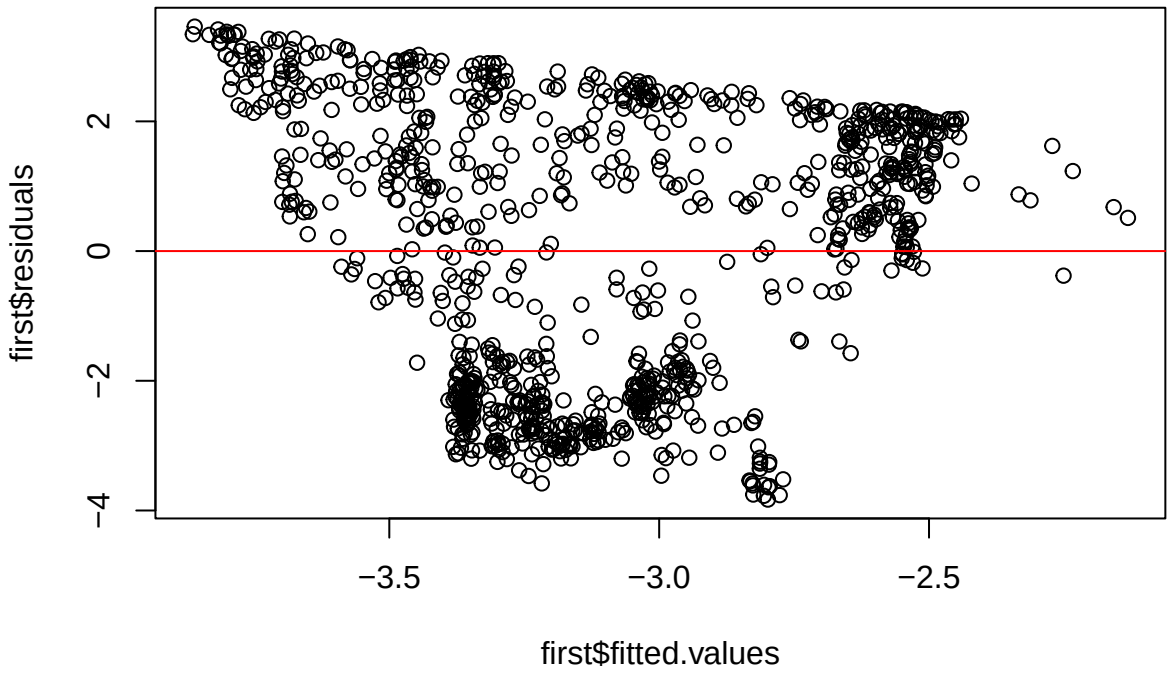
La divisione per zona sembra molto marcata.

2.2 Costruiamo un modello lineare che contenga come covariate solo latitudine e longitudine, ignorando per il momento la variabile categorica zona.

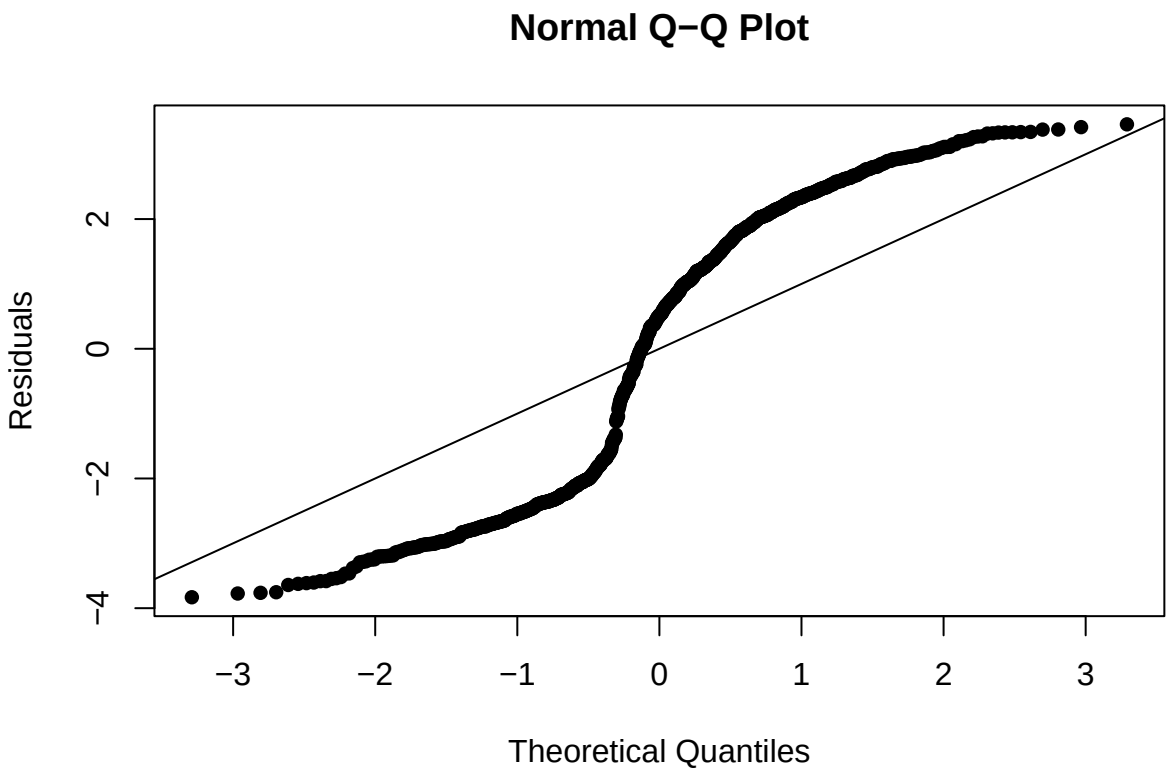
```
first = lm(depth ~ lat + long, data = terr)
summary(first)
##
## Call:
## lm(formula = depth ~ lat + long, data = terr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8314 -2.2432  0.5165  1.9615  3.4606
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.47977    2.04760   3.653 0.000273 ***
## lat         -0.04136    0.01436  -2.880 0.004068 **
## long        -0.06379    0.01190  -5.360 1.04e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.126 on 997 degrees of freedom
## Multiple R-squared:  0.02894,    Adjusted R-squared:  0.02699
## F-statistic: 14.86 on 2 and 997 DF,  p-value: 4.387e-07
```

2.3 Diagnostica dei residui

```
## diagnostica dei residui

plot(first$fitted.values, first$residuals)
abline(h=0, col='red')
```




```
qqnorm( first$residuals, ylab = "Residuals", pch = 16 )
abline( 0, 1 )
```




```
shapiro.test(first$residuals)
##
##  Shapiro-Wilk normality test
##
## data:  first$residuals
## W = 0.90501, p-value < 2.2e-16
```

Modello pessimo: R2 bassissimo, non normalità e omoschedasticità dei residui.

Idea: zone diverse corrispondono a modelli lineari diversi, con parametri diversi ma identica σ^2. Come modellare le variabili categoriche nel modello? Matematicamente possiamo scrivere, se le variabili si dividono in K gruppi:

$$y_{ik} = \beta_{0k} + \beta_{1k} * x1_{ik} + \beta_{2k} * x2_{ik} + \epsilon_{ik},$$

con $Var(\epsilon_{ik}) = \sigma^2$, per ogni i=1,...,N, k = 1,...K. Come modellare tutto in un unico modello?

Prendiamo il primo gruppo come riferimento, cosi' che per il primo gruppo valga il modello

$$y_{i1} = \beta_{01} + \beta_{11} * x1_{i1} + \beta_{21} * x2_{i1} + \epsilon_{i1}.$$

Poi per un generico gruppo k diverso dal primo, definiamo la variabile dummy $d_{ik} = 1$ se la i-esima osservazione e' nel gruppo k, d_{ik} = 0 altrimenti. Scriviamo quindi il modello generale per un'osservazione i qualsiasi:

$$y_i = \beta_{01}+\beta_{11}*x1_i+\beta_{21}*x2_i+\beta_{02}*d_{i2}+\beta_{12}*x1_i*d_{i2}+\beta_{21}*x2_i*d_{i2}+....+\beta_{0K}*d_{iK}+\beta_{1K}*x1_i*d_{iK}+\beta_{2K}*x2_i*d_{iK}$$

Cosi', in un unico modello, otteniamo in realta' K modelli lineari diversi: Per il primo gruppo, il modello e' proprio

$$y_{i1} = \beta_{01} + \beta_{11} * x1_{i1} + \beta_{21} * x2_{i1} + \epsilon_{i1}$$

Per ogni altro gruppo avremo

$$y_{ik} = (\beta_{01} + \beta_{0k}) + (\beta_{11} + \beta_{1k}) * x1_{ik} + (\beta_{21} + \beta_{2k}) * x2_{ik} + \epsilon_{ik}$$

Nota: se ci sono K gruppi, abbiamo bisogno di K-1 variabili dummy!

In questo caso abbiamo due gruppi, quindi abbiamo bisogno di un nuovo vettore che abbia 0 in corrispondenza del primo gruppo e 1 in corrispondenza del secondo (o viceversa, è indifferente).

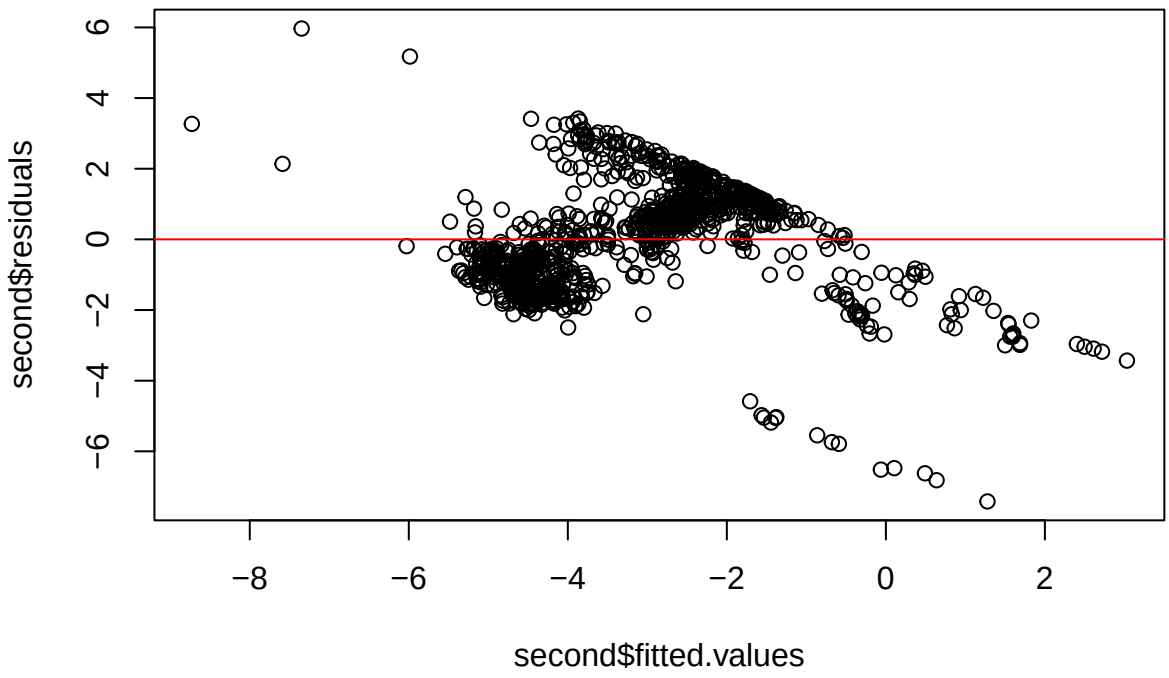
2.4 Costruire un modello lineare in cui aggiungiamo la covariata categorica zone alle due numeriche usate in precedenza.

```
dummy = ifelse(terr$zone == 1, 0,1)

table(terr$zone)
##
##   1   2
## 796 204
table(dummy)
## dummy
##   0   1
## 796 204

terr$dummy = dummy
table(terr$dummy)
##
##   0   1
## 796 204
terr$dummy = as.factor(terr$dummy)


second = lm(depth ~ lat + long + dummy , data = terr)
summary(second)
##
## Call:
## lm(formula = depth ~ lat + long + dummy, data = terr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.4177 -1.0720  0.1003  1.0425  5.9677
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -120.26894    4.42539  -27.18   <2e-16 ***
## lat           -0.19268    0.01144  -16.84   <2e-16 ***
## long           0.61738    0.02384   25.90   <2e-16 ***
## dummy1        11.67393    0.38142   30.61   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.527 on 996 degrees of freedom
## Multiple R-squared:  0.4996, Adjusted R-squared:  0.4981
## F-statistic: 331.5 on 3 and 996 DF,  p-value: < 2.2e-16
```

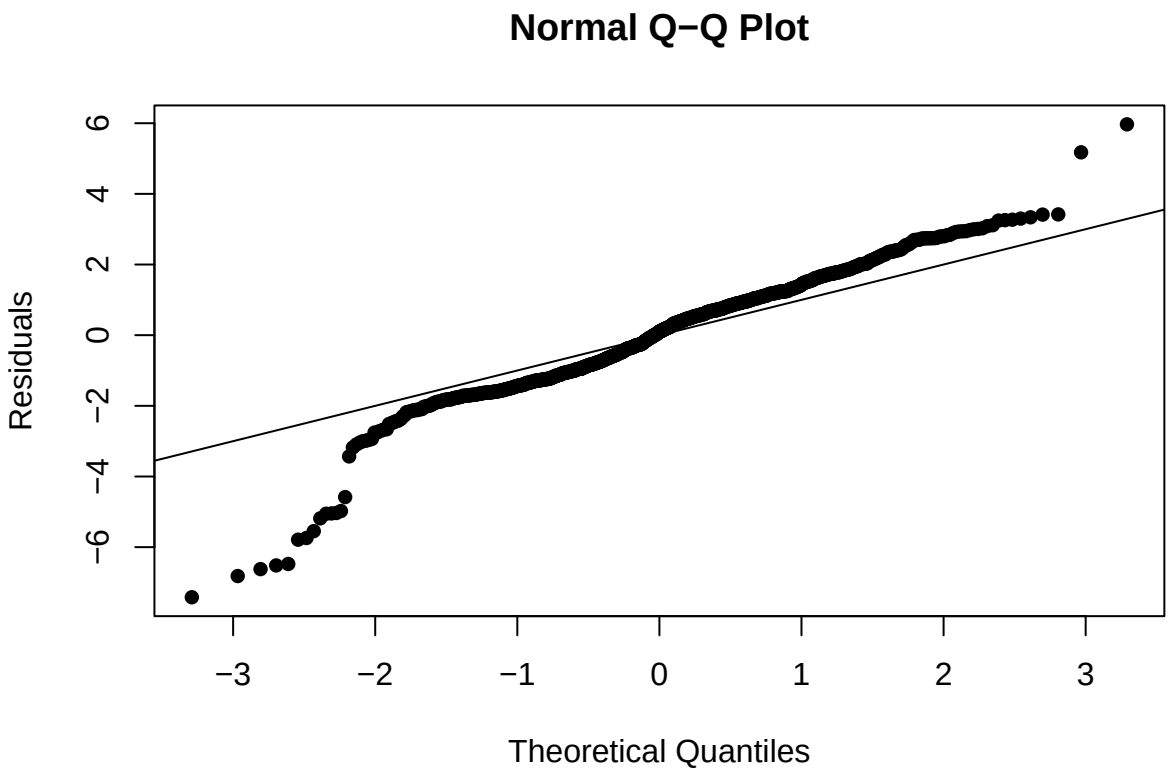
Miglioramento notevole!

Controlliamo i residui:

```r
plot(second$fitted.values, second$residuals)
abline(h=0, col='red')
```




```r
qqnorm( second$residuals, ylab = "Residuals", pch = 16 )
abline( 0, 1 )
```




```
shapiro.test(second$residuals)
##
##  Shapiro-Wilk normality test
##
## data:  second$residuals
## W = 0.96566, p-value = 1.31e-14
```

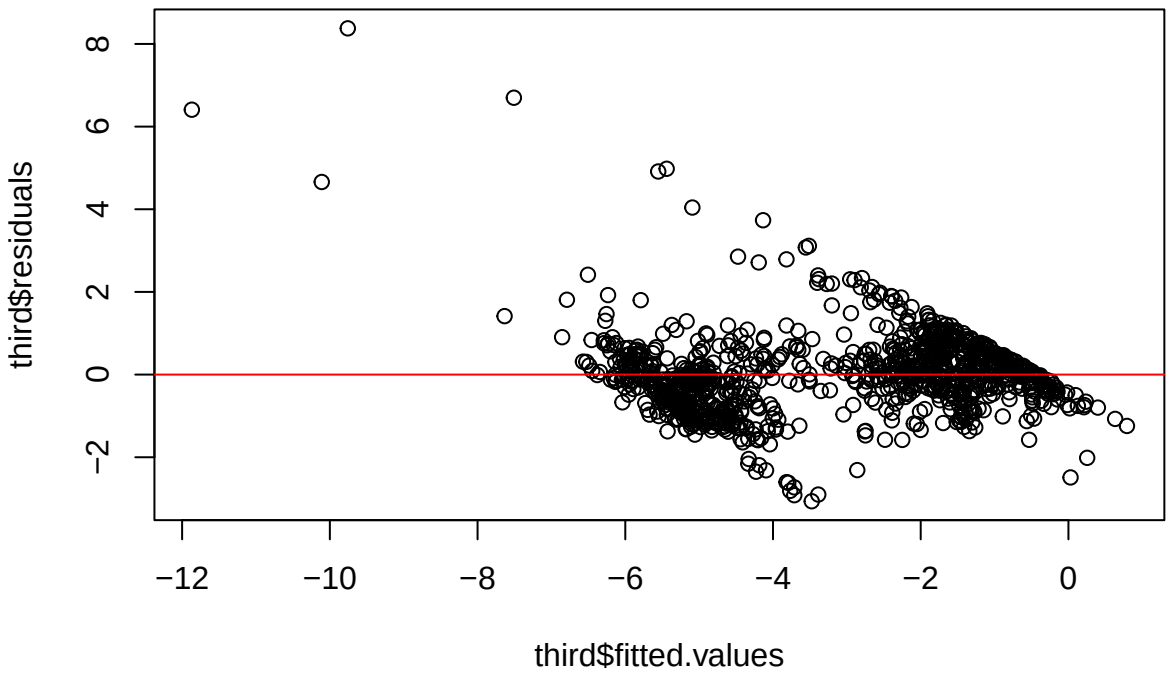
In realtà R può gestire tutto autonomamente, senza che noi dobbiamo definire variabili dummy:

```
terr$zone = factor(terr$zone, ordered = F) # Importante mettere factor e ordered = F
second2 = lm(depth ~ lat + long + zone, data = terr)
summary(second2)
##
## Call:
## lm(formula = depth ~ lat + long + zone, data = terr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.4177 -1.0720  0.1003  1.0425  5.9677
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -120.26894    4.42539  -27.18   <2e-16 ***
## lat           -0.19268    0.01144  -16.84   <2e-16 ***
## long           0.61738    0.02384   25.90   <2e-16 ***
## zone2         11.67393    0.38142   30.61   <2e-16 ***
```

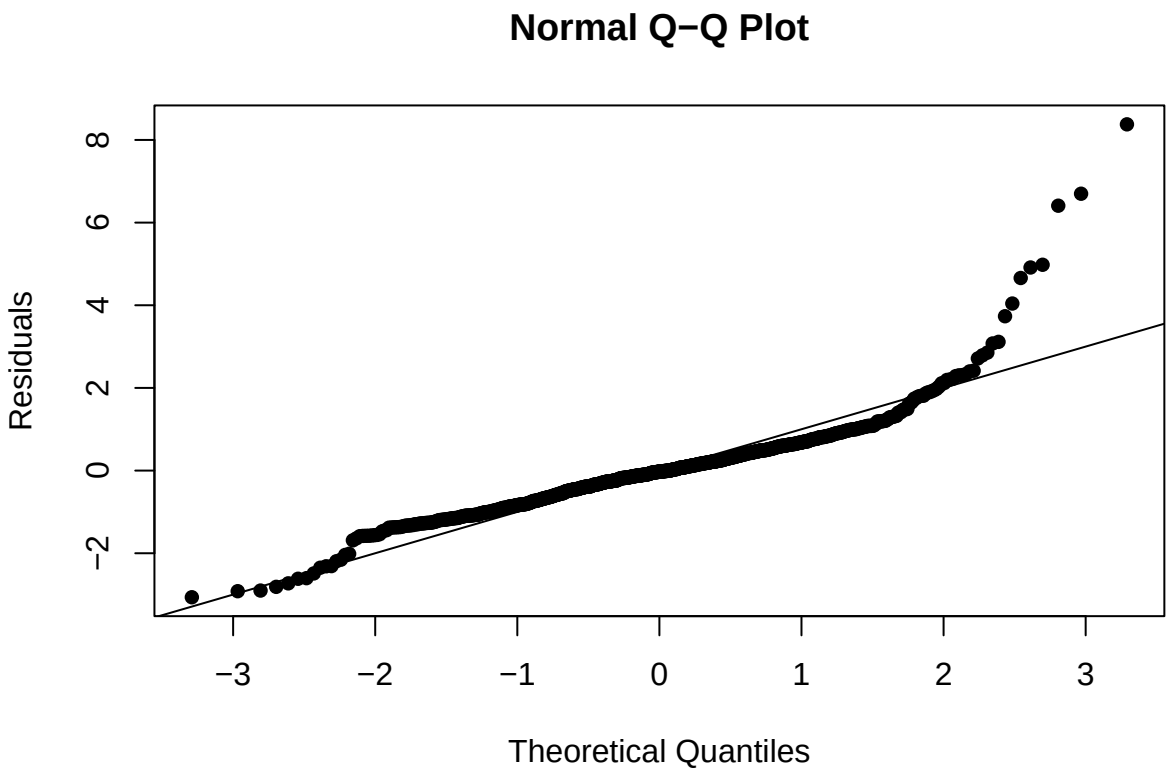
```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.527 on 996 degrees of freedom
## Multiple R-squared:  0.4996, Adjusted R-squared:  0.4981
## F-statistic: 331.5 on 3 and 996 DF,  p-value: < 2.2e-16
## Il modello e' esattamente identico

# i due piani sono paralleli ovviamente, per costruzione!
```

2.5 Dal grafico sembrerebbe ragionevole inserire anche l'interazione tra zone e le altre covariate numeriche. Aggiungere e valutare i due termini di interazione.

```
third = lm(depth ~ lat + long + dummy + dummy * lat + dummy * long, data = terr)
summary(third) ## Miglioramento notevole!
##
## Call:
## lm(formula = depth ~ lat + long + dummy + dummy * lat + dummy *
##     long, data = terr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0644 -0.5425 -0.0247  0.4532  8.3771
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.896e+02  3.351e+00  -56.59   <2e-16 ***
## lat         -3.304e-01  8.412e-03  -39.28   <2e-16 ***
## long         9.813e-01  1.788e-02   54.88   <2e-16 ***
## dummy1       3.061e+02  9.119e+00   33.57   <2e-16 ***
## lat:dummy1  -3.924e-02  2.929e-02   -1.34    0.181
## long:dummy1 -1.718e+00  5.532e-02  -31.07   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.961 on 994 degrees of freedom
## Multiple R-squared:  0.8022, Adjusted R-squared:  0.8012
## F-statistic: 806.2 on 5 and 994 DF,  p-value: < 2.2e-16

plot(third$fitted.values, third$residuals)
abline(h=0, col='red')
```




```
qqnorm( third$residuals, ylab = "Residuals", pch = 16 )
abline( 0, 1 )
```



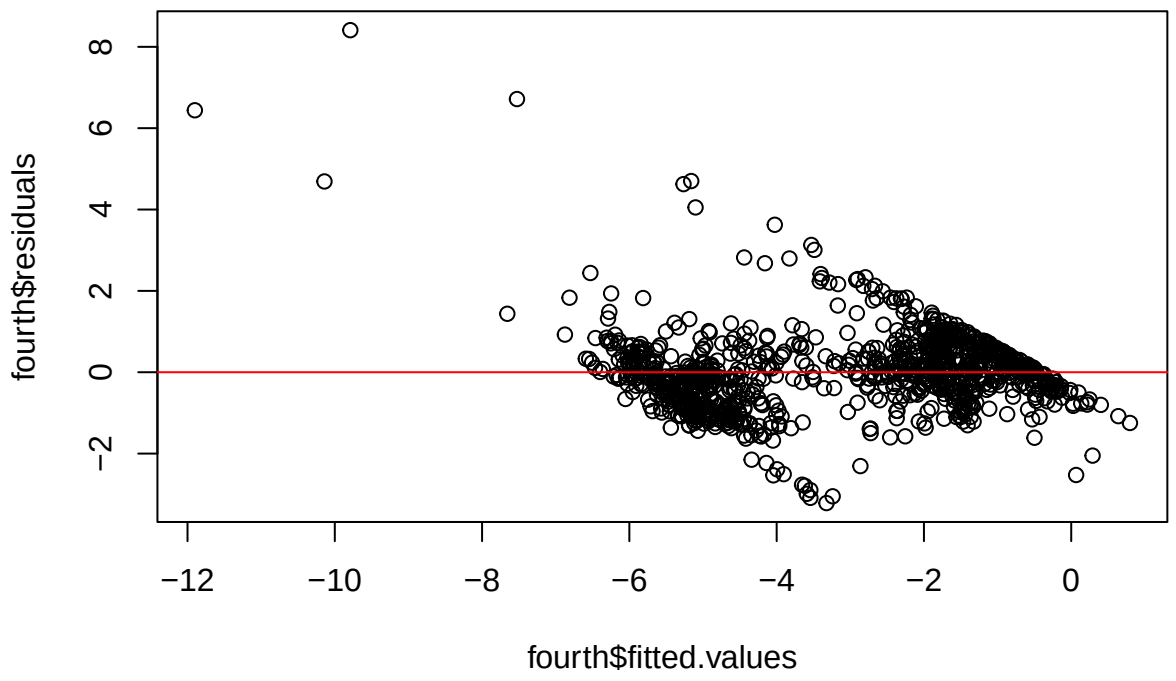

```
shapiro.test(third$residuals)
##
##  Shapiro-Wilk normality test
##
## data:  third$residuals
## W = 0.88338, p-value < 2.2e-16
```

Dal summary, si direbbe che la zona abbia scarsa interazione con la latitudine, proviamo a ridurre il modello:

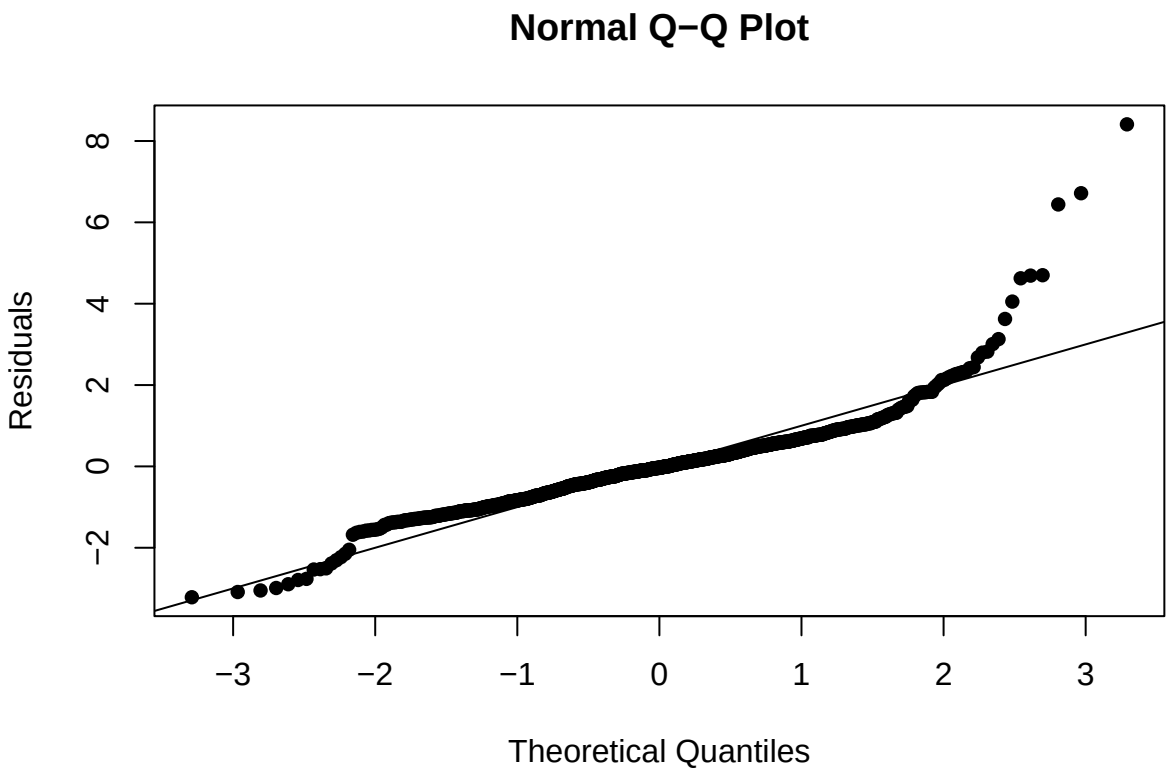
```
fourth = lm(depth ~ lat + long + zone + zone*long, data = terr)
summary(fourth)
##
## Call:
## lm(formula = depth ~ lat + long + zone + zone * long, data = terr)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2167 -0.5394 -0.0330  0.4690  8.4094
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.903e+02  3.316e+00  -57.39   <2e-16 ***
## lat         -3.336e-01  8.061e-03  -41.39   <2e-16 ***
## long         9.845e-01  1.772e-02   55.55   <2e-16 ***
## zone2        2.989e+02  7.378e+00   40.52   <2e-16 ***
## long:zone2  -1.672e+00  4.291e-02  -38.95   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9614 on 995 degrees of freedom
## Multiple R-squared:  0.8018, Adjusted R-squared:  0.801
## F-statistic:  1006 on 4 and 995 DF,  p-value: < 2.2e-16
```

```
plot(fourth$fitted.values, fourth$residuals)
abline(h=0, col='red')
```




```
qqnorm( fourth$residuals, ylab = "Residuals", pch = 16 )
abline( 0, 1 )
```




```
shapiro.test(fourth$residuals) # ancora i residui non sono normali, infatti anche ad occhio le code si
##
##  Shapiro-Wilk normality test
##
## data:  fourth$residuals
## W = 0.88477, p-value < 2.2e-16

rm(list=ls())
```

..idee? poly(long,3)? boxcox? punti influenti?

2. Visualizzazione della decomposizione della varianza

Lavoreremo con il dataset presente nel documento ciclisti.txt. Sono state considerate dieci strade con pista ciclabile ed è stata misurata la distanza tra la linea di mezzeria (linea longitudinale che scorre in mezzo alla strada, dividendo la carreggiata in diverse corsie) e un ciclista sulla pista ciclabile. In queste stesse dieci strade è stata determinata attraverso fotografie la distanza tra lo stesso ciclista e una macchina passante per la strada considerata.

- Center → distanza tra la linea di mezzeria e un ciclista sulla pista ciclabile [misuarata in piedi]
- Car → distanza tra lo stesso ciclista e una macchina passante per la strada considerata [misuarata in piedi]

Quando operiamo una regressione lineare semplice, possiamo usare le sequenti tre quantità per scomporre la varianza (l'informazione) del modello.

Explained sum of squares o **Sum of Squares Regression**:

$$SS_{reg} = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2$$

è una misura della variabilità della variabile dipendente del modello rispetto a come è spiegata dalle variabili indipendenti.

Residual sum of squares o **Sum of Squares Error**:

$$SS_{err} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

é una misura della variabilità della variabile dipendente che NON è spiegata dalla nostra regressione.

Total sum of squares:

$$SS_{tot} = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

rappresenta tutta la variabilità della variabile dipendente.

dove con y_i si sono indicati i valori della variabile risposta in corrispondenza di ciascun valore x_i, con $\hat{y}_i$ i valori stimati sulla retta di regressione e con $\bar{y}$ la media delle y_i. È facile vedere che:

$$SS_{tot} = SS_{reg} + SS_{err}$$

Soluzione

Importiamo il dataset CICLISTI:

```
ciclisti = read.table( "ciclisti.txt", header = TRUE )

head(ciclisti)
##   Center Car
## 1   12.8 5.5
## 2   12.9 6.2
## 3   12.9 6.3
## 4   13.6 7.0
## 5   14.5 7.8
## 6   14.6 8.3

names( ciclisti )
## [1] "Center" "Car"

dim( ciclisti )
## [1] 10  2

n = dim( ciclisti )[1]
```

Fittiamo ora un modello lineare semplice e calcoliamo le quantità di interesse.

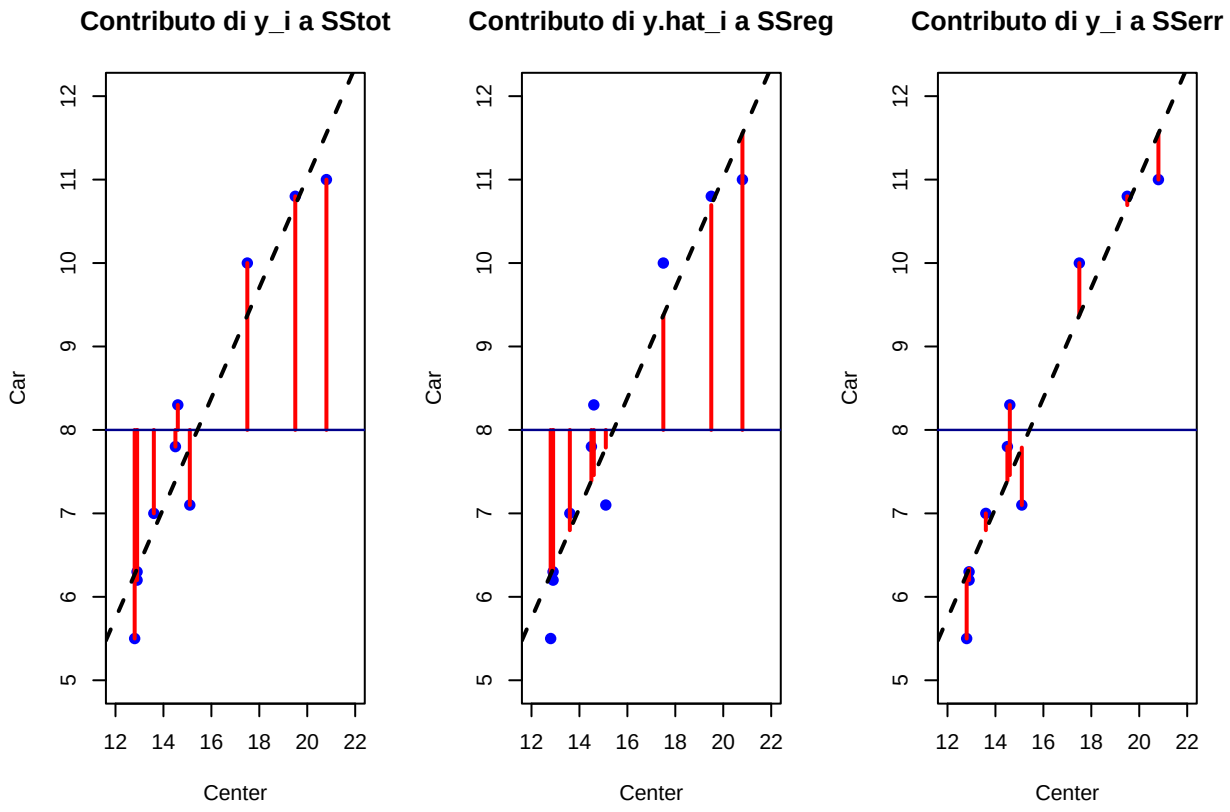
```
reg.ciclisti = lm( Car ~ Center, data = ciclisti )
summary( reg.ciclisti )
##
## Call:
## lm(formula = Car ~ Center, data = ciclisti)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -0.76990 -0.44846  0.03493  0.35609  0.84148
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.18247    1.05669  -2.065   0.0727 .
## Center       0.66034    0.06748   9.786 9.97e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5821 on 8 degrees of freedom
## Multiple R-squared:  0.9229, Adjusted R-squared:  0.9133
## F-statistic: 95.76 on 1 and 8 DF,  p-value: 9.975e-06

# y_i ( dati osservati )
y.i = ciclisti$Car
y.i
##  [1]  5.5  6.2  6.3  7.0  7.8  8.3  7.1 10.0 10.8 11.0

# y medio
y.mean = mean( ciclisti$Car )
y.mean
## [1] 8

# y.hat ( dati fittati dal modello )
y.hat = reg.ciclisti$fitted.value
y.hat
##         1         2         3         4         5         6         7         8
##  6.269904  6.335939  6.335939  6.798178  7.392485  7.458520  7.788691  9.373511
##         9        10
## 10.694195 11.552639
```

Facciamo il grafico delle quantità di interesse.

```
par( mfrow = c( 1, 3 ) )

# SStot = Sum( y_i - y_medio )^2
plot( NULL, xlim = c( 12, 22 ), ylim = c( 5, 12 ),
      xlab = "Center", ylab = "Car", main = "Contributo di y_i a SStot" )
points( ciclisti$Center, ciclisti$Car,
        pch = 16, col = 'blue', cex = 1.2 )

for ( i in 1:n )
  segments( ciclisti$Center [ i ], ciclisti$Car[ i ], ciclisti$Center [ i ], y.mean,
            lwd = 2, col = 'red', lty = 1 )

abline( h = y.mean , col = 'darkblue', lwd = 1.2 )

abline( reg.ciclisti, lwd = 2, col = 'black', lty = 2 )

# SSreg = Sum( y.hat_i - y_medio )^2
plot( NULL, xlim = c( 12, 22 ), ylim = c( 5, 12 ),
      xlab = "Center", ylab = "Car", main = "Contributo di y.hat_i a SSreg" )
points( ciclisti$Center, ciclisti$Car,
```

```
        pch = 16, col = 'blue', cex = 1.2 )

for ( i in 1:n )
  segments( ciclisti$Center [ i ], y.hat[ i ], ciclisti$Center [ i ], y.mean,
            lwd = 2, col = 'red', lty = 1 )

abline( h = y.mean , col = 'darkblue', lwd = 1.2 )

abline( reg.ciclisti, lwd = 2, col = 'black', lty = 2 )


# SSerr = Sum( y_i - y.hat_i )^2
plot( NULL, xlim = c( 12, 22 ), ylim = c( 5, 12 ),
      xlab = "Center", ylab = "Car", main = "Contributo di y_i a SSerr" )
points( ciclisti$Center, ciclisti$Car, pch = 16, col = 'blue', cex = 1.2 )

for ( i in 1:n )
  segments( ciclisti$Center [ i ], ciclisti$Car[ i ], ciclisti$Center [ i ], y.hat [ i ],
            lwd = 2, col = 'red', lty = 1 )

abline( h = y.mean , col = 'darkblue', lwd = 1.2 )

abline( reg.ciclisti, lwd = 2, col = 'black', lty = 2 )
```



```
rm(list=ls())
```

3. One-way ANOVA (I)

Importiamo i dati `chickwts`. Vogliamo investigare se il peso dei polli è influenzato dal tipo di alimentazione (y = weights e τ = feed).

Soluzione

Importiamo i dati.

```
data( chickwts )
head( chickwts )
##   weight      feed
## 1    179 horsebean
## 2    160 horsebean
## 3    136 horsebean
## 4    227 horsebean
## 5    217 horsebean
## 6    168 horsebean
tail( chickwts )
##    weight   feed
## 66    352 casein
## 67    359 casein
## 68    216 casein
## 69    222 casein
## 70    283 casein
## 71    332 casein

attach( chickwts )
```

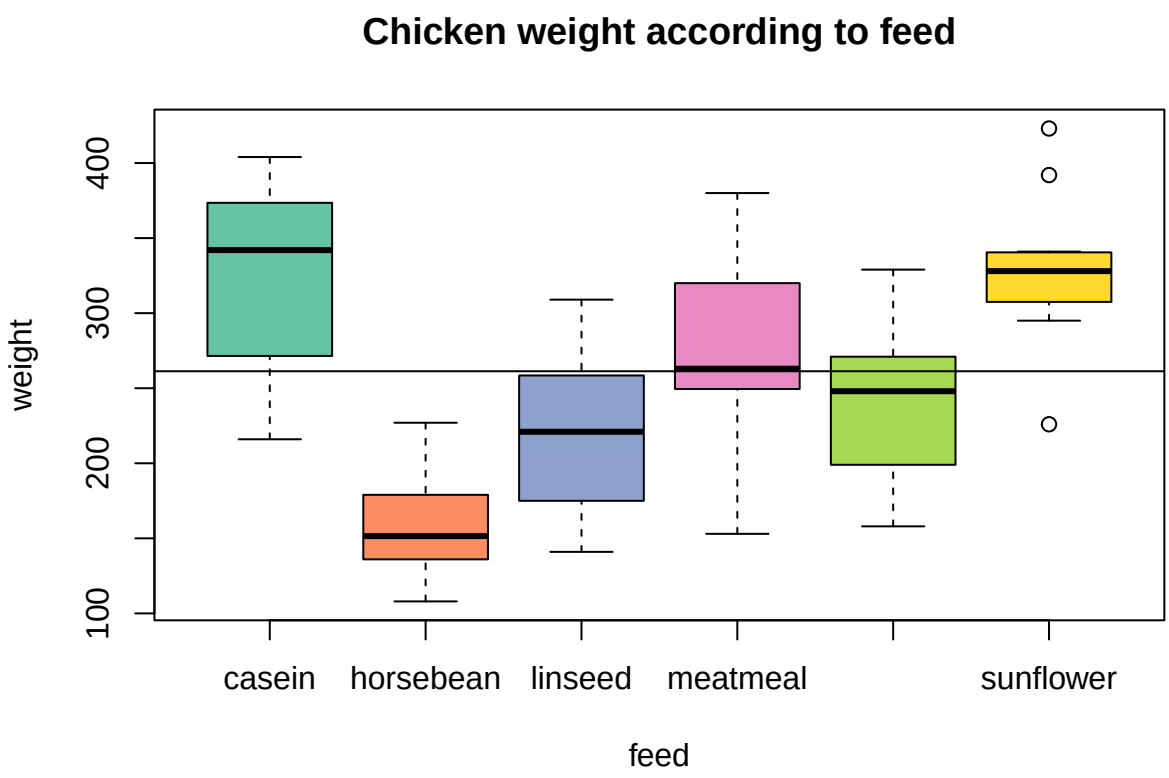
Iniziamo a farci un'idea descrittiva dei dati per avere indicazioni qualitative sulla presenza di differenziazione nella risposta a causa dell'appartenenza ad una o all'altra categoria.

L'analisi della varianza, nota con l'acronimo di ANOVA, è una tecnica statistica che ha come obiettivo il confronto tra le medie di un fenomeno aleatorio fra differenti gruppi di unità statistiche. Tale analisi viene affrontata tramite decomposizione della varianza.

Consiglio grafico: spesso è utile rappresentare i gruppi di dati con colori diversi, potete trovare alcune palette di colori nel pacchetto RColorBrewer.

```
library(RColorBrewer)

# display.brewer.all() # mostra le palette disponibili in RColorBrewer

my_colors = brewer.pal( length( levels( chickwts$feed ) ), 'Set2')

summary( chickwts )
##      weight             feed
##  Min.   :108.0   casein   :12
##  1st Qu.:204.5   horsebean:10
##  Median :258.0   linseed  :12
##  Mean   :261.3   meatmeal :11
##  3rd Qu.:323.5   soybean  :14
##  Max.   :423.0   sunflower:12
```

```r
boxplot( weight ~ feed, xlab = 'feed', ylab = 'weight',
         main = 'Chicken weight according to feed',
         col = my_colors )
abline( h = mean( weight ) )
```




```r
tapply( chickwts$weight, chickwts$feed, length )
##    casein horsebean   linseed  meatmeal   soybean sunflower
##        12        10        12        11        14        12
```

Sembra che un qualche effetto ci sia, le medie appaiono diverse e sembra vi sia una dominanza stocastica delle distribuzioni dei pesi.

Il modello che vogliamo fittare è il seguente:

$$y_{ij} = \mu_j + \varepsilon_{ij} = \mu + \tau_j + \varepsilon_{ij}$$

in cui $i \in \{1, .., n_j\}$ è l'indice dell'unità statistica all'interno del gruppo j, mentre $j \in \{1, .., g\}$ è l'indice di gruppo. τ_j rappresenta la deviazione media rispetto a μ nel gruppo j.

Siamo interessati ad eseguire il seguente test:

$$H_0 : \mu_i = \mu_j \quad \forall i, j \in \{1, .., 6\} \qquad vs \qquad H_1 : \exists\,(i, j)\,|\mu_i \neq \mu_j$$

Parafrasando, H_0 prevede che tutti i polli appartengono ad una sola popolazione, mentre H_1 prevede che i polli appartengono a 2, 3, 4, 5 o 6 popolazioni.

Facciamo un'ANOVA manuale.

Verifichiamo che siano soddisfatte le ipotesi dell'ANOVA:

- **Normalità** intragruppo;

```
n  = length( feed )
ng  = table( feed )
treat  = levels( feed )
g  = length( treat )

# Normalità dei dati nei gruppi
Ps = c( shapiro.test( weight [ feed == treat [ 1 ] ] )$p,
        shapiro.test( weight [ feed == treat [ 2 ] ] )$p,
        shapiro.test( weight [ feed == treat [ 3 ] ] )$p,
        shapiro.test( weight [ feed == treat [ 4 ] ] )$p,
        shapiro.test( weight [ feed == treat [ 5 ] ] )$p,
        shapiro.test( weight [ feed == treat [ 6 ] ] )$p )
Ps
## [1] 0.2591841 0.5264499 0.9034734 0.9611795 0.5063768 0.3602904

# In maniera più compatta ed elegante:
Ps = tapply( weight, feed, function( x ) ( shapiro.test( x )$p ) )

Ps # tutti p-value alti = > non rifiuto mai hp di Normalità
##    casein horsebean   linseed  meatmeal   soybean sunflower
## 0.2591841 0.5264499 0.9034734 0.9611795 0.5063768 0.3602904
```

- **Omoschedasticità** fra i gruppi.

```
# Varianze dei gruppi omogenee
Var = c( var( weight [ feed == treat [ 1 ] ] ),
         var( weight [ feed == treat [ 2 ] ] ),
         var( weight [ feed == treat [ 3 ] ] ),
         var( weight [ feed == treat [ 4 ] ] ),
         var( weight [ feed == treat [ 5 ] ] ),
         var( weight [ feed == treat [ 6 ] ] ) )
Var
## [1] 4151.720 1491.956 2728.568 4212.091 2929.956 2384.992

# In maniera più compatta ed elegante:
Var = tapply( weight, feed, var )
Var
##    casein horsebean   linseed  meatmeal   soybean sunflower
##  4151.720  1491.956  2728.568  4212.091  2929.956  2384.992
```

Per verificare l'omogeneità tra le varianze abbiamo diverse possibilità.

Bartlett's test

Il Bartlett test è il seguente:

$$H_0 : \sigma_1 = \sigma_2 = ... = \sigma_g \qquad vs \qquad H_1 : H_0^C$$

La statistica test 'K' del test di Bartlett è approssimativamente distribuita come una χ^2 con (g-1) gradi di libertà e si definisce come:

$$K = \frac{(N-g)\ln(S_p^2) - \sum_{i=1}^{g}(n_i - 1)\ln(S_i^2)}{1 + \frac{1}{3(g-1)}\left(\sum_{i=1}^{g}(\frac{1}{n_i - 1}) - \frac{1}{N-g}\right)}$$

dove $N = \sum_{i=1}^{g} n_i$ è la numerosità totale del campione, n_i la numerosità del gruppo i, $S_p^2 = \frac{1}{N-g}\sum_i (n_i - 1)S_i^2$ la stima della varianza pooled.

La statistica test 'K' confronta una varianza globale (pooled) con uno stimatore costruito sulla base delle varianze dei singoli gruppi.

Valori piccoli di 'K' mi portano ad accettare H_0, mentre valori grandi di 'K' mi portano a rifiutare H_0.

Il test di Bartlett assume che le osservazioni appartenenti ai vari gruppi siano iid da una Normale. Il test è costruito su questa ipotesi, quindi scostamenti anche di un solo gruppo dalla normalità hanno ripercussioni significative sulla validità e l'esito del test. Ci sono altri test, più robusti, che vagliano le stesse ipotesi (e.g. Levene, Brown-Forsythe).

```
# Test di uniformità delle varianze
bartlett.test( weight, feed )
##
##  Bartlett test of homogeneity of variances
##
## data:  weight and feed
## Bartlett's K-squared = 3.2597, df = 5, p-value = 0.66
```

Il test di Bartlett in questo caso accetta H_0.

Levene's test Anche il test di Levene serve per valutare l'omogeneità delle varianze di una variabile calcolato per due o più gruppi. Questo test è un'alternativa a quello di Bartlett, meno sensibile alla non normalità dei dati.

```
# Alternative: Levene-Test
leveneTest( weight, feed )
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  5  0.7493 0.5896
##       65
```

Anche il test di Levene è concorde nell'accettare l'ipotesi nulla.

Ora che abbiamo verificato che le ipotesi sono soddisfatte possiamo procedere con una **One-Way ANOVA**.

Prima, però, osserviamo come possiamo decomporre la varianza tenendo conto di questi gruppi.

- **Varianza inter-gruppi (Variance Between group)**

$$SS_B = \sum_{j=1}^{g} n_j(\bar{y}_j - \bar{y})^2$$

dove n_j è la numerosità del j-esimo gruppo, $\bar{y}_j$ è la media del j-esimo gruppo, $\bar{y}$ la media dell'intero campione.

- **Varianza intra-gruppi (Variance Within group)**

$$SS_W = \sum_{j=1}^{g} \sum_{i=1}^{n_j} (y_{ij} - \bar{y}_j)^2$$

- **Varianza totale**

$$SS_{TOT} = SS_B + SS_W$$

Queste statistiche a meno di coefficienti di normalizzazione, sono distribuite come χ^2. Allora possiamo definire una statistica F_0 come:

$$F_0 = \frac{SS_B/(g-1)}{SS_W/(N-g)}$$

Sotto l'ipotesi $H_0 : \tau_1 = \tau_2 = \ldots = \tau_g$, abbiamo $F_0 \sim F_{g-1,N-g}$, da testare contro $H_1 : \tau_i \neq \tau_j$ *per qualche* (i, j).

```
Media  = mean( weight )
Media.1 = mean( weight [ feed == treat [ 1 ] ] )
Media.2 = mean( weight [ feed == treat [ 2 ] ] )
Media.3 = mean( weight [ feed == treat [ 3 ] ] )
Media.4 = mean( weight [ feed == treat [ 4 ] ] )
Media.5 = mean( weight [ feed == treat [ 5 ] ] )
Media.6 = mean( weight [ feed == treat [ 6 ] ] )
Mediag  = c( Media.1, Media.2, Media.3, Media.4, Media.5, Media.6 )

# oppure (FORTEMENTE CONSIGLIATO):
Media  = mean( weight )
Mediag  = tapply( weight, feed, mean )

SStot  = var( weight ) * ( n-1 )
SSB = sum( ng * ( Mediag-Media )^2 )
SSW  = SStot - SSB

alpha = 0.05
Fstatistic = ( SSB / ( g-1 ) ) / ( SSW / ( n-g ) )

# valori "piccoli" non ci portano a rifiutare
cfr.fisher = qf( 1-alpha, g-1, n-g )
Fstatistic > cfr.fisher
## [1] TRUE
Fstatistic
## [1] 15.3648
cfr.fisher
## [1] 2.356028
```

F_0 siamo proprio ben oltre la soglia trovata con la distribuzione F, quindi abbiamo un'evidenza forte per rifiutare.

Calcoliamo anche il p-value.

```
P = 1-pf( Fstatistic, g-1, n-g )
P
## [1] 5.93642e-10
```

In R si può anche eseguire l'ANOVA in modo automatico.

Costruiamo un modello anova e guardiamo il summary

```
# help( aov )

fit = aov( weight ~ feed )
summary( fit )
##             Df Sum Sq Mean Sq F value   Pr(>F)
## feed         5 231129   46226   15.37 5.94e-10 ***
## Residuals   65 195556    3009
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Alternativamente, possiamo anche eseguire l'ANOVA in questo modo. Notate che anche nel summary del modello lineare si può trovare la statistica F.

```
# Oppure:
mod = lm( weight ~ feed )
```

```r
summary( mod )
##
## Call:
## lm(formula = weight ~ feed)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -123.909  -34.413    1.571   38.170  103.091
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    323.583     15.834  20.436  < 2e-16 ***
## feedhorsebean -163.383     23.485  -6.957 2.07e-09 ***
## feedlinseed   -104.833     22.393  -4.682 1.49e-05 ***
## feedmeatmeal   -46.674     22.896  -2.039 0.045567 *
## feedsoybean    -77.155     21.578  -3.576 0.000665 ***
## feedsunflower    5.333     22.393   0.238 0.812495
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 54.85 on 65 degrees of freedom
## Multiple R-squared:  0.5417, Adjusted R-squared:  0.5064
## F-statistic: 15.36 on 5 and 65 DF,  p-value: 5.936e-10

# Meglio fare:
anova( mod )
## Analysis of Variance Table
##
## Response: weight
##           Df Sum Sq Mean Sq F value    Pr(>F)
## feed       5 231129   46226  15.365 5.936e-10 ***
## Residuals 65 195556    3009
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

detach(chickwts)
```

Osserviamo che la conferma che ci siano medie diverse fra i gruppi ci è data dai seguenti elementi:

ANOVA MANUALE P-value del test di differenza fra medie dei gruppi (5.93642e-10);

ANOVA AUTOMATICA P-value del comando ANOVA (5.94e-10);

LINEAR MODEL P-value del test di significatività dei regressori (5.936e-10).

REMARK: La statistica F può essere interpretata in generale come una statistica test sulla significatività totale di un modello di regressione. Immaginiamo di analizzare il solito summary di un modello lineare (anche con variabili continue). La statistica F che troviamo, testa l'ipotesi nulla che tutti i coefficienti della regressione siano uguali a zero. Viene testato il modello completo contro un modello senza variabili indipendenti (la stima della variabile dipendente è la media dei valori della variabile dipendente). Un valore alto di F implica che almeno qualche parametro della regressione è non nullo e che l'equazione della regressione ha validità nel fitting dei dati (cioè, le variabili indipendenti non sono puramente randomiche rispetto alla variabile dipendente).

Affermiamo quindi che c'è differenza delle medie fra i gruppi.

È interessante notare che SS_B calcolato a mano fa parte dell'output del comando summary dell'ANOVA. Lo stesso vale per per SS_W.

4. Costruzione della matrice disegno di ANOVA

Poniamoci nel caso di ONE-WAY ANOVA. Per verificare l'esistenza di diversi gruppi, di fatto quello che vogliamo fare è un modello di regressione lineare con una variabile *factor* (**variabile dummy**, categorica).

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

Cerchiamo di capire come mai il numero dei regressori sarà $g - 1$, quindi X avrà dimensioni $n \times g$ (perchè viene aggiunta l'intercetta).

Supponiamo di avere un campione di 18 osservazioni suddivisi in 7 gruppi con numerosità {3,2,3,2,3,2,3} rispettivamente, la matrice disegno X dovrebbe essere:

$$X = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Tuttavia questa matrice disegno (X.full nel codice) è singolare, cioè non invertibile. Per invertirla manualmente dobbiamo ricorrere alla pseudoinversa di Moore-Penrose.

In alternativa, possiamo scrivere la matrice disegno sotto forma di contrasto, ovvero rimuoviamo una colonna (un parametro), denotando gli elementi dell'ultimo gruppo come assenti in tutti gli altri nel seguente modo:

$$X = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

Notiamo subito che questa matrice è analoga alla precedente (nel senso che studio la significatività degli stessi regressori) ma è non singolare e quindi inveritibile.

REMARK Se facciamo `tapply(feed, feed, length)`, R calcola le numerosità dei gruppi e le riordina in ordine alfabetico di nome del gruppo. Se vogliamo utilizzare le numerosità nell'ordine in cui si presentano i gruppi nel dataset (feed), dobbiamo fare:

```
data( chickwts )
head( chickwts )
##   weight      feed
## 1    179 horsebean
## 2    160 horsebean
## 3    136 horsebean
## 4    227 horsebean
## 5    217 horsebean
## 6    168 horsebean
tail( chickwts )
##    weight   feed
## 66    352 casein
## 67    359 casein
## 68    216 casein
## 69    222 casein
## 70    283 casein
## 71    332 casein

attach( chickwts )
n  = length( feed )

group_names = unique( as.character( feed ) )
ng = tapply( feed, feed, length )[ group_names ]

ng
## horsebean   linseed   soybean sunflower  meatmeal    casein
##        10        12        14        12        11        12
```

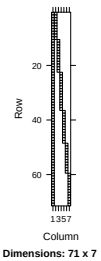
Costruiamo la matrice X.full, ovvero una matrice disegno dove consideriamo tutti i gruppi (dimensione =

$n \times (g+1)$).

```
# gruppo 1 (nell'ordine dei dati in ( weight,feed )
x1.full = c( rep( 1, ng [ 1 ] ),
             rep( 0, n - ng [ 1 ] ) )

# gruppo 2 (nell'ordine dei dati in ( weight,feed )
x2.full = c( rep( 0, ng [ 1 ] ),
             rep( 1, ng [ 2 ] ),
             rep( 0, n - ng [ 1 ] - ng [ 2 ] ) )

# gruppo 3 (nell'ordine dei dati in ( weight,feed )
x3.full = c( rep( 0, ng [ 1 ] + ng [ 2 ] ),
             rep( 1, ng [ 3 ] ),
             rep( 0, n - ng [ 1 ] - ng [ 2 ] - ng [ 3 ] ) )

# gruppo 4 (nell'ordine dei dati in ( weight,feed )
x4.full = c( rep( 0, n - ng [ 6 ] - ng [ 5 ] - ng [ 4 ] ),
             rep( 1, ng [ 4 ] ),
             rep( 0, ng [ 5 ]  + ng [ 6 ] ) )

# gruppo 5 (nell'ordine dei dati in ( weight,feed )
x5.full = c( rep( 0, n - ng [ 6 ] - ng [ 5 ] ),
             rep( 1, ng [ 5 ] ),
             rep( 0, ng [ 6 ] ) )

# gruppo 6 (nell'ordine dei dati in ( weight,feed )
x6.full = c( rep( 0, n - ng [ 6 ] ),
             rep( 1, ng [ 6 ] ) )

X.full = cbind( rep( 1, n ),
                    x1.full,
                    x2.full,
                    x3.full,
                    x4.full,
                    x5.full,
                    x6.full )
```

Visualizziamo questa matrice disegno.

```
#corrplot( X.full, corr = F, method = 'ellipse' )
image(Matrix(X.full))
```



Vediamo che non ha rango pieno (proviamo che una colonna è combinazione lineare di un'altra).

```
X.full[ , 1 ] - rowSums( X.full[ , - 1 ] )
##  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [39] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Stimiamo ora i $\hat{\boldsymbol{\beta}}$. Ricordiamo che X è singolare, quindi la matrice di proiezione H sarà calcolata come segue:

$$H = X \cdot (X^T \cdot X)^{\dagger} \cdot X^T$$

in cui $(X^T \cdot X)^{\dagger}$ indica la pseudo-inversa di Moore-Penrose. E i $\hat{\boldsymbol{\beta}}$ saranno calcolati come:

$$\hat{\boldsymbol{\beta}} = (X^T \cdot X)^{\dagger} \cdot X^T \cdot \mathbf{y}$$

```
# H.full = X.full %*% solve( t( X.full ) %*% X.full ) %*% t( X.full )
# R dà errore, perchè singolare!

H.full = X.full %*% ginv( t( X.full ) %*% X.full ) %*% t( X.full )

y  = weight

betas.full = as.numeric( ginv( t( X.full ) %*% X.full ) %*% t( X.full ) %*% y )
betas.full
## [1] 222.112523 -61.912523  -3.362523  24.316048 106.804143  54.796568 101.470810
```

`ginv` calcola la matrice pseudo-inversa di Moore-Penrose di una matrice.

La media nel gruppo j-esimo è:

$$\mathbb{E}[y_j] = \mu_j = \beta_0 + \beta_j \qquad j = 1 : g$$

```
means_by_group = betas.full[ 1 ] + betas.full[ 2:length( betas.full ) ]
names( means_by_group ) = group_names

means_by_group
## horsebean   linseed   soybean sunflower  meatmeal    casein
##  160.2000  218.7500  246.4286  328.9167  276.9091  323.5833
tapply( weight, feed, mean )[ unique( as.character( feed ) ) ]
## horsebean   linseed   soybean sunflower  meatmeal    casein
##  160.2000  218.7500  246.4286  328.9167  276.9091  323.5833
```

La media globale è:

$$\mu = \sum_{j=1}^{g} \frac{n_j \cdot \mu_j}{n}$$

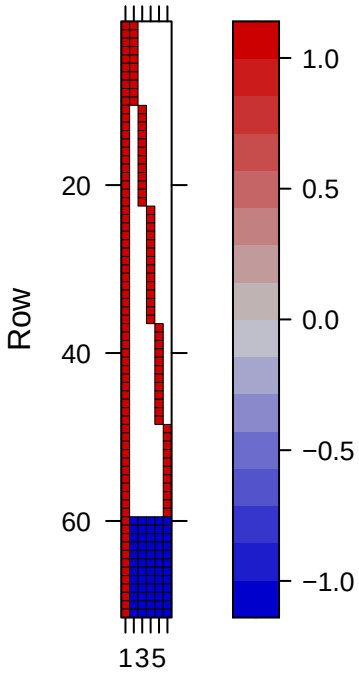
```
global_mean = ng %*% means_by_group / n
global_mean
##          [,1]
## [1,] 261.3099

mean( weight )
## [1] 261.3099
```

Ora, costruiamo la matrice disegno X basata sui contrasti, ovvero esprimendo l'ultimo gruppo come la negazione di tutti gli altri.

```
x1.red = c( rep( 1, ng [ 1 ] ),
            rep( 0, n - ng [ 1 ] - ng [ 6 ] ),
            rep( -1, ng [ 6 ] ) )
stopifnot( sum( x1.red - ( x1.full - x6.full ) ) == 0 )

x2.red = c( rep( 0, ng [ 1 ] ),
            rep( 1, ng [ 2 ] ),
            rep( 0, n - ng [ 1 ] - ng [ 2 ] - ng [ 6 ] ),
            rep( -1, ng [ 6 ] ) )
stopifnot( sum( x2.red - ( x2.full - x6.full ) ) == 0 )

x3.red = c( rep( 0, ng [ 1 ] + ng [ 2 ] ),
            rep( 1, ng [ 3 ] ),
            rep( 0, n - ng [ 1 ] - ng [ 2 ] - ng [ 3 ] - ng [ 6 ] ),
            rep( -1, ng [ 6 ] ) )
stopifnot( sum( x3.red - ( x3.full - x6.full ) ) == 0 )

x4.red = c( rep( 0, n - ng [ 6 ] - ng [ 5 ] - ng [ 4 ] ),
            rep( 1, ng [ 4 ] ),
            rep( 0, ng [ 5 ] ),
            rep( -1, ng [ 6 ] ) )
stopifnot( sum( x4.red - ( x4.full - x6.full ) ) == 0 )

x5.red = c( rep( 0, n - ng [ 6 ] - ng [ 5 ] ),
            rep( 1, ng [ 5 ] ),
            rep( -1, ng [ 6 ] ) )
stopifnot( sum( x5.red - ( x5.full - x6.full ) ) == 0 )


X.red  = cbind( rep( 1, n ),
                x1.red,
                x2.red,
                x3.red,
                x4.red,
                x5.red )
```

Visualizziamo questa matrice di dimensione $n \times g$.

```
#corrplot( X.red, corr = F, method = 'ellipse' )
image(Matrix(X.red))
```



Dimensions: 71 x 6

Stimiamo ora i $\hat{\beta}$.

```
# solve( t( X.red ) %*% X.red )
# solve( t( X.red ) %*% X.red ) - ginv( t( X.red ) %*% X.red )

H.red  = X.red %*% solve( t( X.red ) %*% X.red ) %*% t( X.red )

betas.red = as.numeric( solve( t( X.red ) %*% X.red ) %*% t( X.red ) %*% y )
betas.red
## [1] 259.13128 -98.93128 -40.38128 -12.70271  69.78539  17.77781
```

La media nel gruppo i-esimo si ottiene nel seguente modo:

$$\mu_j = \beta_0 + \beta_i \qquad i = 1, ..., g-1$$

$$\mu_g = \beta_0 - (\beta_1 + ... + \beta_{g-1})$$

```
means_by_group = betas.red[ 1 ] + betas.red[ -1 ]
means_by_group = c( means_by_group, betas.red[ 1 ] - sum( betas.red[ -1 ] ) )

means_by_group.full = betas.full[ 1 ] + betas.full[ -1 ]
means_by_group.full = c( means_by_group.full, betas.full[ 1 ] - sum( betas.full[ -1 ] ) )

names( means_by_group ) = group_names
means_by_group
## horsebean   linseed   soybean sunflower  meatmeal    casein
##  160.2000  218.7500  246.4286  328.9167  276.9091  323.5833
```

```
tapply( weight, feed, mean )[ group_names ]
## horsebean   linseed   soybean sunflower  meatmeal    casein
##  160.2000  218.7500  246.4286  328.9167  276.9091  323.5833

names( means_by_group.full ) = group_names
means_by_group.full
##    horsebean      linseed      soybean    sunflower     meatmeal       casein
## 1.602000e+02 2.187500e+02 2.464286e+02 3.289167e+02 2.769091e+02 3.235833e+02
##         <NA>
## 8.526513e-14
```

In questo caso abbiamo in tutti i casi dei risultati molto coerenti (nonostante le approssimazioni).

REMARK Ma cosa fa `R` in automatico?

```
mod_aov = aov( weight ~ feed )
X_aov = model.matrix( mod_aov )
```

Vediamo che la matrice disegno creata dall'ANOVA è di dimensioni $n \times g$. Notiamo che manca la variabile (livello) `casein` che è usata come baseline.

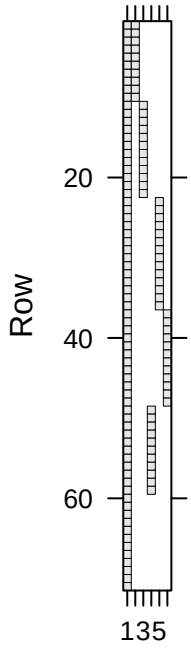
```
mod_lm = lm( weight ~ feed )
X_lm = model.matrix( mod_lm )
```

Idem nel caso di modello lineare.

Visualizziamo la matrice disegno dell'ANOVA e del modello lineare implementate in `R`.

```
#corrplot( X_lm, corr = F, method = 'ellipse' )

par(mfrow=c(1,2))
image( Matrix( X_aov )  , main='Matrice disegno anova')
```

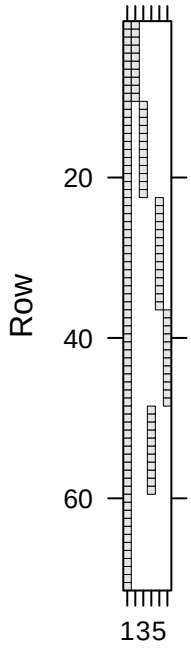
Matrice disegno anova



Dimensions: 71 x 6

```
image( Matrix( X_lm )  , main='Matrice disegno lm')
```

Matrice disegno lm



Dimensions: 71 x 6

```
par(mfrow=c(1,1))

levels(feed)
## [1] "casein"    "horsebean" "linseed"   "meatmeal"  "soybean"   "sunflower"
unique(feed)
## [1] horsebean linseed   soybean   sunflower meatmeal  casein
## Levels: casein horsebean linseed meatmeal soybean sunflower
```

REMARK Il **primo gruppo** (nell'ordine *alfanumerico* dei livelli della variabile di stratificazione, feed, e NON nell'ordine di comparsa dei dati) viene soppresso e preso come riferimento (baseline).

Calcoliamo ora i $\hat{\beta}$.

```
betas.lm = coefficients( mod_lm )
```

La media nel gruppo j-esimo si ottiene nel seguente modo:

$$\mu_{baseline} = \beta_0$$

$$\mu_j = \beta_0 + \beta_j, \qquad j \neq baseline$$

```
means_by_group = c( betas.lm[ 1 ], betas.lm[ 1 ] + betas.lm[ -1 ] )
names( means_by_group ) = levels( feed )

means_by_group
##    casein horsebean   linseed  meatmeal   soybean sunflower
```

```
##  323.5833  160.2000  218.7500  276.9091  246.4286  328.9167
tapply( weight, feed, mean )
##    casein horsebean   linseed  meatmeal   soybean sunflower
##  323.5833  160.2000  218.7500  276.9091  246.4286  328.9167
detach(chickwts)
```

5. One-way ANOVA (II)

The example dataset we will use is a set of 24 blood coagulation times. 24 animals were randomly assigned to four different diets and the samples were taken in a random order. This data comes from Box, Hunter, and Hunter (1978).

```
coagulation= read.table(file='coagulation.txt', header=T)

str( coagulation )
## 'data.frame':    24 obs. of  2 variables:
##  $ coag: int  62 60 63 59 63 67 71 64 65 66 ...
##  $ diet: chr  "A" "A" "A" "A" ...
dim( coagulation )
## [1] 24  2
names( coagulation )
## [1] "coag" "diet"
head( coagulation )
##   coag diet
## 1   62    A
## 2   60    A
## 3   63    A
## 4   59    A
## 5   63    B
## 6   67    B

coagulation$diet = as.factor(coagulation$diet)
str( coagulation )
## 'data.frame':    24 obs. of  2 variables:
##  $ coag: int  62 60 63 59 63 67 71 64 65 66 ...
##  $ diet: Factor w/ 4 levels "A","B","C","D": 1 1 1 1 2 2 2 2 2 2 ...
```

The first step is to plot the data to check for:

1. Normality assumption;

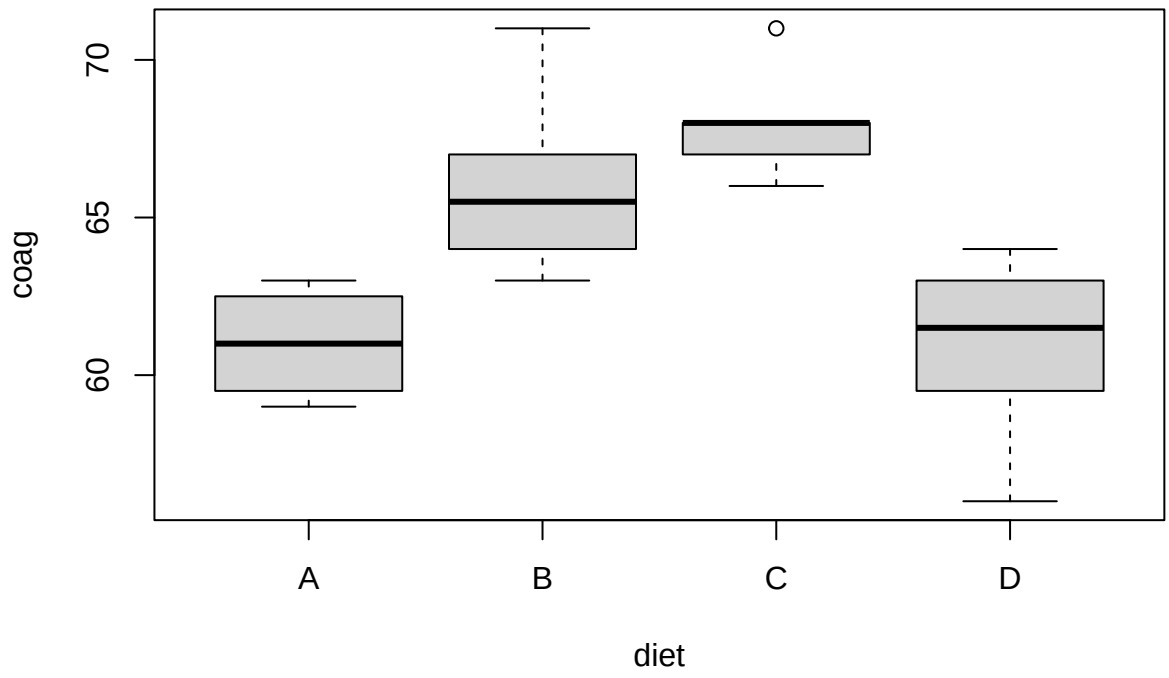
2. Equal variances for each level of the factor.

We don't want to detect:

1. Skewness - this will be suggested by an asymmetrical form of the boxes.

2. Unequal variance - this will be suggested by unequal box sizes. Some care is required because often there is very little data to be used in the construction of the boxplots and so even when the variances are truly equal in the groups, we can expect a bit variability.

```
boxplot( coag ~ diet, data = coagulation )
```



In this case, there are no obvious problems. For group C, there are only 4 distinct observations and one is somewhat separated which accounts for the slightly odd looking plot. Always look at sample sizes.

```
table( coagulation$diet )
##
## A B C D
## 4 6 6 8
```

Anyway, let's check assumptions.

```
Ps = tapply(coagulation$coag, coagulation$diet, function( x ) ( shapiro.test( x )))

Ps
## $A
##
##  Shapiro-Wilk normality test
##
## data:  x
## W = 0.94971, p-value = 0.7143
##
##
## $B
##
##  Shapiro-Wilk normality test
##
## data:  x
## W = 0.92239, p-value = 0.5227
```

```
##
##
## $C
##
##  Shapiro-Wilk normality test
##
## data:  x
## W = 0.87279, p-value = 0.2375
##
##
## $D
##
##  Shapiro-Wilk normality test
##
## data:  x
## W = 0.93173, p-value = 0.5319
```

```
# Alternative: Levene-Test
leveneTest( coagulation$coag, coagulation$diet )
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  3  0.6492 0.5926
##       20
```

We accept normality and homoscedasticity.

Now let's fit the model.

```
mod = lm( coag ~ diet, coagulation )
summary( mod )
##
## Call:
## lm(formula = coag ~ diet, data = coagulation)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -5.00  -1.25   0.00   1.25   5.00
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.100e+01  1.183e+00  51.554  < 2e-16 ***
## dietB       5.000e+00  1.528e+00   3.273 0.003803 **
## dietC       7.000e+00  1.528e+00   4.583 0.000181 ***
## dietD       2.719e-15  1.449e+00   0.000 1.000000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.366 on 20 degrees of freedom
## Multiple R-squared:  0.6706, Adjusted R-squared:  0.6212
## F-statistic: 13.57 on 3 and 20 DF,  p-value: 4.658e-05
```

What kind of design matrix has been used in this case? Look at the design matrix to understand the coding:

```
model.matrix( mod ) #n x g
##    (Intercept) dietB dietC dietD
## 1            1     0     0     0
```

```
## 2            1     0     0     0
## 3            1     0     0     0
## 4            1     0     0     0
## 5            1     1     0     0
## 6            1     1     0     0
## 7            1     1     0     0
## 8            1     1     0     0
## 9            1     1     0     0
## 10           1     1     0     0
## 11           1     0     1     0
## 12           1     0     1     0
## 13           1     0     1     0
## 14           1     0     1     0
## 15           1     0     1     0
## 16           1     0     1     0
## 17           1     0     0     1
## 18           1     0     0     1
## 19           1     0     0     1
## 20           1     0     0     1
## 21           1     0     0     1
## 22           1     0     0     1
## 23           1     0     0     1
## 24           1     0     0     1
## attr(,"assign")
## [1] 0 1 1 1
## attr(,"contrasts")
## attr(,"contrasts")$diet
## [1] "contr.treatment"
```

The effects returned by ANOVA have to be interpreted as *differences* from a reference level, the baseline (the first in alphabetical order).

What do we conclude by looking at the p-value?

We can read the output in this way: Group A is the reference level (**baseline**) and has a mean equal to 61 ($\hat{\beta}_0$), groups B, C and D are 5 ($\hat{\beta}_1$), 7 ($\hat{\beta}_2$) and 0 ($\hat{\beta}_3$) seconds larger on average.

For completeness, look at:

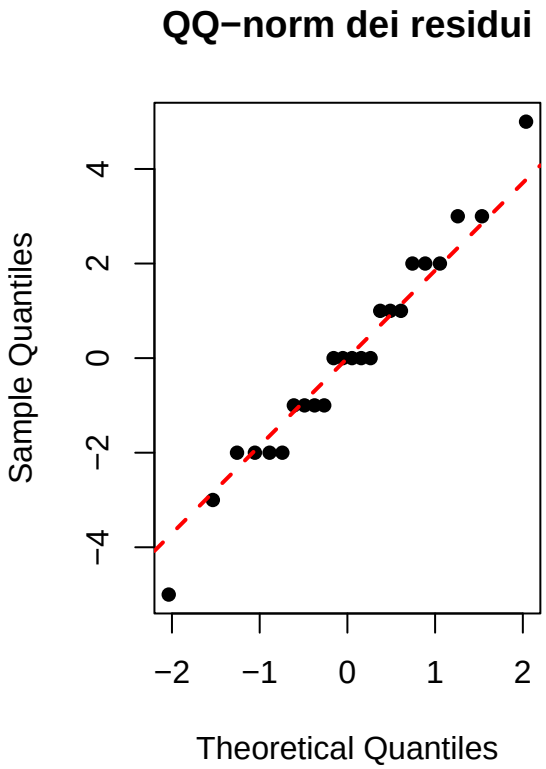
```
anova(mod)
## Analysis of Variance Table
##
## Response: coag
##           Df Sum Sq Mean Sq F value    Pr(>F)
## diet       3    228    76.0  13.571 4.658e-05 ***
## Residuals 20    112     5.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

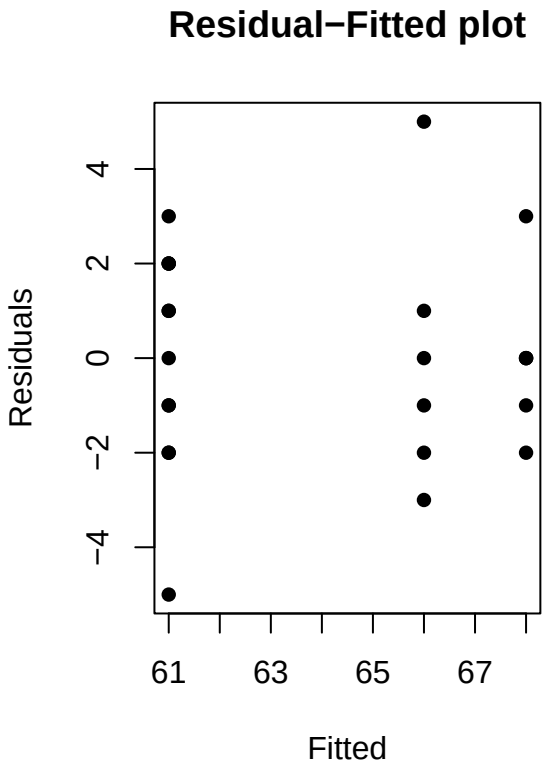
We have evidence of the fact that different groups have significantly different means.

Diagnostics

```
par( mfrow = c(1,2) )

qqnorm( mod$res, pch = 16, col = 'black', main = 'QQ-norm dei residui' )
```

```r
qqline( mod$res, lwd = 2, col = 'red', lty = 2 )

shapiro.test( mod$res )
##
##  Shapiro-Wilk normality test
##
## data:  mod$res
## W = 0.97831, p-value = 0.8629

plot( mod$fit, mod$res, xlab = "Fitted", ylab = "Residuals", main = "Residual-Fitted plot",
      pch = 16 )
```





Since data are integers and fitted values are integers too, a discrete-like pattern must be expected in the QQ plot. Of course, discrete data cannot be normally distributed. However, here residuals are approximately normal and so we can go ahead with the inference.

The discrete behaviour in the residuals and fitted values shows up in the residual-fitted plot because we can see fewer points than the sample size. This is due to the overplotting of points' symbols. In this case, we see no heteroscedasticity in residuals.

6. Two-ways ANOVA

Two-ways ANOVA design is thought for variance decomposition in cases where we have two factors, and not only one as before.

The model we want to use here is

$$y_{ijk} = \mu + \tau_j + \gamma_k + \alpha_{jk} + \varepsilon_{ijk}$$

where $i \in \{1, .., n_{jk}\}$ is the index of the statistical units inside the group identified in the class j of the first factor and k of the second, with $j \in \{1, .., g_1\}$, $j \in \{1, .., g_2\}$. $\tau_j, \gamma_k, \alpha_{jk}$ are related to the average deviances of different groups with respect to the global mean μ. The interaction effect α_{jk} is interpreted as that part of the mean response not attributable to the additive effect of τ_j and γ_k. For example, you may enjoy strawberries and cream individually, but the combination is superior. In contrast, you may like fish and ice cream but not together.

As part of an investigation of toxic agents, 48 rats were allocated to

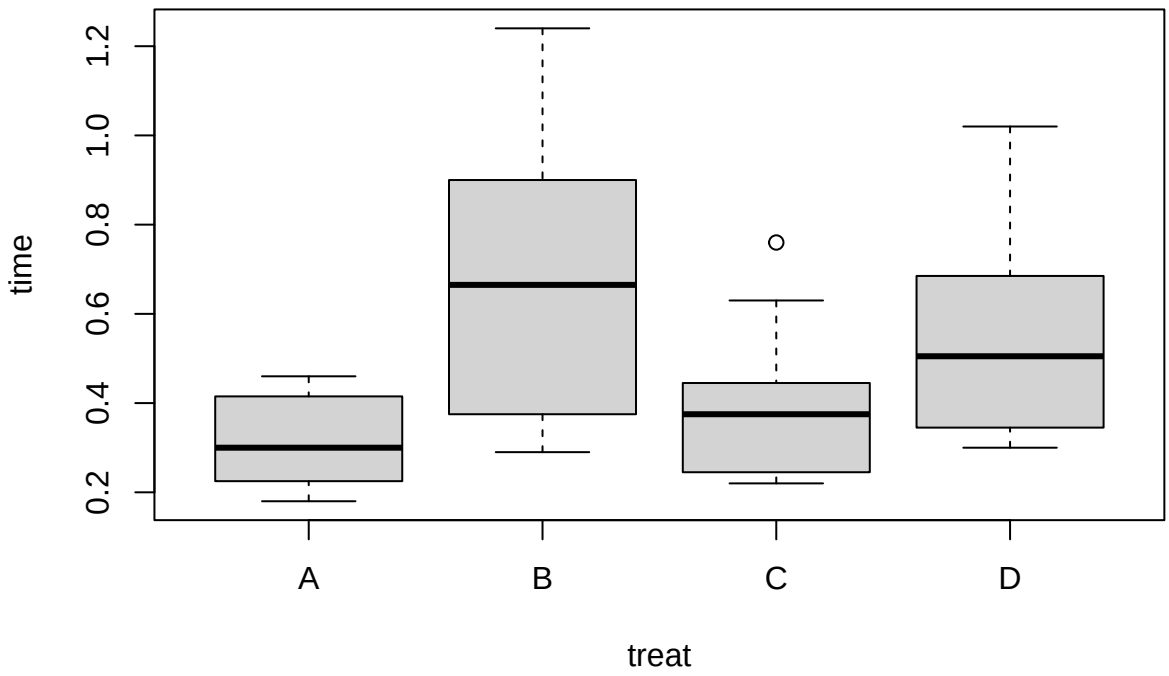
- 3 poisons (I, II, III) and
- 4 treatments (A, B, C, D).

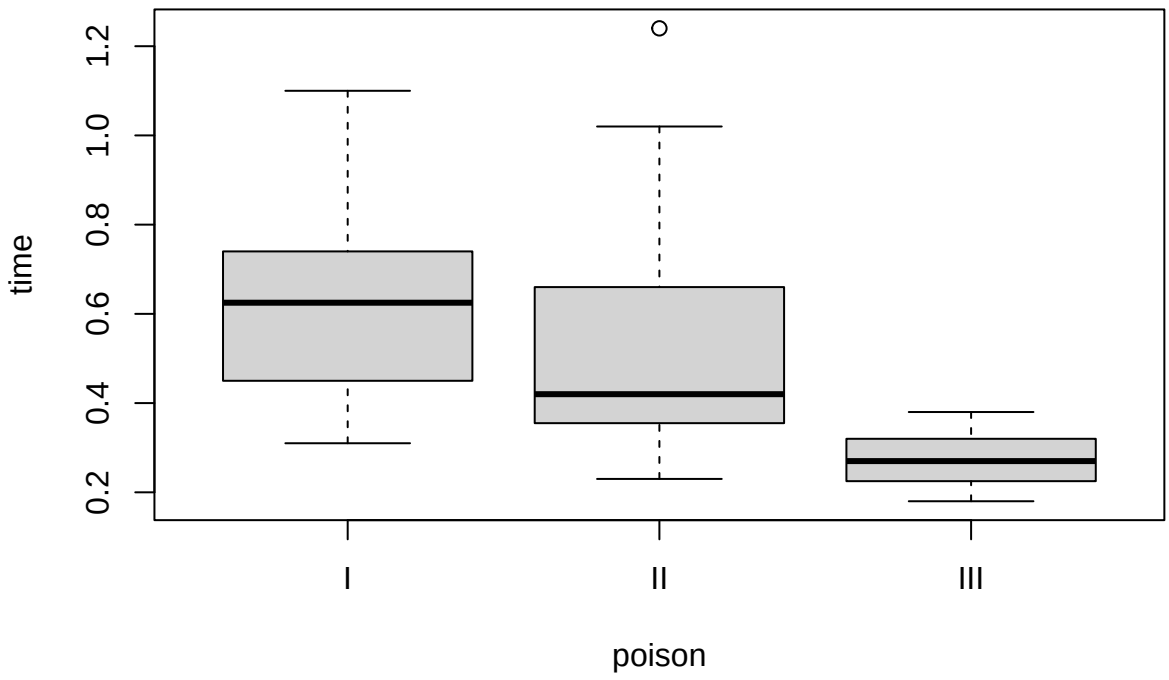
The response was survival time in tens of hours.

```
rats =read.table(file='rats.txt', header=T)

str( rats )
## 'data.frame':    48 obs. of  3 variables:
##  $ time  : num  0.31 0.82 0.43 0.45 0.45 1.1 0.45 0.71 0.46 0.88 ...
##  $ poison: chr  "I" "I" "I" "I" ...
##  $ treat : chr  "A" "B" "C" "D" ...
rats$poison= as.factor(rats$poison)
rats$treat= as.factor(rats$treat)

head( rats )
##   time poison treat
## 1 0.31      I     A
## 2 0.82      I     B
## 3 0.43      I     C
## 4 0.45      I     D
## 5 0.45      I     A
## 6 1.10      I     B
tail( rats )
##    time poison treat
## 43 0.24    III     C
## 44 0.31    III     D
## 45 0.23    III     A
## 46 0.29    III     B
## 47 0.22    III     C
## 48 0.33    III     D
names( rats )
## [1] "time"   "poison" "treat"
```

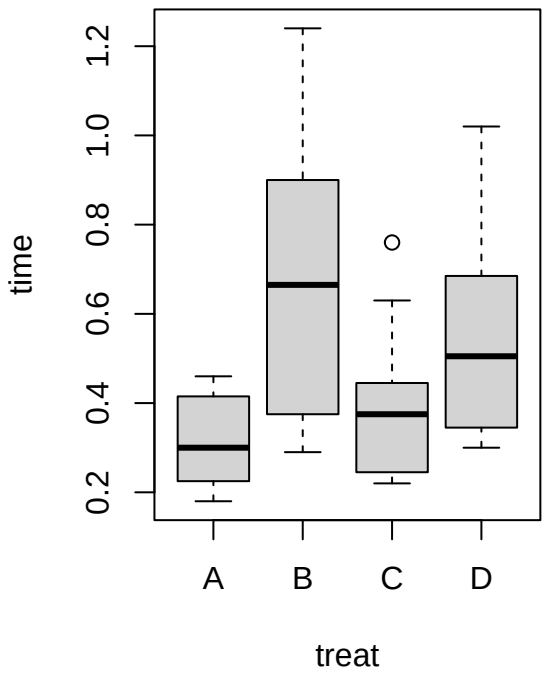
Some automatic plots.

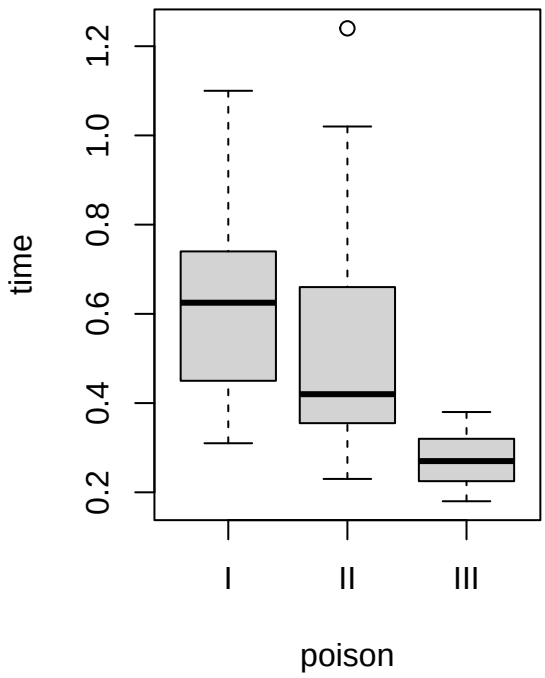
```
plot( time ~ treat + poison, data = rats )
```


time
0.2
0.4
0.6
0.8
1.0
1.2
A
B
C
D
treat

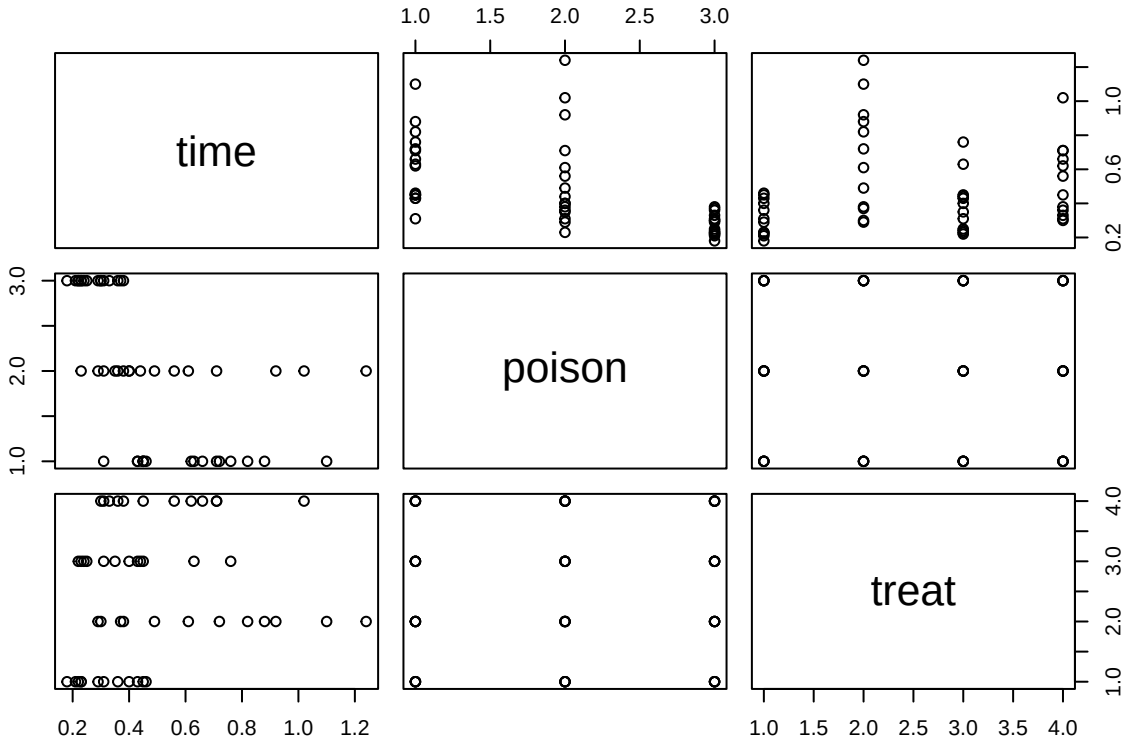



```
par( mfrow = c( 1,2 ) )
boxplot( time ~ treat, data = rats )
boxplot( time ~ poison, data = rats )
```





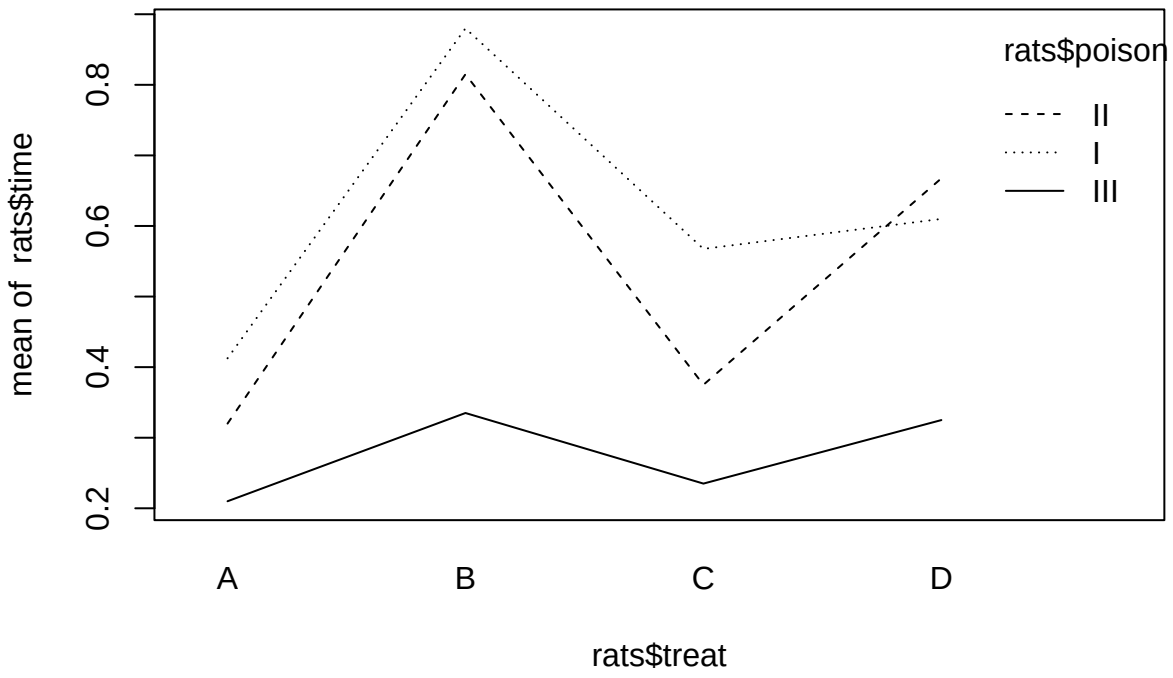

```
pairs(rats) #poco interpretabile
```



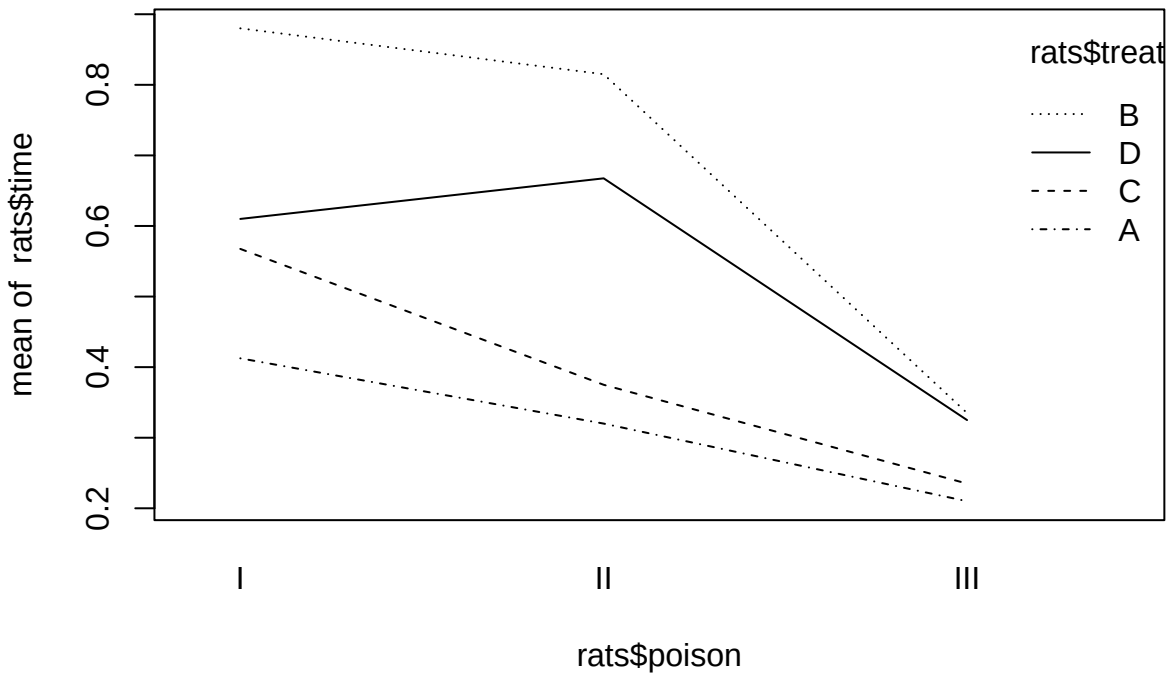
Some evidence of skewness can be seen, especially since it appears that variance is in some way related to the mean response.

Check for an interaction using graphical methods:

```
#help(interaction.plot)

interaction.plot( rats$treat, rats$poison, rats$time )
```




```
interaction.plot( rats$poison, rats$treat, rats$time )
```



Parallel lines would suggest the absence of interaction, yet it is not always easy to figure it out with these plots.

Before applying a two-way ANOVA, we have to test the following hypotheses:

- NORMALITY (in all groups, $3 \times 4 = 12$ tests);
- HOMOGENEOUS VARIANCES (among groups).

```
tapply( rats$time, rats$treat:rats$poison, function( x ) shapiro.test( x )$p )
##        A:I       A:II      A:III        B:I       B:II      B:III        C:I
## 0.07414486 0.84756406 0.57735490 0.69983383 0.70083721 0.17057001 0.40503490
##       C:II      C:III        D:I       D:II      D:III
## 0.92091109 0.97187706 0.42739119 0.90650963 0.68893644

leveneTest( rats$time, rats$treat:rats$poison )
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value    Pr(>F)
## group 11  4.1323 0.0005833 ***
##       36
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
bartlett.test( rats$time, rats$treat:rats$poison )
##
##  Bartlett test of homogeneity of variances
##
## data:  rats$time and rats$treat:rats$poison
## Bartlett's K-squared = 45.137, df = 11, p-value = 4.59e-06
```

We notice that normality is respected in all 12 groups (even thogh we observe low p-value for the first group A-I). The variances homogeneity is violated (see p-value of Levene's test).

It could be possible to consider variable transformation. We should try a Box-Cox transformation for the output variable, considering the complete model.

We fit the complete model considering **interactions** between considered factors.

REMARK We assumed that the effect on dependent variables of increasing one explanatory variables is independent of effect of other explanatories (indeed, explanatory variables are called independent variables). In general we modeled as:

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon$$

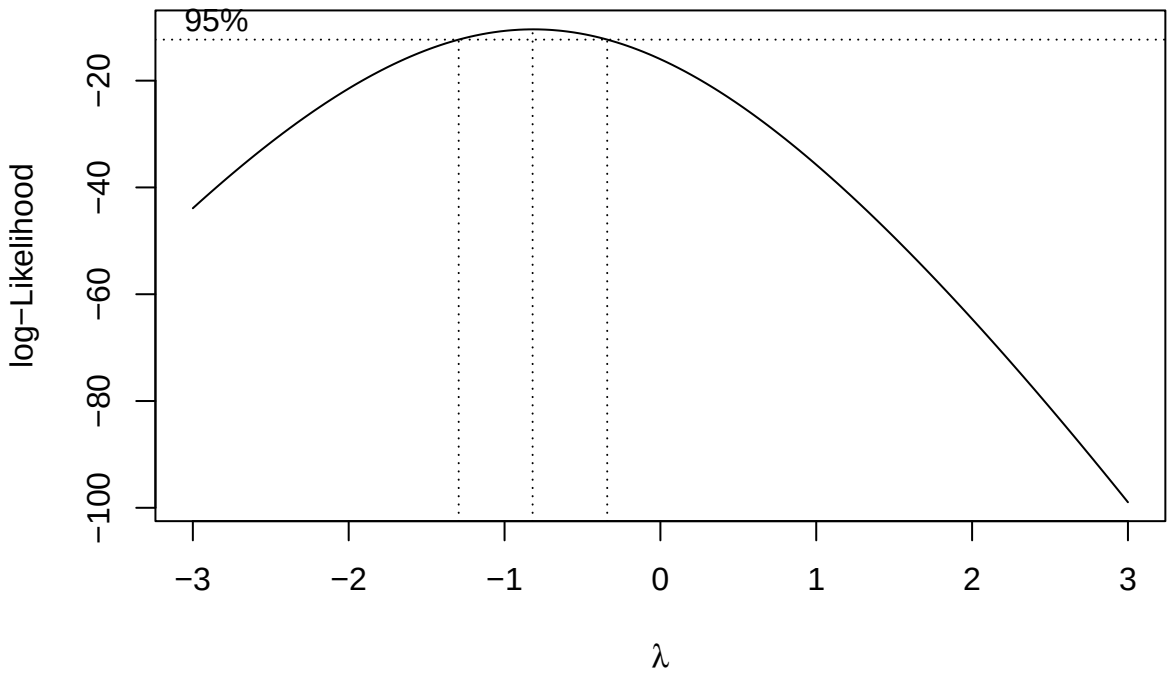
Imagine that as x_1 and x_2 increase, the change related to y_i is not linear with respect to both these two terms. A classic example is related to marketing: if we spend money in advertising, we do not have an increasing of sales always proportional to the amount of money spent for TV and social media, since the audience on different platforms is different. In marketing, this is known as a synergy effect, and in statistics this is referred to as an interaction effect. One way of extending this model to allow for interaction effects is to include a third predictor, called interaction term, which is constructed by computing the product of x_1 and x_2. This can be written as:

$$y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 \times x_2 + \varepsilon$$

```
g = lm( time ~ poison * treat, rats )
#"*" gives the full model: linear effect AND interaction
#g = lm( time ~ poison + treat + poison : treat , rats )

summary( g )
##
## Call:
## lm(formula = time ~ poison * treat, data = rats)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.32500 -0.04875  0.00500  0.04312  0.42500
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.41250    0.07457   5.532 2.94e-06 ***
## poisonII          -0.09250    0.10546  -0.877   0.3862
## poisonIII         -0.20250    0.10546  -1.920   0.0628 .
## treatB             0.46750    0.10546   4.433 8.37e-05 ***
## treatC             0.15500    0.10546   1.470   0.1503
## treatD             0.19750    0.10546   1.873   0.0692 .
## poisonII:treatB    0.02750    0.14914   0.184   0.8547
## poisonIII:treatB -0.34250    0.14914  -2.297   0.0276 *
## poisonII:treatC  -0.10000    0.14914  -0.671   0.5068
## poisonIII:treatC -0.13000    0.14914  -0.872   0.3892
## poisonII:treatD   0.15000    0.14914   1.006   0.3212
## poisonIII:treatD -0.08250    0.14914  -0.553   0.5836
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1491 on 36 degrees of freedom
## Multiple R-squared:  0.7335, Adjusted R-squared:  0.6521
## F-statistic:  9.01 on 11 and 36 DF,  p-value: 1.986e-07
anova( g )
```

```
## Analysis of Variance Table
##
## Response: time
##              Df  Sum Sq Mean Sq F value    Pr(>F)
## poison        2 1.03301 0.51651 23.2217 3.331e-07 ***
## treat         3 0.92121 0.30707 13.8056 3.777e-06 ***
## poison:treat  6 0.25014 0.04169  1.8743    0.1123
## Residuals    36 0.80073 0.02224
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
b = boxcox( g, lambda = seq(-3,3,by=0.01) )
```




```
best_lambda = b$x[ which.max( b$y ) ]
best_lambda
## [1] -0.82
```

The best λ is -0.82, which we approximate to -1 (we can interpret the reciprocal as the death rate).

```
tapply( 1/rats$time, rats$treat:rats$poison, function( x ) shapiro.test( x )$p )
##        A:I       A:II      A:III        B:I       B:II      B:III        C:I
## 0.03115001 0.65891022 0.38884991 0.95061185 0.79724850 0.17554581 0.38264156
##       C:II      C:III        D:I       D:II      D:III
## 0.87818060 0.96578666 0.16801940 0.84342484 0.78353223

leveneTest( 1/rats$time, rats$treat:rats$poison )
## Levene's Test for Homogeneity of Variance (center = median)
```

```
##       Df F value Pr(>F)
## group 11  1.1272 0.3698
##       36
bartlett.test(  1/rats$time, rats$treat:rats$poison )
##
##  Bartlett test of homogeneity of variances
##
## data:  1/rats$time and rats$treat:rats$poison
## Bartlett's K-squared = 9.8997, df = 11, p-value = 0.5394
```

After Box-Cox transformation (without scale-location adjustment), assumptions are respected!

```
g1 = lm( 1/time ~ poison * treat, rats )

summary(g1)
##
## Call:
## lm(formula = 1/time ~ poison * treat, data = rats)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.76847 -0.29642 -0.06914  0.25458  1.07936
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       2.48688    0.24499  10.151 4.16e-12 ***
## poisonII          0.78159    0.34647   2.256 0.030252 *
## poisonIII         2.31580    0.34647   6.684 8.56e-08 ***
## treatB           -1.32342    0.34647  -3.820 0.000508 ***
## treatC           -0.62416    0.34647  -1.801 0.080010 .
## treatD           -0.79720    0.34647  -2.301 0.027297 *
## poisonII:treatB  -0.55166    0.48999  -1.126 0.267669
## poisonIII:treatB -0.45030    0.48999  -0.919 0.364213
## poisonII:treatC   0.06961    0.48999   0.142 0.887826
## poisonIII:treatC  0.08646    0.48999   0.176 0.860928
## poisonII:treatD  -0.76974    0.48999  -1.571 0.124946
## poisonIII:treatD -0.91368    0.48999  -1.865 0.070391 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.49 on 36 degrees of freedom
## Multiple R-squared:  0.8681, Adjusted R-squared:  0.8277
## F-statistic: 21.53 on 11 and 36 DF,  p-value: 1.289e-12
anova(g1)
## Analysis of Variance Table
##
## Response: 1/time
##              Df Sum Sq Mean Sq F value    Pr(>F)
## poison        2 34.877 17.4386 72.6347 2.310e-13 ***
## treat         3 20.414  6.8048 28.3431 1.376e-09 ***
## poison:treat  6  1.571  0.2618  1.0904    0.3867
## Residuals    36  8.643  0.2401
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

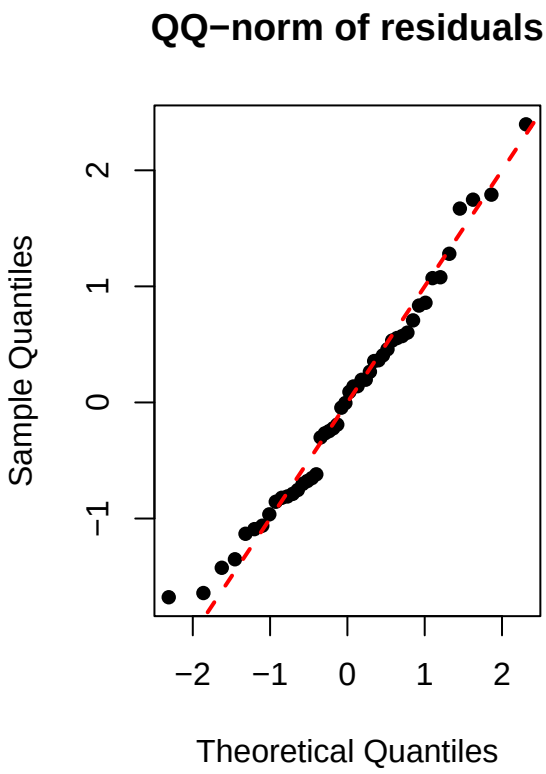
The results tell us that levels of different factors have significantly different means taken singularly, but interactions are not significant.

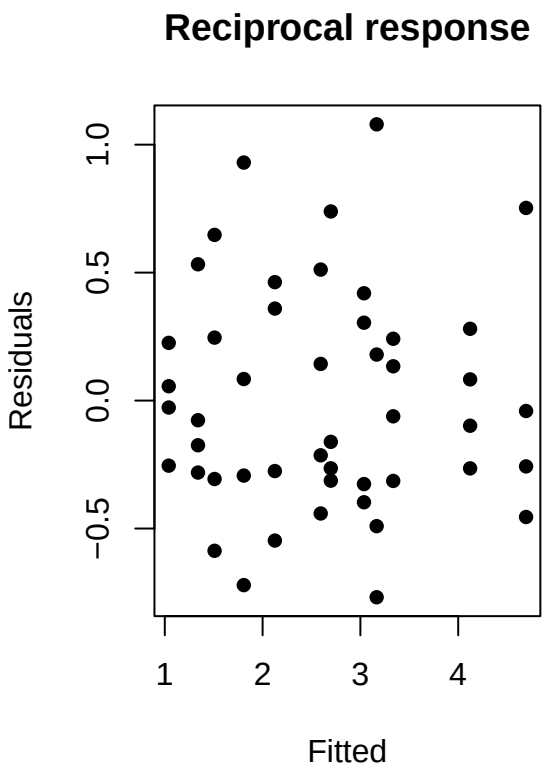
Then, we should not consider the interaction:

```
g1_sel = lm( 1/time ~ poison + treat, data = rats )
summary( g1_sel )
##
## Call:
## lm(formula = 1/time ~ poison + treat, data = rats)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.82757 -0.37619  0.02116  0.27568  1.18153
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.6977     0.1744  15.473  < 2e-16 ***
## poisonII      0.4686     0.1744   2.688  0.01026 *
## poisonIII     1.9964     0.1744  11.451 1.69e-14 ***
## treatB       -1.6574     0.2013  -8.233 2.66e-10 ***
## treatC       -0.5721     0.2013  -2.842  0.00689 **
## treatD       -1.3583     0.2013  -6.747 3.35e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4931 on 42 degrees of freedom
## Multiple R-squared:  0.8441, Adjusted R-squared:  0.8255
## F-statistic: 45.47 on 5 and 42 DF,  p-value: 6.974e-16
anova( g1_sel )
## Analysis of Variance Table
##
## Response: 1/time
##           Df Sum Sq Mean Sq F value    Pr(>F)
## poison     2 34.877 17.4386  71.708 2.865e-14 ***
## treat      3 20.414  6.8048  27.982 4.192e-10 ***
## Residuals 42 10.214  0.2432
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We should check the hypotheses (residual normality and homoscedasticity) of the model.

```
par( mfrow = c( 1, 2 ) )
qqnorm( g1_sel$res/summary( g1_sel )$sigma, pch = 16, main = 'QQ-norm of residuals'  )
abline( 0, 1, lwd = 2, lty = 2, col = 'red' )

shapiro.test( g1_sel$res )
##
##  Shapiro-Wilk normality test
##
## data:  g1_sel$res
## W = 0.97918, p-value = 0.5451

plot( g1_sel$fitted, g1$res, xlab = "Fitted", ylab = "Residuals",
      main = "Reciprocal response", pch = 16 )
```





The hypotheses are respected.