Laboratorio con R - 2

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

30/05/2019

---

0. Librerie

```
library( MASS )
library( car ) #for LEVENE TEST
## Warning: package 'car' was built under R version 3.4.4
## Loading required package: carData
## Warning: package 'carData' was built under R version 3.4.4
library( faraway )
## Warning: package 'faraway' was built under R version 3.4.4
##
## Attaching package: 'faraway'
## The following objects are masked from 'package:car':
##
##     logit, vif
library( Matrix )
```

---

1. Visualizzazione della decomposizione della varianza

Analizzare il dataset CICLISTI andando a visualizzare in maniera efficacie le seguenti quantità di interesse:

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

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

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

dove con y_i si sono indicati i valori della variabile risposta (**Car**) in corrispondenza di ciascun valore x_i (**Center**), 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 del dataset:

```
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]
```

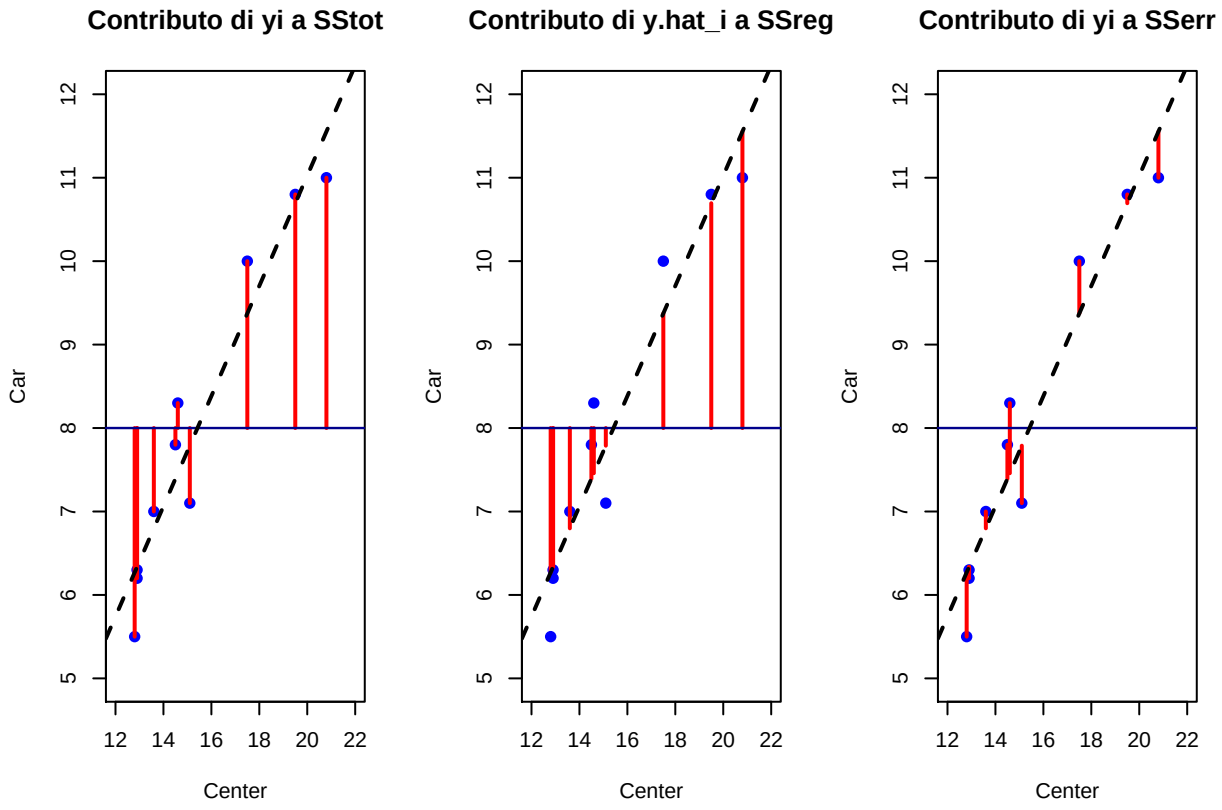
Fittiamo ora un modello lineare semplice e calcoliamo le quantità di interesse.

```
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 predetti/stimati dal modello )
y.hat = reg.ciclisti$fitted.value
```

```
y.hat
##         1         2         3         4         5         6         7
##  6.269904  6.335939  6.335939  6.798178  7.392485  7.458520  7.788691
##         8         9        10
##  9.373511 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 yi 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 yi 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 )
```



---

2. One-way ANOVA

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.

```
str( chickwts )
## 'data.frame':    71 obs. of  2 variables:
##  $ weight: num  179 160 136 227 217 168 108 124 143 140 ...
##  $ feed  : Factor w/ 6 levels "casein","horsebean",..: 2 2 2 2 2 2 2 2 2 2 ...
head( chickwts )
##   weight      feed
## 1    179 horsebean
## 2    160 horsebean
## 3    136 horsebean
## 4    227 horsebean
## 5    217 horsebean
## 6    168 horsebean
tail( chickwts )
```

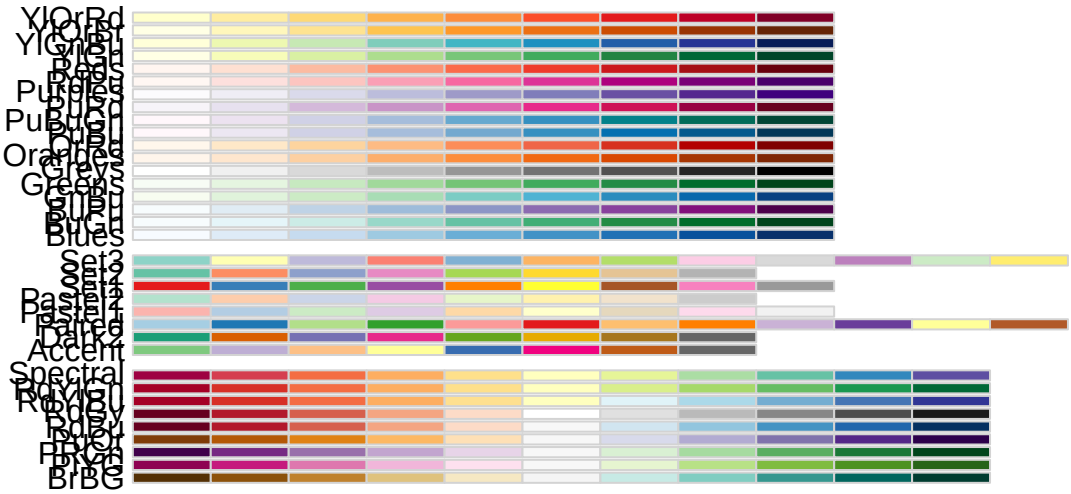
```
##    weight   feed
## 66    352 casein
## 67    359 casein
## 68    216 casein
## 69    222 casein
## 70    283 casein
## 71    332 casein
```

```
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

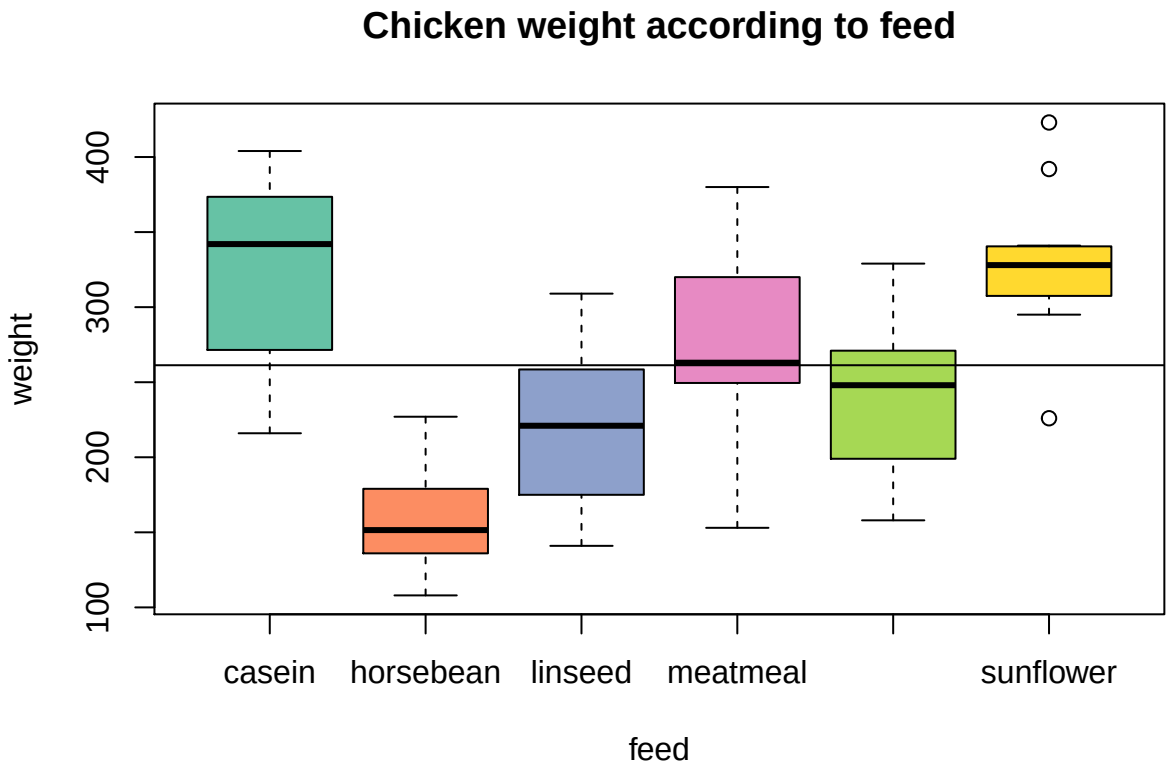
Consiglio grafico: spesso è utile rappresentare i gruppi di dati con colori diversi, potete trovare alcune palette di colori nel pacchetto RColorBrewer

```
library(RColorBrewer)

display.brewer.all()
```




```
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

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




```
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:

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

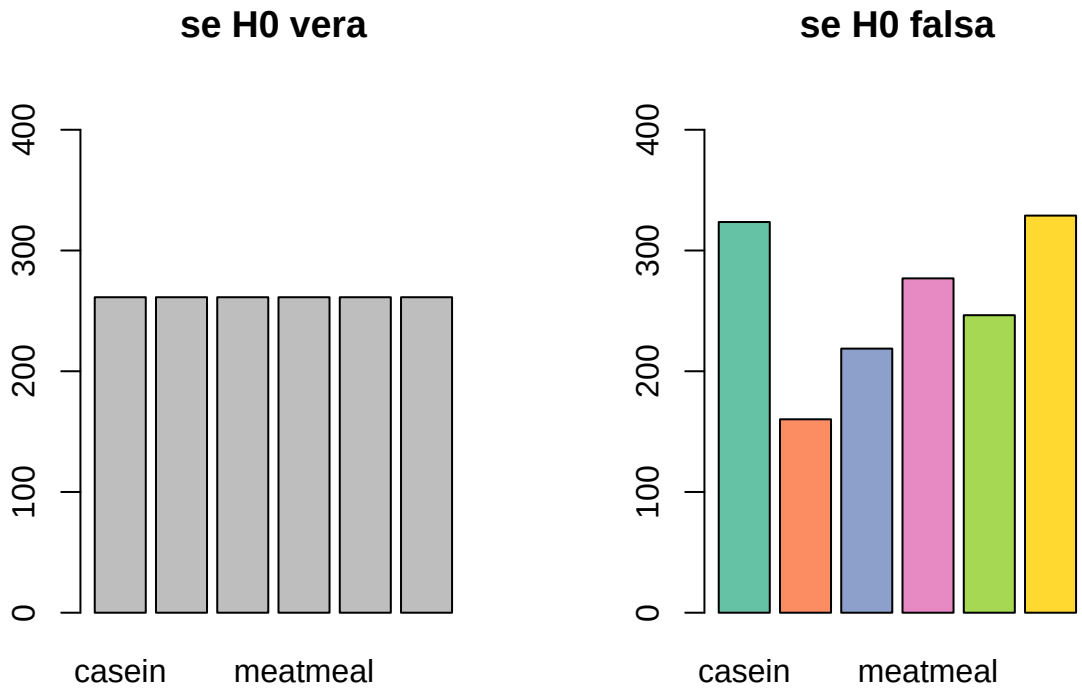
in cui $i \in \{1, .., n_j\}$ è l'indice dell'unità statistica all'interno del grupp j, mentre $j \in \{1, .., g\}$ è l'indice di gruppo.

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.

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

barplot( rep( mean( weight ), 6 ), names.arg = levels( feed ),
         ylim = c( 0, max( weight ) ), main = "se H0 vera", col = 'grey' )

barplot( tapply( weight, feed, mean ), names.arg = levels( feed ),
         ylim = c( 0, max( weight ) ), main = "se H0 falsa", col = my_colors )
```



Facciamo un' ANOVA manuale.

Verifichiamo che siano soddisfatte le ipotesi dell'ANOVA:

- **Normalità** intragruppo;
- **Omoschedasticità** fra i gruppi.

```
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

# 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

# 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

# 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
```

Bartlett's test

Il Bartlett test è il seguente:

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

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

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).

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

$$F_0 = \frac{SS_{TREAT}/r}{SS_{RES}/(n-p)} \sim F(r, n-p)$$

in cui:

$$SS_{TREAT} = \sum_{j=1}^{g} (\mu_j - \mu)^2 \cdot n_j$$

Nella one-way ANOVA il numero di regressori $r = g - 1$.

```
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 )
SStreat = sum( ng * ( Mediag-Media )^2 )
SSres  = SStot - SStreat

alpha = 0.05
Fstatistic = ( SStreat / ( g-1 ) ) / ( SSres / ( n-g ) )

# valori "piccoli" non ci portano a rifiutare
cfr.fisher = qf( 1-alpha, g-1, n-g )
Fstatistic > cfr.fisher
## [1] TRUE
Fstatistic # siamo proprio ben oltre la soglia = > evidenza forte per rifiutare
## [1] 15.3648
cfr.fisher
## [1] 2.356028

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

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

```
help( aov )
## starting httpd help server ... done

# Costruiamo un modelo anova e guardiamo il summary
fit = aov( weight ~ feed )
```

```r
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

# Oppure:
mod = lm( weight ~ feed )
# Qui tuttavia dobbiamo sapere dove guardare per trovare le informazioni di prima.
# Confrontare il p-value del test di significatività globale.
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
```

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).

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

E' interessante notare che SS_{TREAT} calcolato a mano fa parte dell'output del comando summary dell'ANOVA. Idem per SS_{RES}.

---

3. Riflessione costruzione matrice disegno 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**).

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

Se avessimo 7 gruppi con numerosità {3,2,3,2,3,2,3} rispettivamente, la matrice disegno X sarebbe:

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

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:

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

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 grupponSe vogliamo utilizzare le numerosità nell'ordine in cui si presentano i gruppi nel dataset (feed), dobbiamo fare:

```
n
## [1] 71

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

treat
## [1] "casein"    "horsebean" "linseed"   "meatmeal"  "soybean"   "sunflower"
```

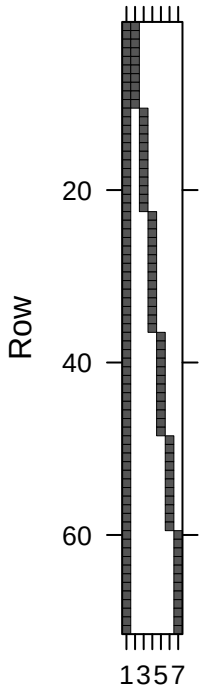
Costruiamo la matrice Xfull, ovvero una matrice disegno dove consideriamo tutti i gruppi (dimensione = n x (g + 1)).

```
# 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
## [36] 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
## [71] 0
```

Stimiamo ora i $\hat{\boldsymbol{\beta}}$. Ricordiamo che X è singolare, quindi H sarà calcolato 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, perché 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 )
```

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
```

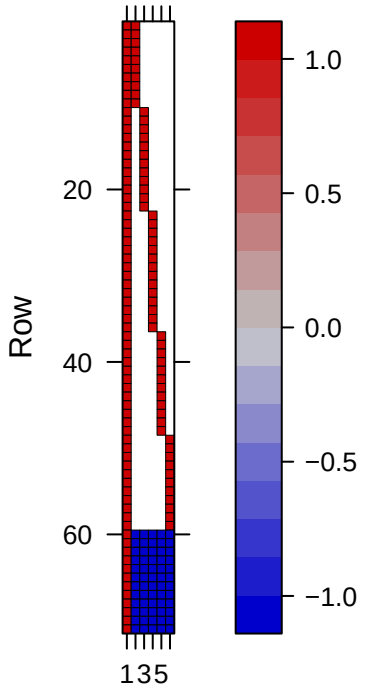
Costruiamo la matrice disegno basata sui contrasti.

```
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. Dimensione = n x g.

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



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

```
solve( t( X.red ) %*% X.red )
##                            x1.red        x2.red      x3.red        x4.red
##         0.0142316017  0.002435065 -0.0003427128 -0.00232684 -0.0003427128
## x1.red  0.0024350649  0.080898268 -0.0163239538 -0.01433983 -0.0163239538
## x2.red -0.0003427128 -0.016323954  0.0697871573 -0.01156205 -0.0135461760
## x3.red -0.0023268398 -0.014339827 -0.0115620491  0.06185065 -0.0115620491
## x4.red -0.0003427128 -0.016323954 -0.0135461760 -0.01156205  0.0697871573
## x5.red  0.0009199134 -0.017586580 -0.0148088023 -0.01282468 -0.0148088023
##                x5.red
##         0.0009199134
## x1.red -0.0175865801
## x2.red -0.0148088023
## x3.red -0.0128246753
## x4.red -0.0148088023
## x5.red  0.0748376623
solve( t( X.red ) %*% X.red ) - ginv( t( X.red ) %*% X.red )
##                              x1.red        x2.red        x3.red
##        -5.204170e-18  4.336809e-19  2.168404e-19  0.000000e+00
## x1.red  8.673617e-19  4.163336e-17 -1.040834e-17  3.469447e-18
## x2.red -7.589415e-19 -6.938894e-18 -1.387779e-17  0.000000e+00
## x3.red  8.673617e-19  6.938894e-18 -3.469447e-18  6.938894e-18
## x4.red -1.029992e-18 -6.938894e-18 -3.469447e-18  1.387779e-17
## x5.red -9.757820e-19 -2.428613e-17  4.163336e-17 -6.938894e-18
##                x4.red        x5.red
##        -3.252607e-19  3.252607e-19
```

```
## x1.red  2.081668e-17  3.469447e-18
## x2.red  1.561251e-17 -8.673617e-18
## x3.red  8.673617e-18 -6.938894e-18
## x4.red -1.387779e-17  3.469447e-18
## x5.red -5.030698e-17  2.775558e-17

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 )
```

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 ] ) )

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
```

Abbiamo quindi visto che sia con la matrice disegno singolare che non giungiamo allo stesso risultato.

REMARK Ma cosa fa `R` in automatico?

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

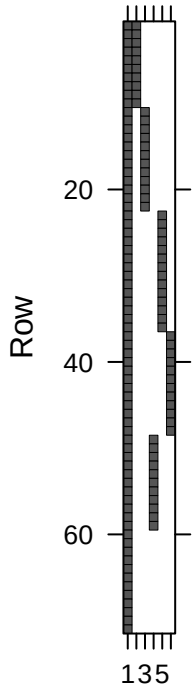
Vediamo che la matrice disegno creata dall'ANOVA è di dimensioni n x g. Notiamo che manca la variabile (livello) `casein` che è usata come baseline.

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

Idem nel caso di modello lineare.

Visualizziamo la matrice disegno dell'ANOVA implementata in `R`.

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



Dimensions: 71 x 6

```
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 comparizione dei dati) viene soppresso e preso come riferimento (baseline).

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

```
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
```

4. One-way ANOVA

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).

```
data( coagulation )

str( coagulation )
## 'data.frame':    24 obs. of  2 variables:
##  $ coag: num  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 ...
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
```

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

1. Normality assumption;

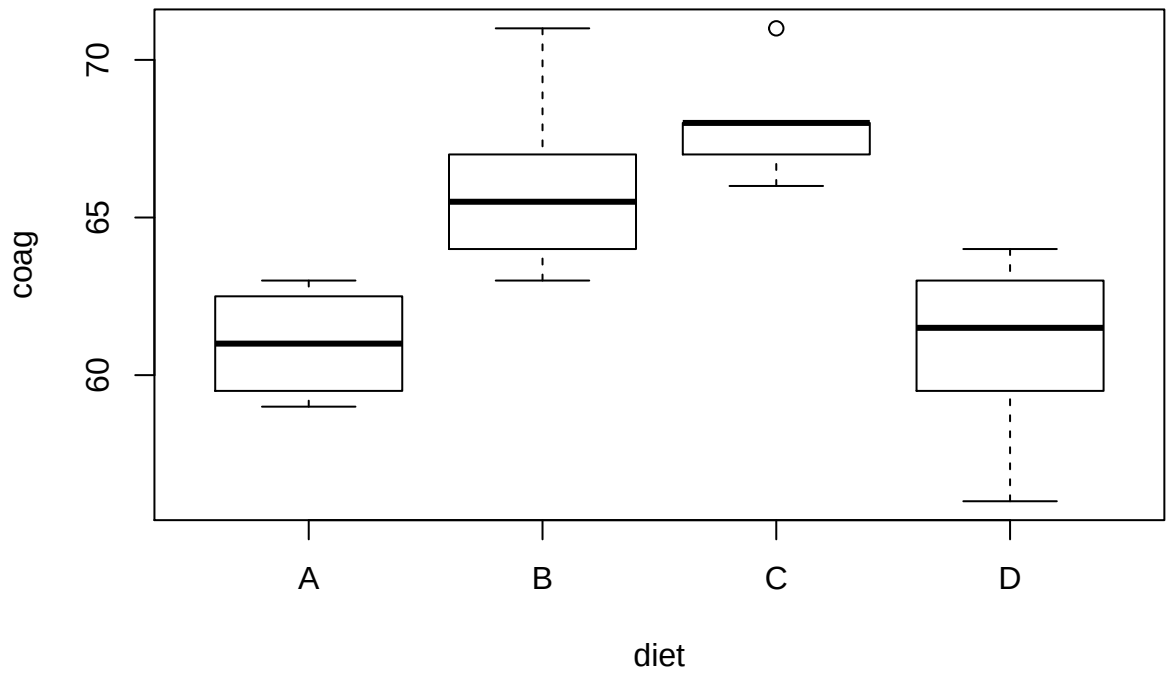
2. Equal variances for each level of the factor.

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.

```
plot( 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

coagulation$coag[ coagulation$diet == 'C' ]
## [1] 68 66 71 67 68 68
unique( coagulation$coag[ coagulation$diet == 'C' ] )
## [1] 68 66 71 67
```

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       -1.071e-14  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"
```

Effects have to be interpreted as differences from a reference level (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, groups B, C and D are 5, 7 and 0 seconds larger on average.

We can fit the model without an intercept term.

```
mod_i = lm( coag ~ diet - 1, coagulation )

summary( mod_i )
##
```

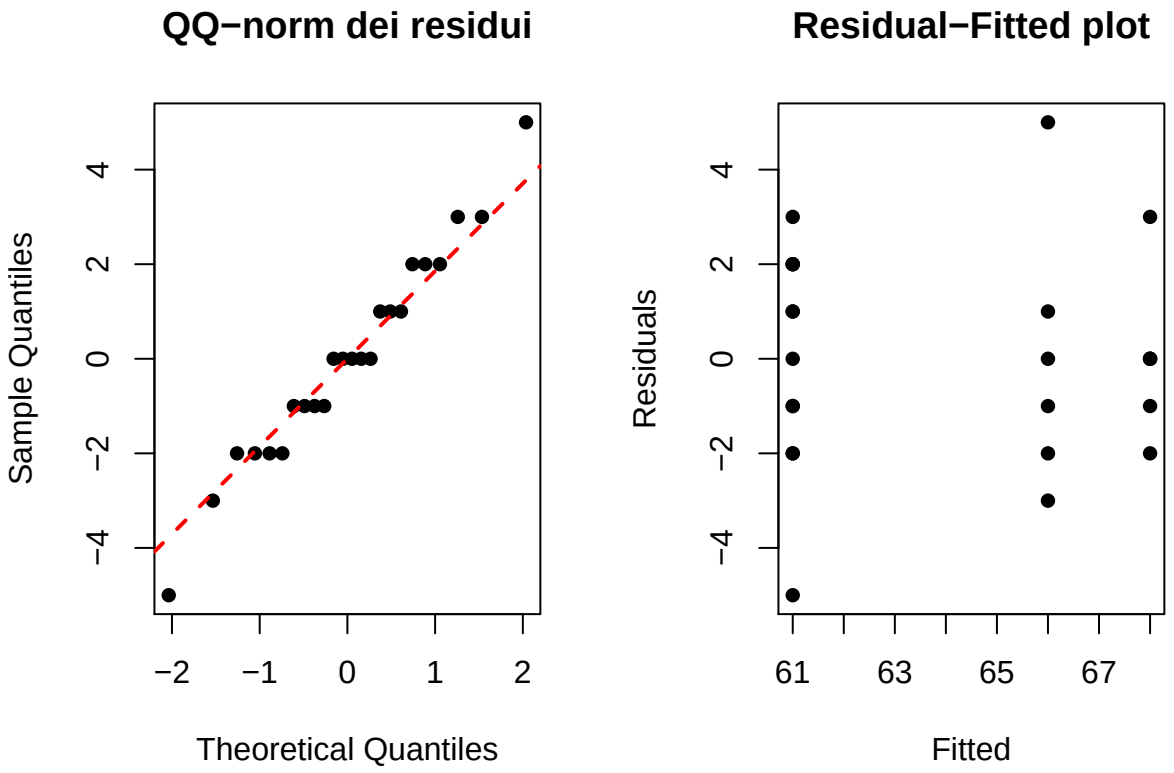
```
## Call:
## lm(formula = coag ~ diet - 1, data = coagulation)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -5.00  -1.25   0.00   1.25   5.00
##
## Coefficients:
##       Estimate Std. Error t value Pr(>|t|)
## dietA  61.0000     1.1832   51.55   <2e-16 ***
## dietB  66.0000     0.9661   68.32   <2e-16 ***
## dietC  68.0000     0.9661   70.39   <2e-16 ***
## dietD  61.0000     0.8367   72.91   <2e-16 ***
## ---
## 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:  0.9989, Adjusted R-squared:  0.9986
## F-statistic:  4399 on 4 and 20 DF,  p-value: < 2.2e-16

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

Now, without the intercept, each coefficients estimates the effect in that group

We can directly read the level means. Tests are not useful since they involve comparisons with zero. Note the **miscalculation** of R^2 (which is derived on the assumption that the intercept is present).

Diagnostics

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

qqnorm( mod$res, pch = 16, col = 'black', main = 'QQ-norm dei residui' )
qqline( mod$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 can't 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.

POST ANOVA ANALYSIS

For investigating differences between groups there are other techniques, such as:

DOE (design of experiments)

Blocking

Factorial Design and Latin Squares

---

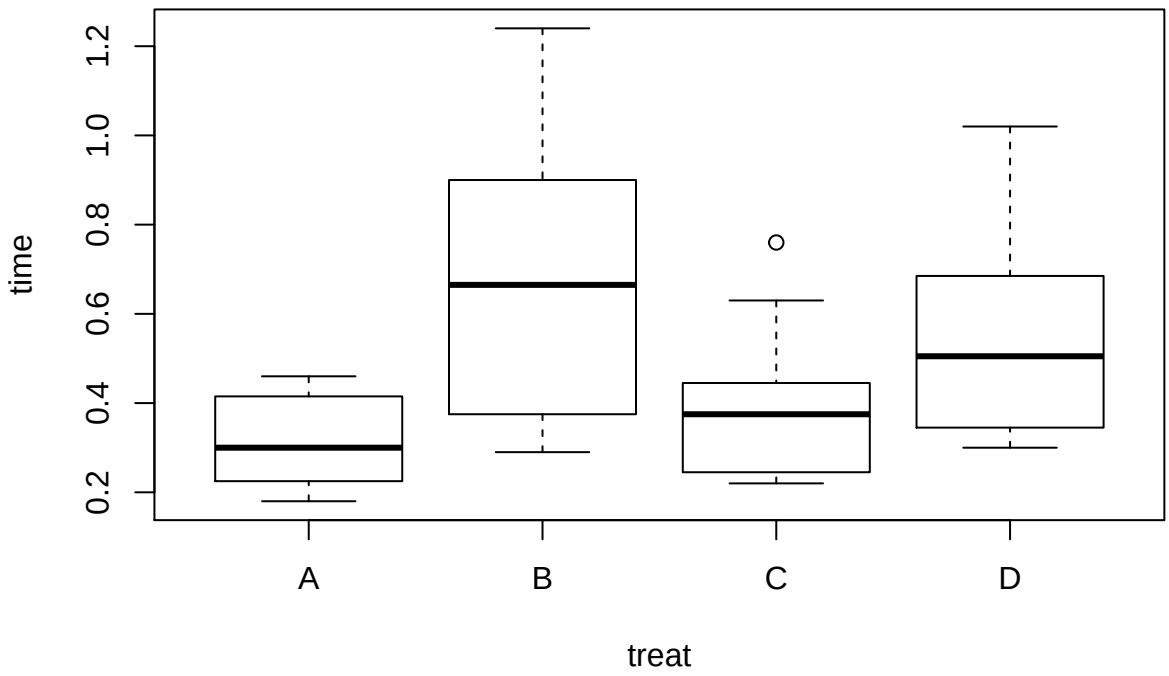
5. Two-ways ANOVA

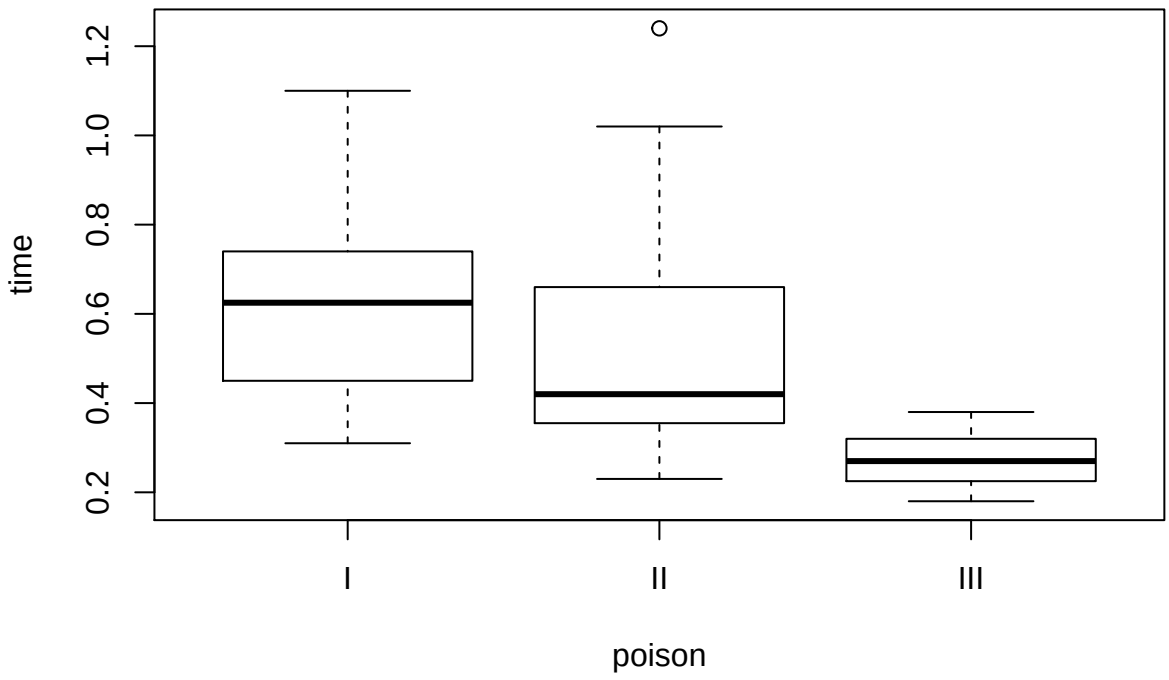
Two-way anova design where there are 4 replicates. As part of an investigation of toxic agents, 48 rats were allocated to - 3 poisons (I, II, III) and - 4 treatments (A, B, C, D). The response was survival time in tens of hours.

```
data( rats )

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: Factor w/ 3 levels "I","II","III": 1 1 1 1 1 1 1 1 1 1 ...
##  $ treat : Factor w/ 4 levels "A","B","C","D": 1 2 3 4 1 2 3 4 1 2 ...

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"
```

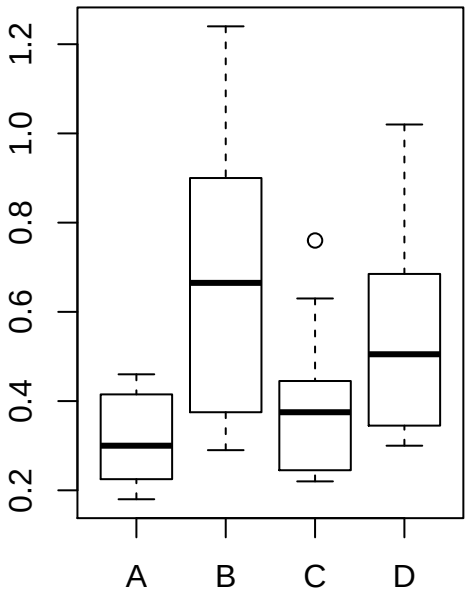
Some automatic plots.

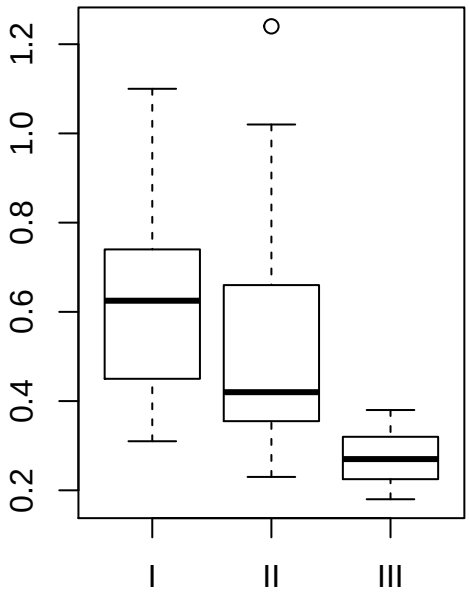
```
plot( time ~ treat + poison, data = rats )
```


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

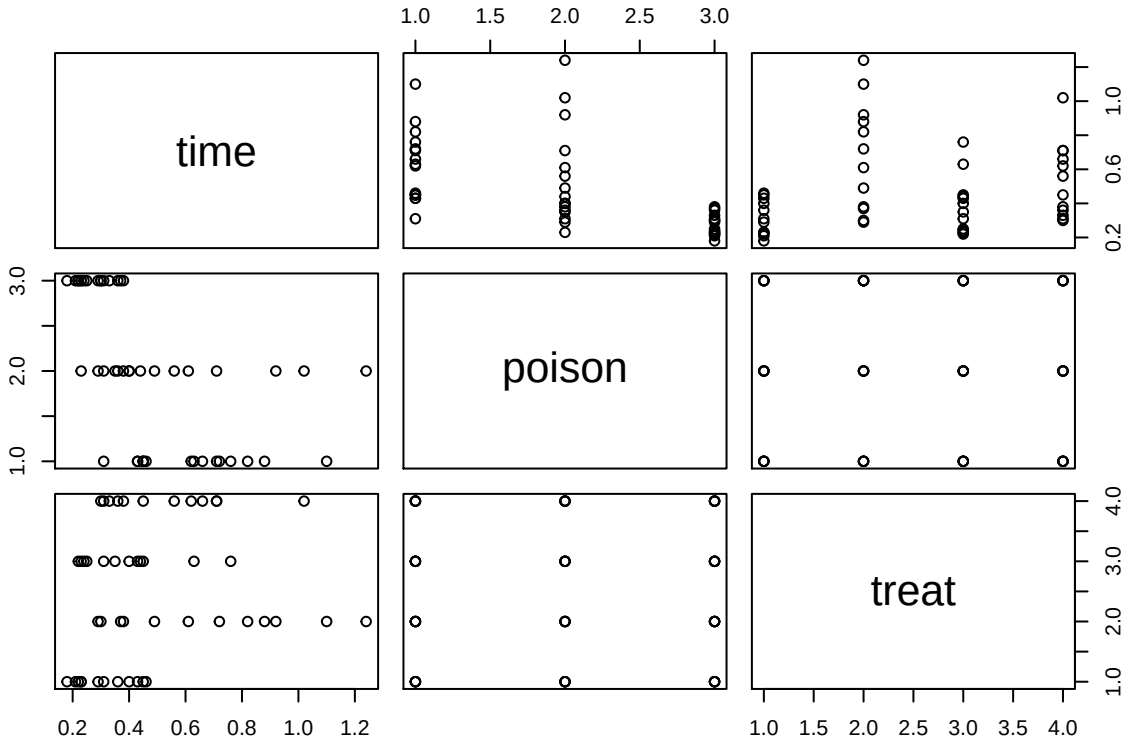



```
par( mfrow = c( 1,2 ) )
boxplot( time ~ treat, data = rats )
boxplot( time ~ poison, data = rats )
```





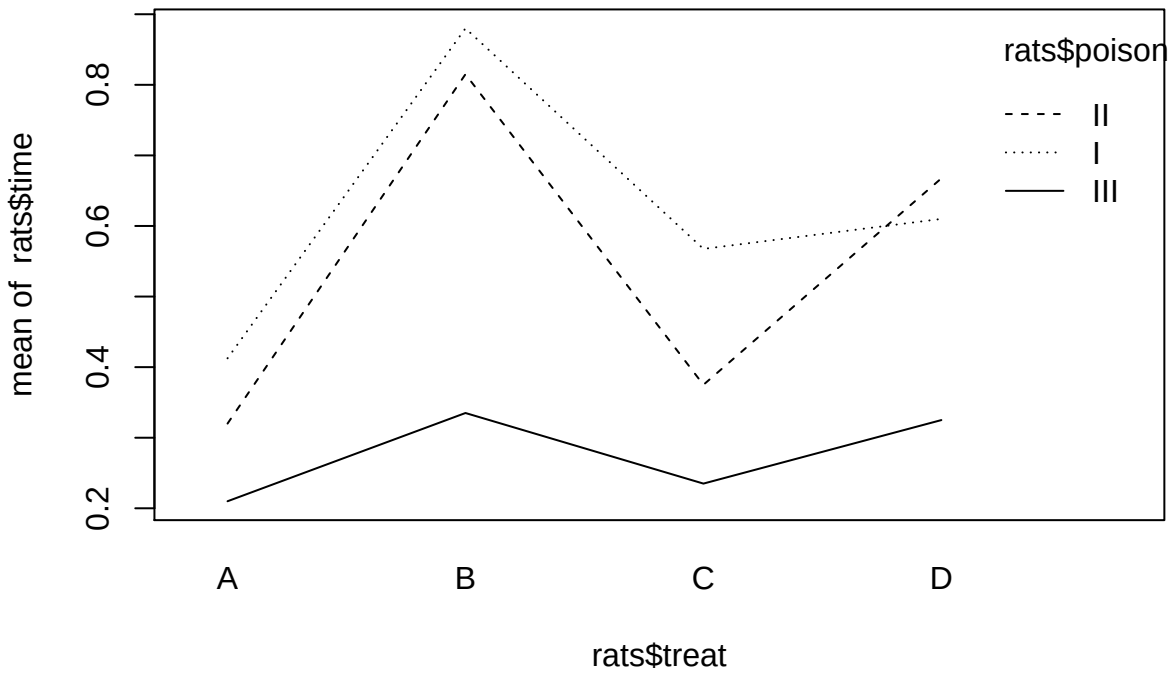

```r
pairs(rats) #poco interpretabile
```



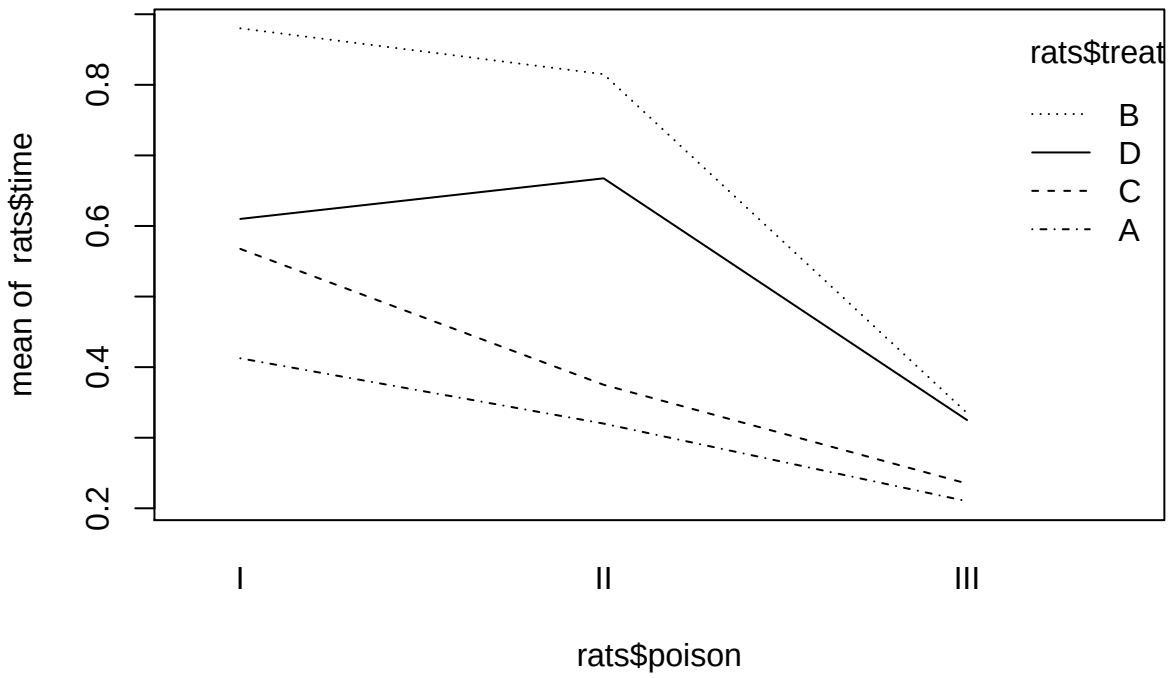
Some evidence of skewness can be seen, especially since it appears that variance is in some way related to the mean response.

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, 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
## 0.07414486 0.84756406 0.57735490 0.69983383 0.70083721 0.17057001
##        C:I       C:II      C:III        D:I       D:II      D:III
## 0.40503490 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
```

```r
#what is Bartlett test?
t = bartlett.test( 1/rats$time, rats$pois )
t
##
##  Bartlett test of homogeneity of variances
##
## data:  1/rats$time and rats$pois
## Bartlett's K-squared = 3.1163, df = 2, p-value = 0.2105
names( t )
## [1] "statistic" "parameter" "p.value"   "data.name" "method"
# t$statistic Bartlett's K-squared test statistic.
# t$parameter the degrees of freedom of the approximate chi-squared
# distribution of the test statistic.
# t$p.value the p-value of the test.
# t$method the character string "Bartlett test of homogeneity of variances".
# t$data.name a character string giving the names of the data.
```

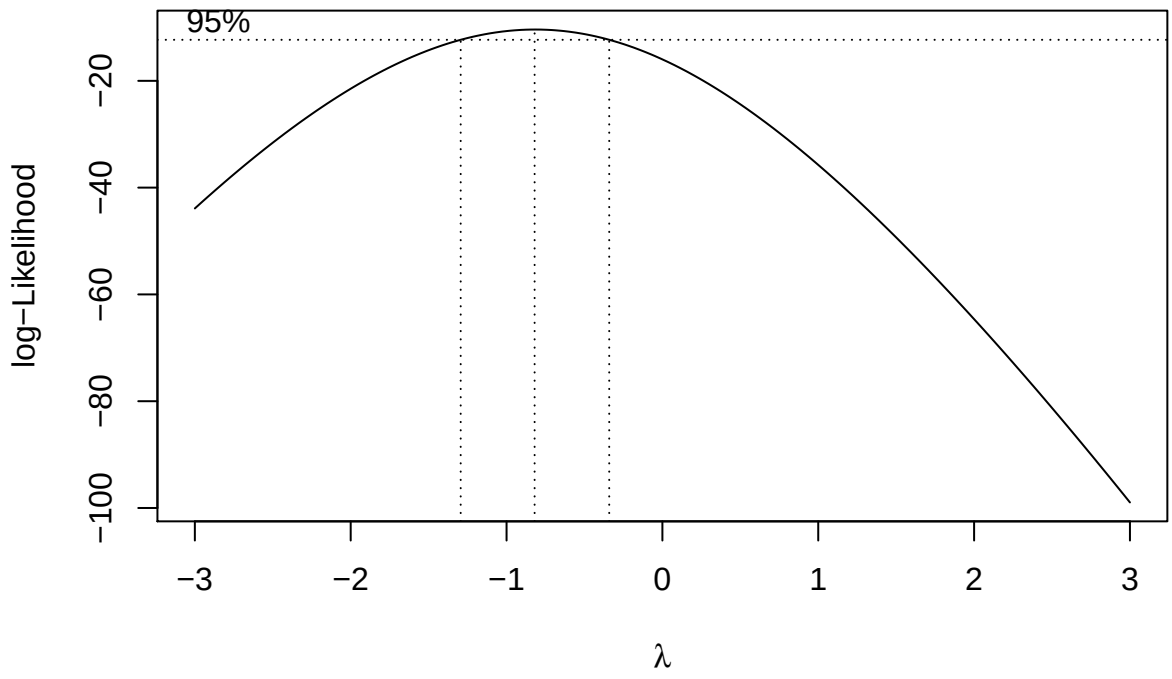
We notice that normality is respected in all 12 groups (even thogh we observe low p-value for the first group A-I). The variances homogeneity is violated (see p-value of Levene's test).

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

Fitting the complete model means considering **interaction** between considered factors.

```r
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 0.80072 0.02224
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

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



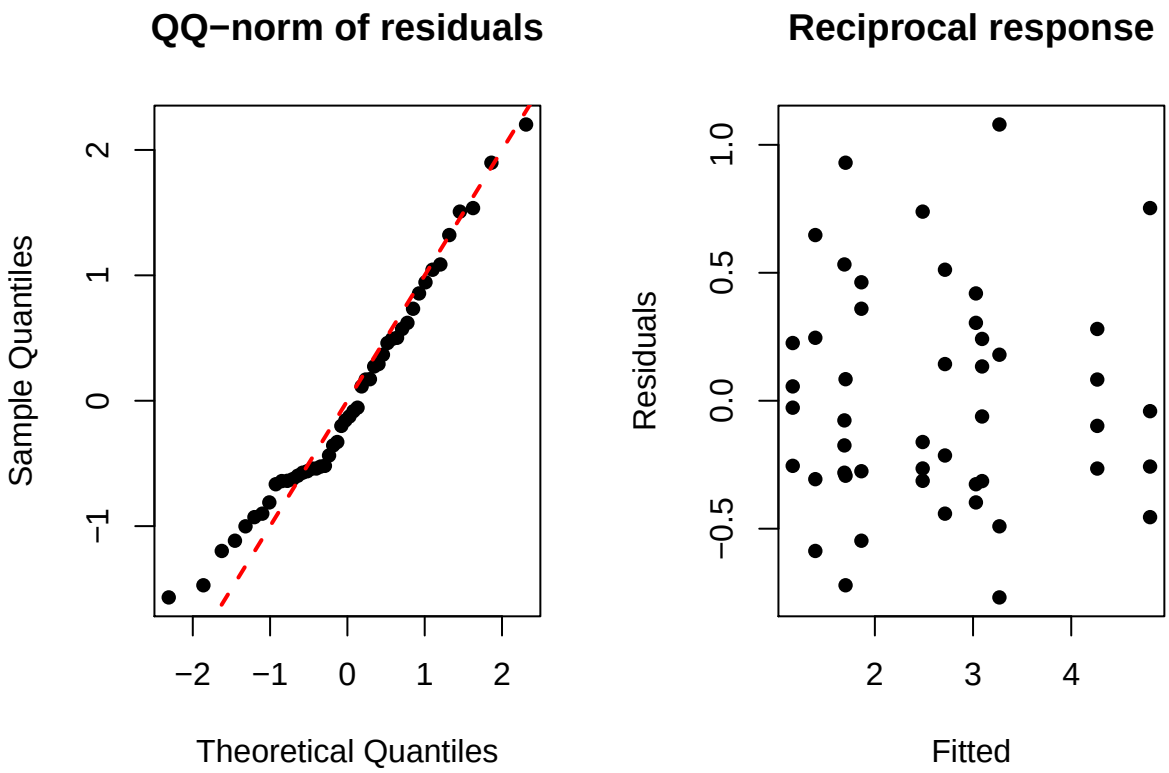

```
best_lambda = b$x[ which.max( b$y ) ]
best_lambda
## [1] -0.82

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 best λ is -1.

We should check the hypotheses of the model.

```
par( mfrow = c( 1, 2 ) )
qqnorm( g1$res/summary( g1 )$sigma, pch = 16, main = 'QQ-norm of residuals'  )
abline( 0, 1, lwd = 2, lty = 2, col = 'red' )

shapiro.test( g1$res )
##
##  Shapiro-Wilk normality test
##
## data:  g1$res
## W = 0.96817, p-value = 0.2148

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



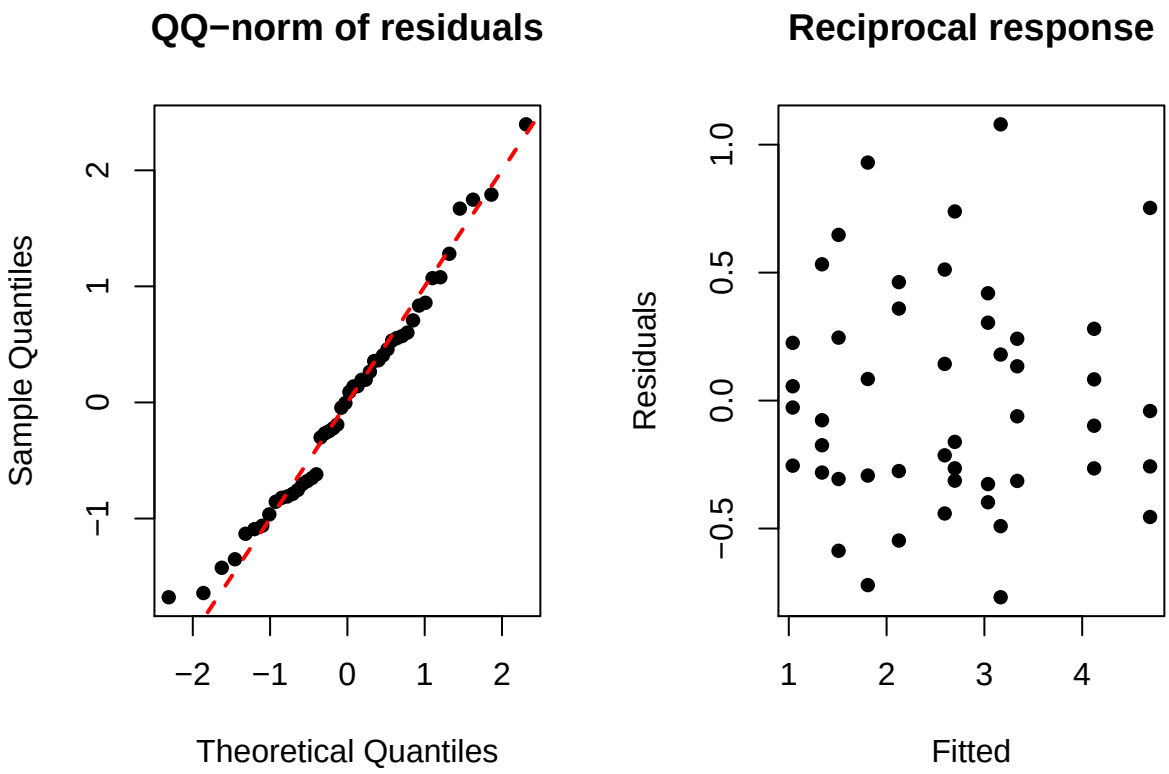
The hypotheses of the model are respected. We should not consider the interaction then:

```
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 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 )
```



This gives us the same output of the following hypotheses investigation:

```
tapply( 1/rats$time, rats$poison, function( x ) shapiro.test( x )$p )
##         I        II       III
## 0.1672488 0.8944364 0.3944087
tapply( 1/rats$time, rats$treat, function( x ) shapiro.test( x )$p )
##         A         B         C         D
## 0.2221106 0.1021497 0.3632241 0.2712347

leveneTest( 1/rats$time, rats$poison )
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  2   1.715 0.1915
##       45
leveneTest( 1/rats$time, rats$treat )
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group  3   0.614 0.6096
##       44

bartlett.test( 1/rats$time, rats$poison )
##
##  Bartlett test of homogeneity of variances
##
## data:  1/rats$time and rats$poison
## Bartlett's K-squared = 3.1163, df = 2, p-value = 0.2105
bartlett.test( 1/rats$time, rats$treat )
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  1/rats$time and rats$treat
## Bartlett's K-squared = 1.5477, df = 3, p-value = 0.6713
```