for multiple regression

https://andyigg.com/spa3-1/

Software

- R: http://cran.r-project.org/
- RStudio: https://www.rstudio.com/products/rstudio/download/#download

Assessment of residuals

- Outliers
- Influential observations
- Normality of residuals
- Equal variance of residuals
- Independence of residuals

Multiple Linear Regression Example

```
set.seed(108) ; n= 100 ; beta0 = array( c(3,2,1,1), c(1,4)  )

x1=rgamma( n, 1, 1/10 ) ; x2=rchisq( n, df = 3 )  ;  x3=rexp( n )
X= cbind( x1, x2, x3 )
y= beta0[,1] + beta0[,2]*x1 + beta0[,3]*x2 + beta0[,4]*x1*x2 +
rnorm( n,sd=1.5 )
```

- summary(mlm)

```
Call:
lm(formula = y ~ x1 + x2 + x3 + x1:x2 + x1:x3)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.7979 -1.0429 -0.0360  0.9585  3.8980 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)  2.770501   0.392648   7.056 2.89e-10 ***
x1           2.044134   0.031253  65.406  < 2e-16 ***
x2           1.075617   0.092715  11.601  < 2e-16 ***
x3          -0.225363   0.229452  -0.982    0.329    
x1:x2        0.991471   0.008786 112.846  < 2e-16 ***
x1:x3       -0.003571   0.024526  -0.146    0.885    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.52 on 94 degrees of freedom
Multiple R-squared:  0.9989,    Adjusted R-squared:  0.9989 
F-statistic: 1.731e+04 on 5 and 94 DF,  p-value: < 2.2e-16
```

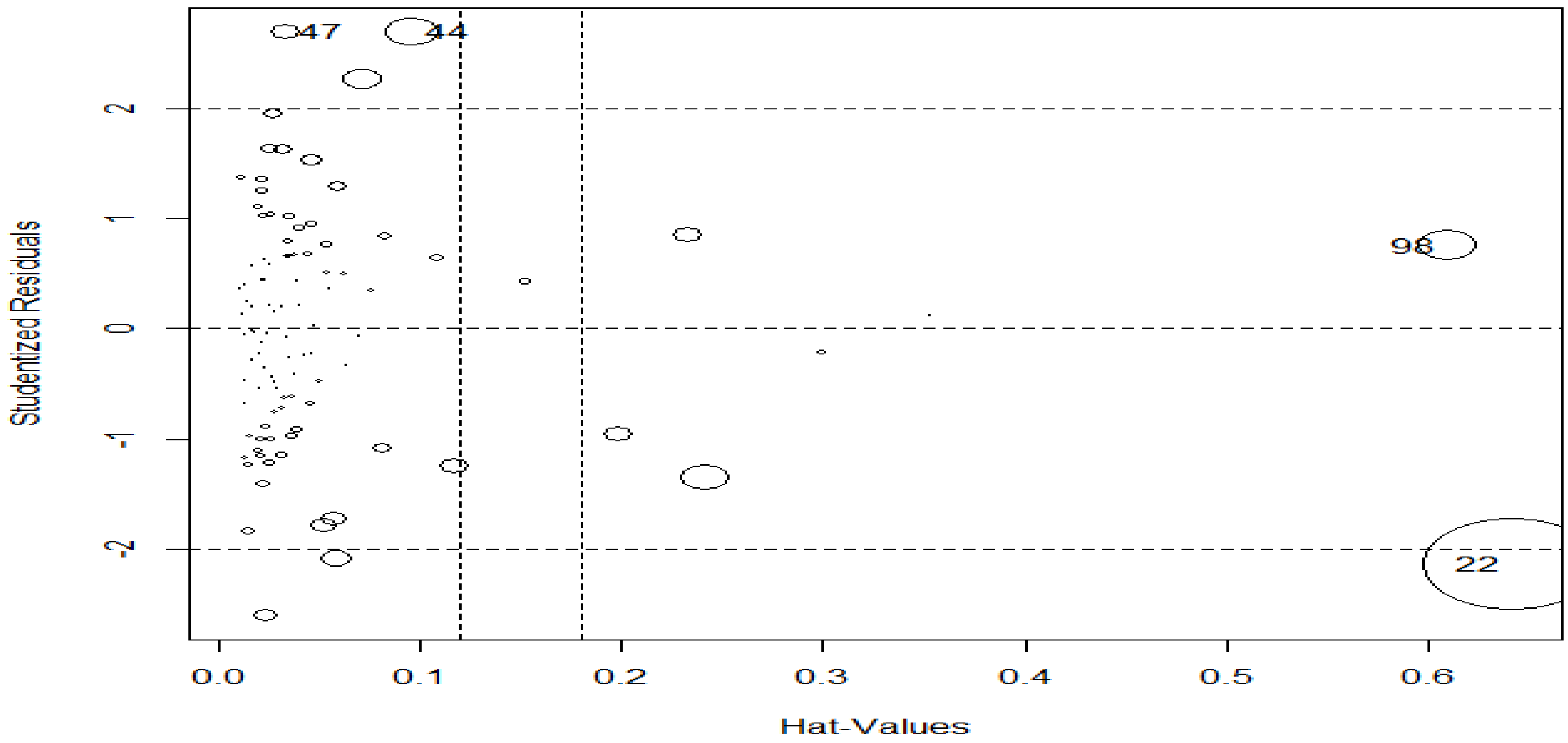
- confint(mlm, level=0.95)

```
> confint(mlm, level=0.95)
                 2.5 %     97.5 %
(Intercept)  1.9908882 3.55011356
x1           1.9820801 2.10618759
x2           0.8915291 1.25970485
x3          -0.6809452 0.23021915
x1:x2        0.9740257 1.00891550
x1:x3       -0.0522689 0.04512595
```

- anova(mlm)

```
> anova(mlm)
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq    F value Pr(>F)    
x1         1 132044  132044 57176.0092 <2e-16 ***
x2         1  36281   36281 15709.9284 <2e-16 ***
x3         1      0       0     0.1101 0.7408    
x1:x2      1  31555   31555 13663.6479 <2e-16 ***
x1:x3      1      0       0     0.0212 0.8845    
Residuals 94    217       2                      
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> 
```

Outliers?



- #install.packages("car") ; library(car)

influencePlot(mlm)

Assessing outliers

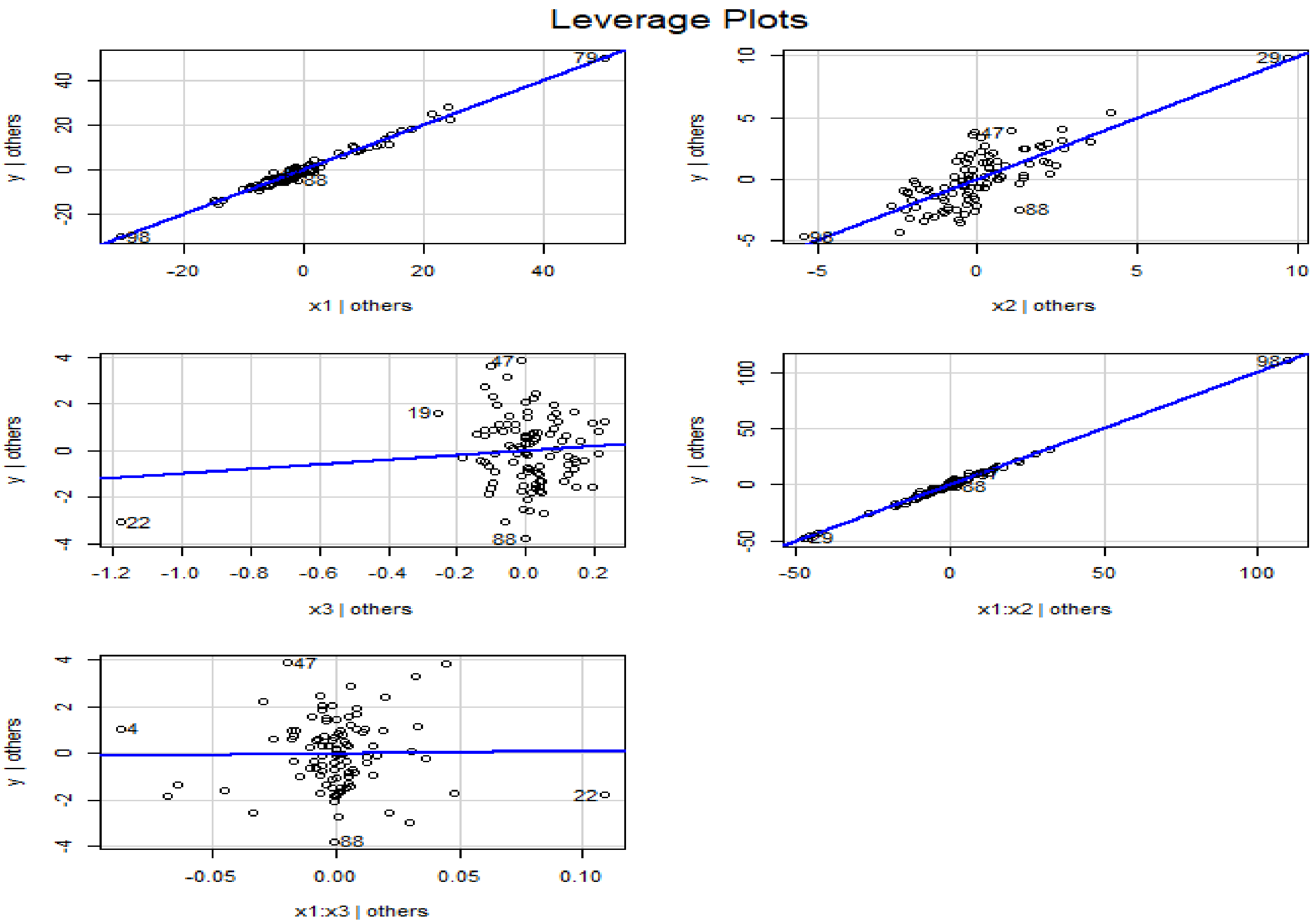
- outlierTest(mlm) # Bonferonni p-value

```
> outlierTest(mlm)
No Studentized residuals with Bonferroni p < 0.05
Largest |rstudent|:
   rstudent unadjusted p-value Bonferroni p
44 2.696387          0.0083212      0.83212
```

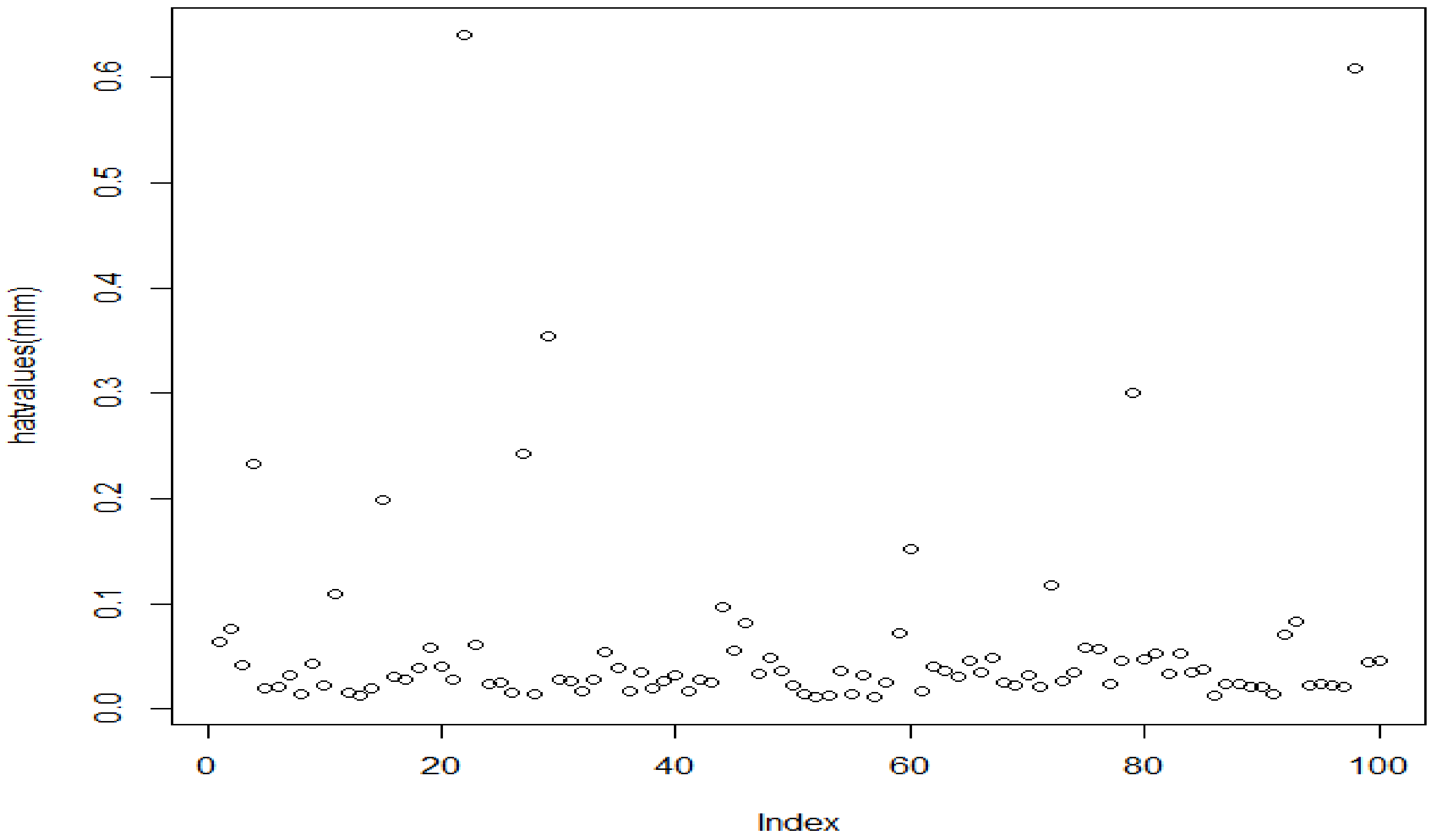
Now, Bonferroni: $1\text{-}\dfrac{\alpha}{k} = \alpha'$, where k is the number of test

e.g. $1\text{-}(1\text{-pnorm}(-2.696387))/6 = 1\text{-pnorm}(2.696387)/6 \approx 0.832$

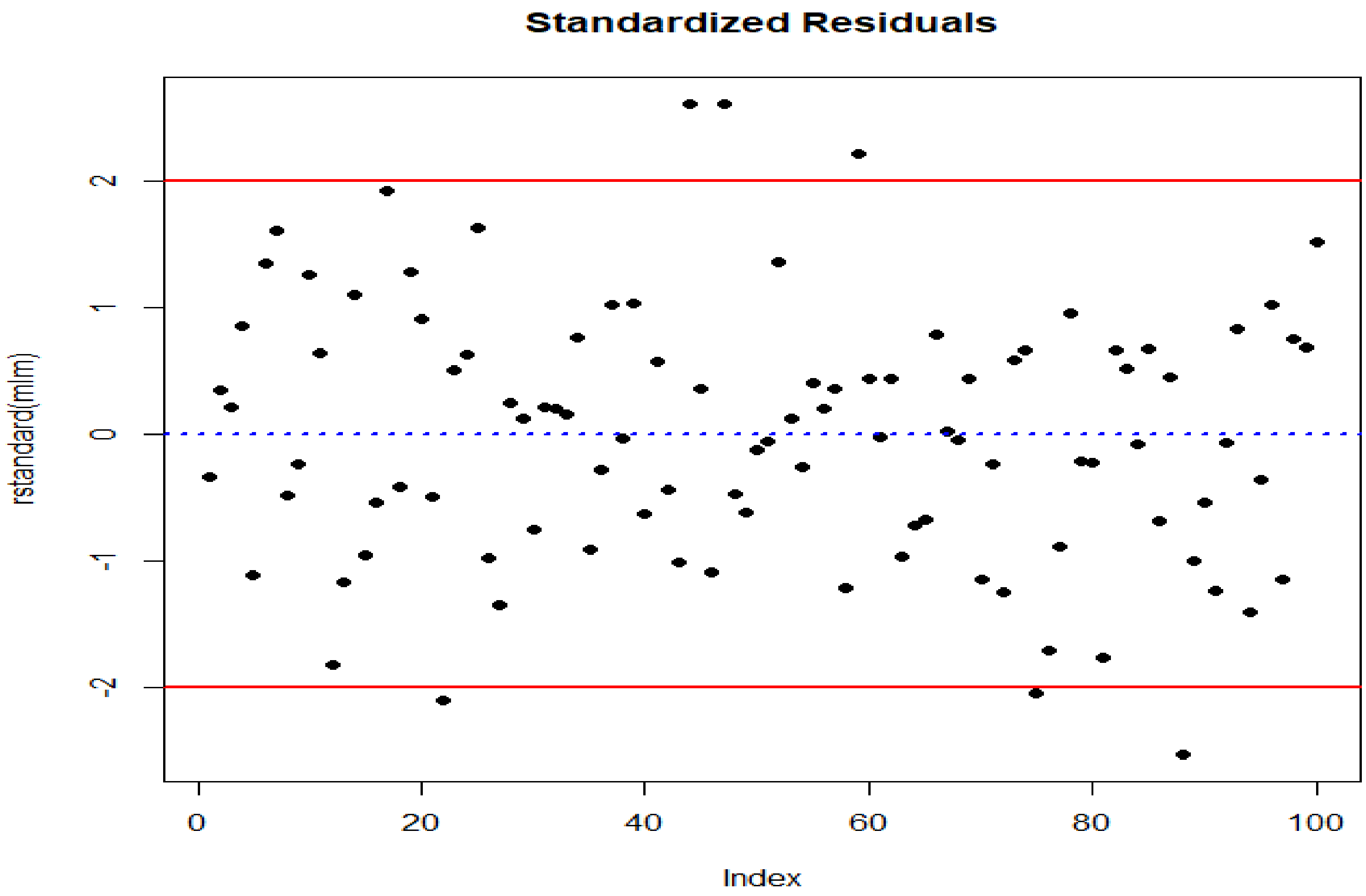
leveragePlots(mlm) # leverage plots



plot(hatvalues(mlm)) # leverage plots



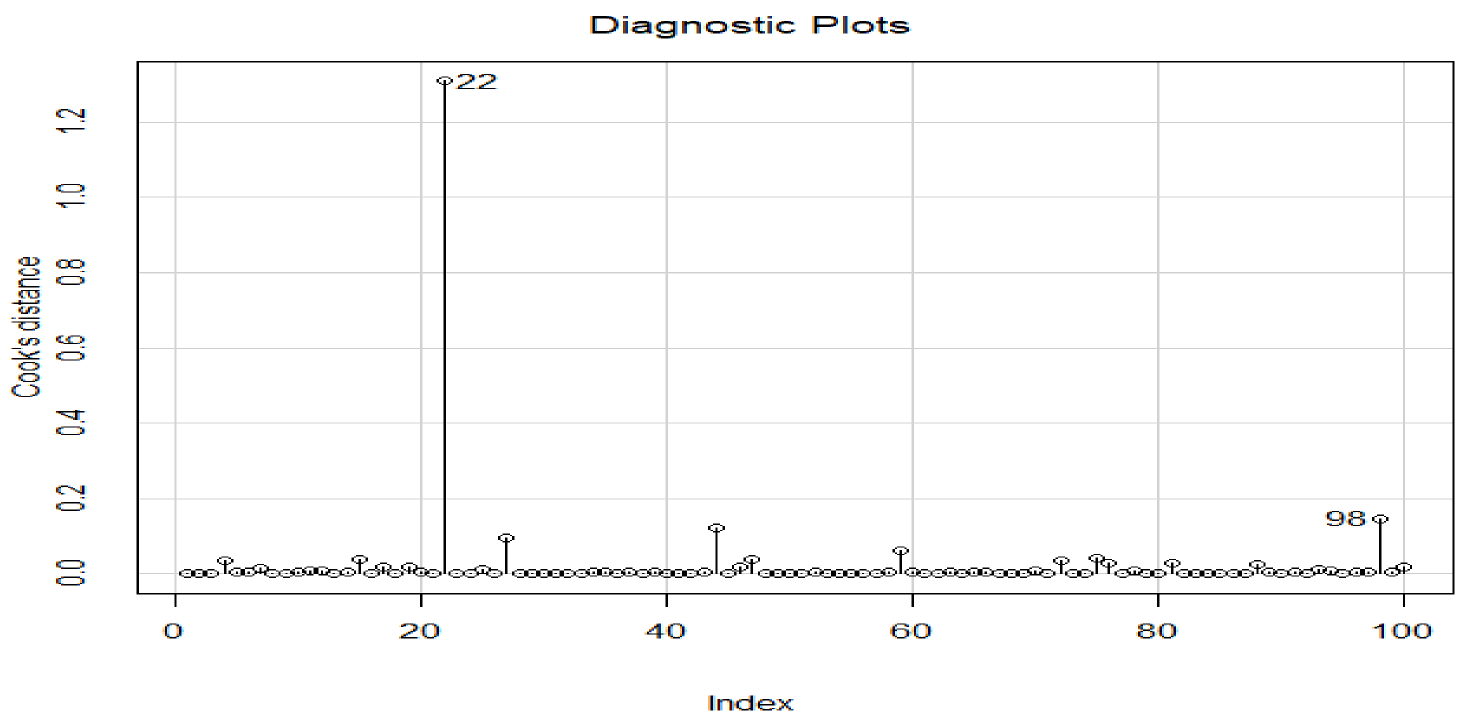
Standardized residuals plot



```
plot(rstandard(mlm), type = "p",main =
"Standardized Residuals")
points( 1:n, rstandard(mlm), pch = 19)

abline( a =-2,b=0, lty=1, lwd=2, col = "red" )
abline( a = 2,b=0, lty=1, lwd=2, col = "red" )
abline( a = 0,b=0, lty=3, lwd=2, col = "blue" )
```

Influential observations: cook’s distance

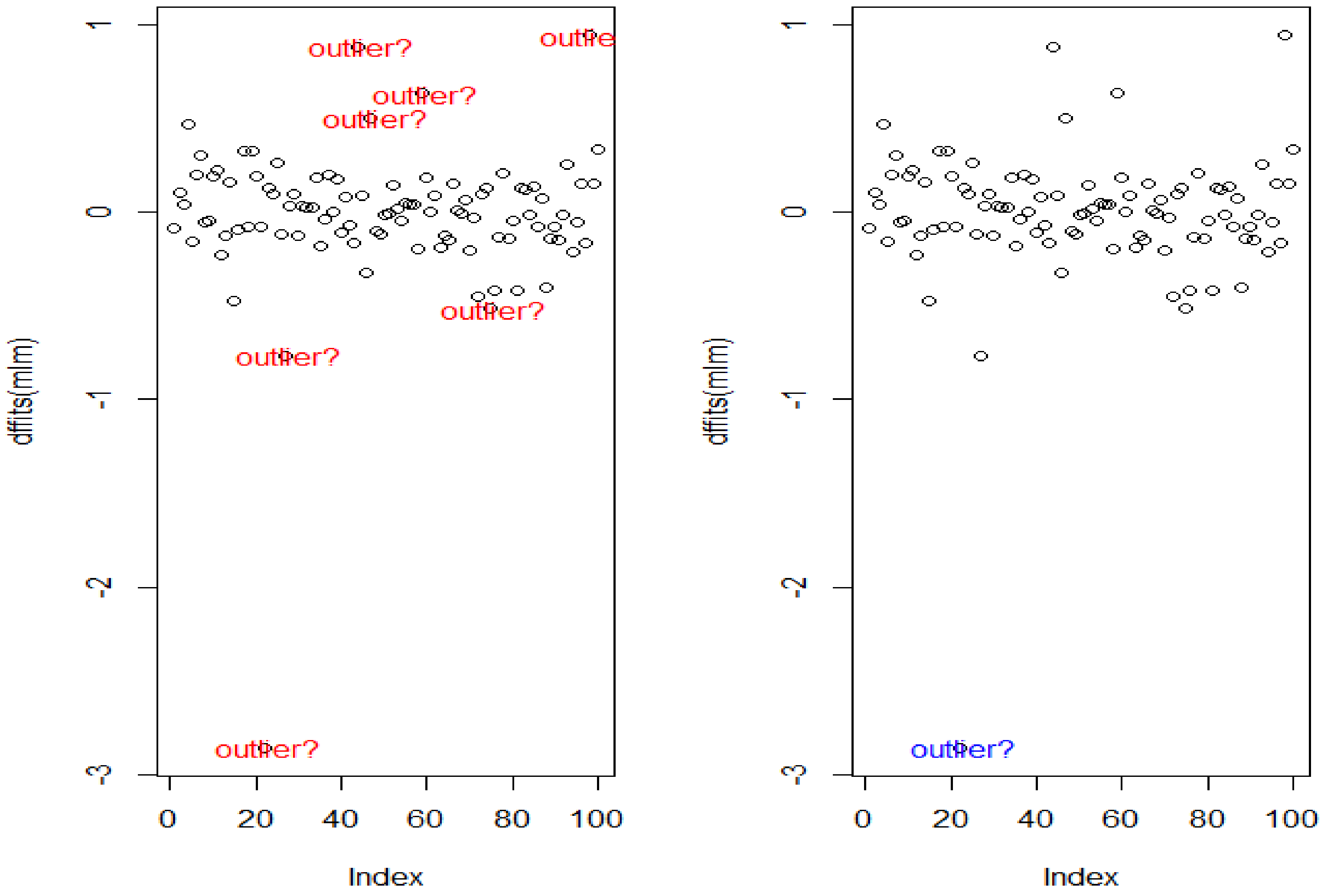
```
# Cook's D plot
# Identify D values > 4/(n-p-1)
cooks=cooks.distance(mlm)
cooks_mlm=cooks[cooks > 4/( n - length(mlm$coefficients ) )]
cooks_mlm

influenceIndexPlot(mlm, vars="Cook", id.n=length(cooks_out))
```



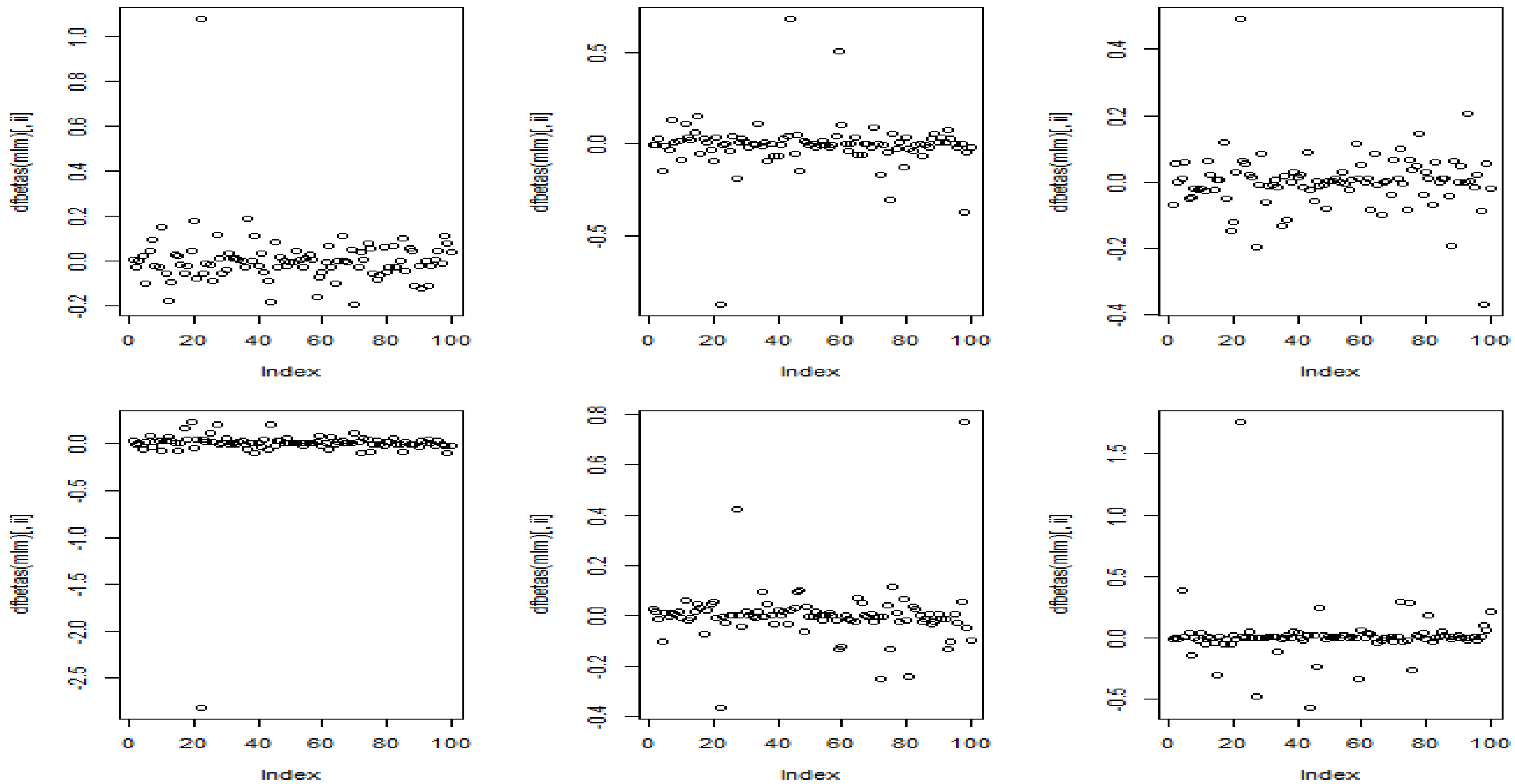

```
> cooks_mlm=cooks[cooks > 4/( n - length(mlm$coefficients ) )]
> cooks_mlm
        22         27         44         59         75         98
1.30878691 0.09684502 0.12119111 0.06350814 0.04309227 0.14736014
>
```

Influential observations: dffits

```
# dffits
par( mfrow=c(1,2), oma = c( 0, 0, 2, 0) ) ; par(mar=c(4,4,2,2))
plot(dffits(mlm)) ; # now, p=5
# dffits > sqrt((p+1)/n)
text( x=1:length(dffits(mlm))+1, y=dffits(mlm),
labels=ifelse(abs(dffits(mlm))>2*sqrt((5+1)/n), "outlier?",""),
col="red" )
# dffits > 1)
plot(dffits(mlm)) ; # now, p=5
text( x=1:length(dffits(mlm))+1, y=dffits(mlm),
labels=ifelse(abs(dffits(mlm))>1, "outlier?",""), col="blue" )
```


dffits(mlm)
Index
outlier?
outlier?
outlier?
outlier?
outlier?
outlier?
outlie

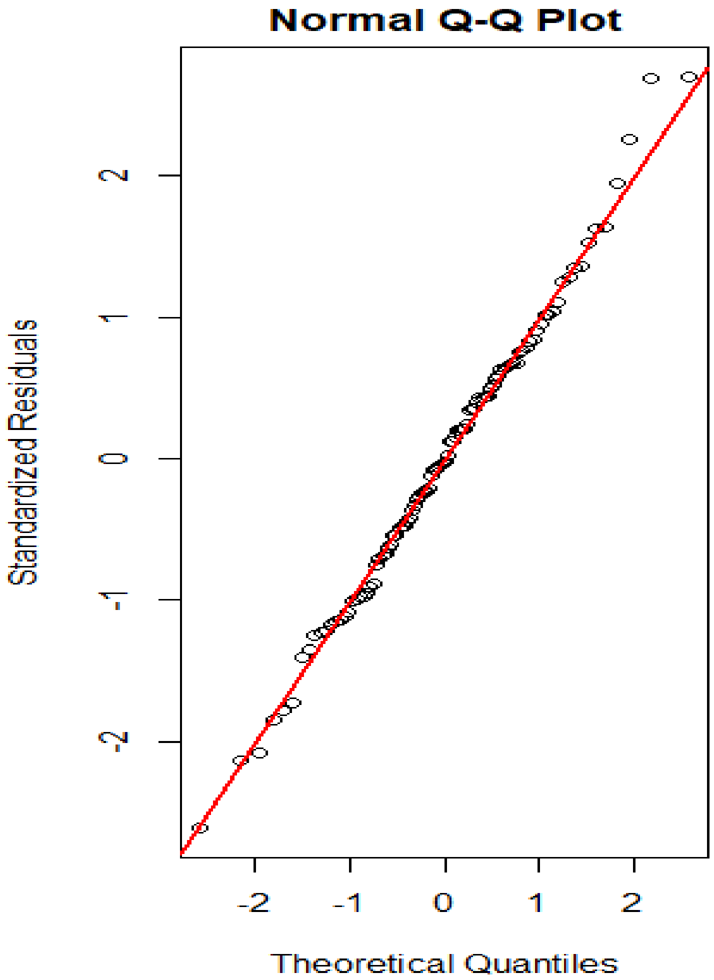
Influential observations: dfbetas

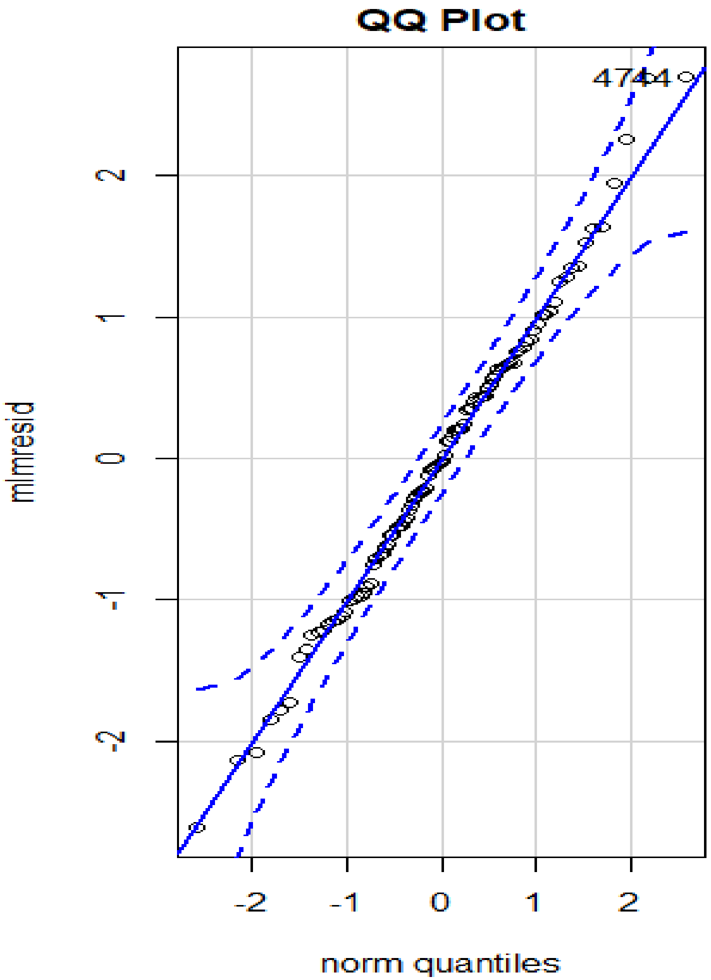


par(mfrow=c(2,3), oma = c(0, 0, 2, 0)) ; par(mar=c(4,4,2,2))

for(ii in 1:length(mlm$coefficients)){ plot(dfbetas(mlm)[,ii]) }

Normality of residuals





```
# library(MASS)
mlmresid=studres(mlm)
par( mfrow=c(1,2), oma = c( 0, 0, 2, 0) ) ;
par(mar=c(4,4,2,2))
qqnorm(mlmresid, ylab ="Standardized Residuals")
qqline(mlmresid, lty=1, lwd=2,col="red" )

qqPlot(mlmresid, main="QQ Plot") # QQ plot for
studentized residual
```

Formal Tests for normality

- **shapiro.test**(mlmresid)
- **ks.test**(mlmresid, "pnorm", mean(mlmresid), sd(mlmresid))
- #require(nortest)

  lillie.test(mlmresid)
- ...

```
> shapiro.test(mlmresid) # Shapiro-Wilks normality test

        Shapiro-Wilk normality test

data:  mlmresid
W = 0.99349, p-value = 0.9155

> ks.test(mlmresid, "pnorm", mean(mlmresid), sd(mlmresid) ) # Kolmogorov-Smirnov normality test

        One-sample Kolmogorov-Smirnov test

data:  mlmresid
D = 0.035714, p-value = 0.9996
alternative hypothesis: two-sided

> require(nortest)
> lillie.test(mlmresid) # Lilliefors (Kolomorov-Smirnov) normality test

        Lilliefors (Kolmogorov-Smirnov) normality test

data:  mlmresid
D = 0.035714, p-value = 0.9895
```

Which Normality Test Should I Use?

- **Kolmogorov-Smirnov test**
  It is more sensitive near the center of the density than at the tails than other tests; large n
- **Shapiro-Wilks test**
  Doesn't work well if several values in the data set are the same. Works best for data sets with small n, but can be used with larger data sets.
- **Lillie test** (adjusted Kolmogorov-Smirnov test)
  More powerful than Kolmogorov-Smirnov test

http://www.hmwu.idv.tw/web/R_AI/v2/hmwu_StatR-05-2_NonParametric_adv.pdf
https://www.nrc.gov/docs/ML1714/ML17143A100.pdf

Parametric Test (normality)

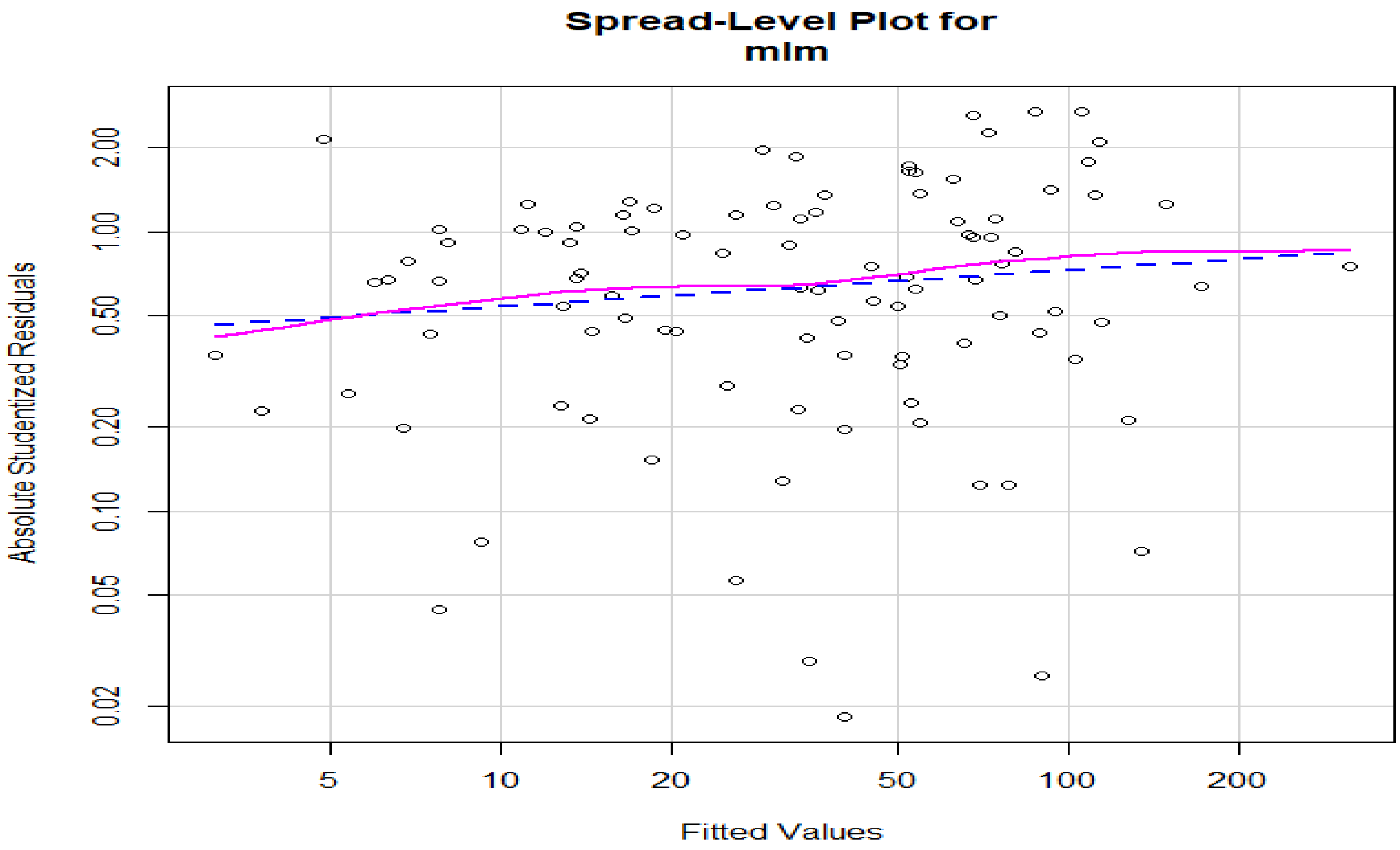
- T-test
- Anova
- Pearson correlation test
- …

Which Normality Test Should I Use?

- It is preferable that normality be assessed both visually and through normality tests, of which the **Shapiro-Wilk test** is recommended.

- The KS or Lillie test should be "carefully" used owing to its low power.

Evaluate homoscedasticity



spreadLevelPlot(mlm)

Test for homoscedasticity

- **ncvTest**(mlm) # Non-constant Variance Score Test
- Computes a score test of the hypothesis of constant error variance against the alternative that the error variance changes with the level of the response (fitted values), or with a linear combination of predictors.

```
> ncvTest(mlm) # non-constant error variance test
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 2.617584, Df = 1, p = 0.10569
```

Introduce some test homogeneity of variances

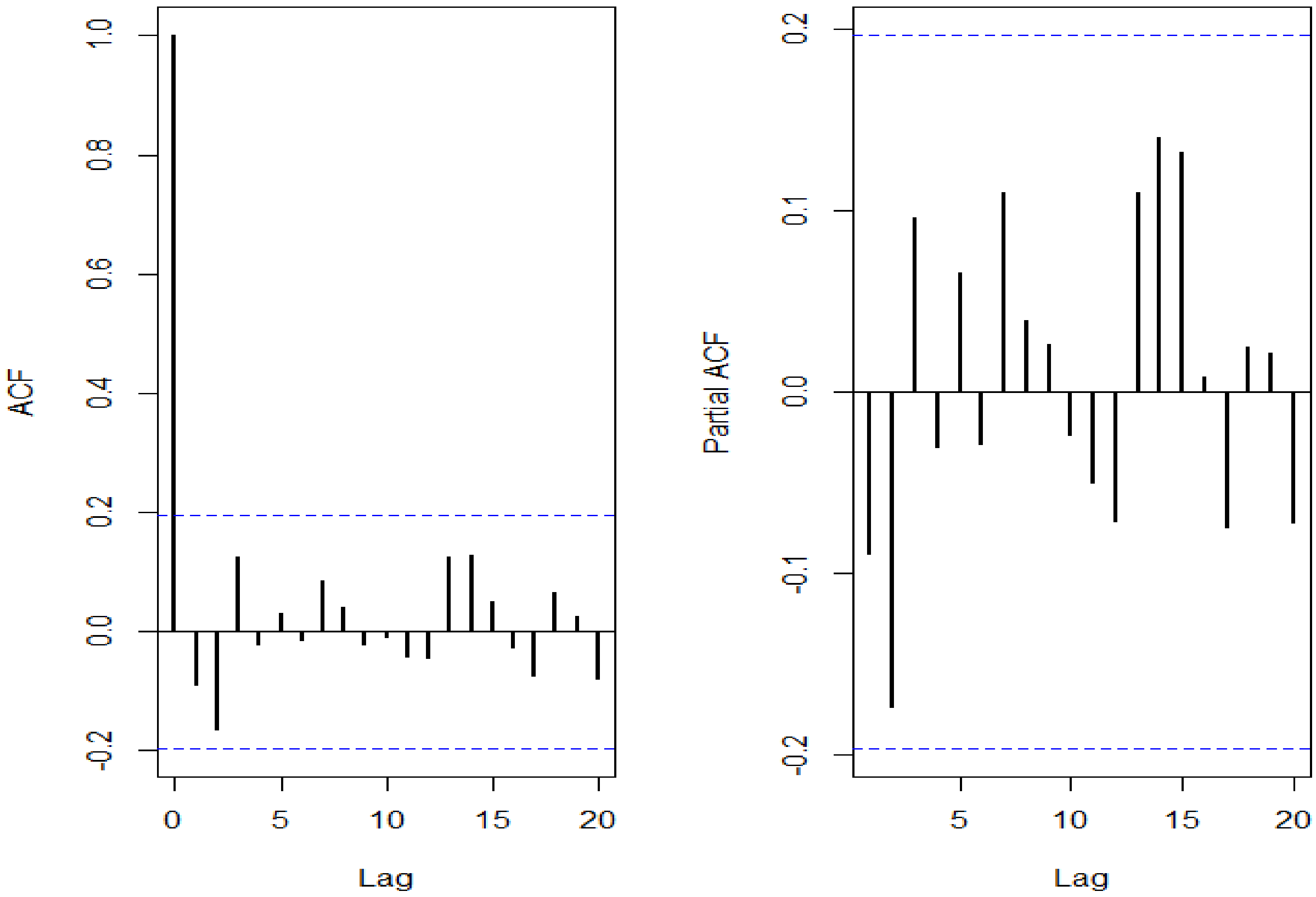
- **leveneTest** # package{car}
- **fligner.test**: (nonparametric) k-sample test
- **mood.test**: two-sample test for a difference in scale parameters
- **ansari.test**: two-sample test for a difference in scale parameters. (testing for equal variance for non-normal samples)
- **bartlett.test**: a parametric test of the null that the variances in each of the groups (samples) are the same.
- **var.test**: an F test to compare the variances of two samples from normal populations.
- …

Test for independence

- durbinWatsonTest(mlm) # test for autocorrelated errors

```
> durbinWatsonTest(mlm) # test for autocorrelated errors
 lag Autocorrelation D-W Statistic p-value
   1     -0.09174254      2.158922   0.462
 Alternative hypothesis: rho != 0
>
> |
```

Residuals: dependence diagnosis



Multi-collinearity: variance inflation factors

- vif >10: multi-collinearity

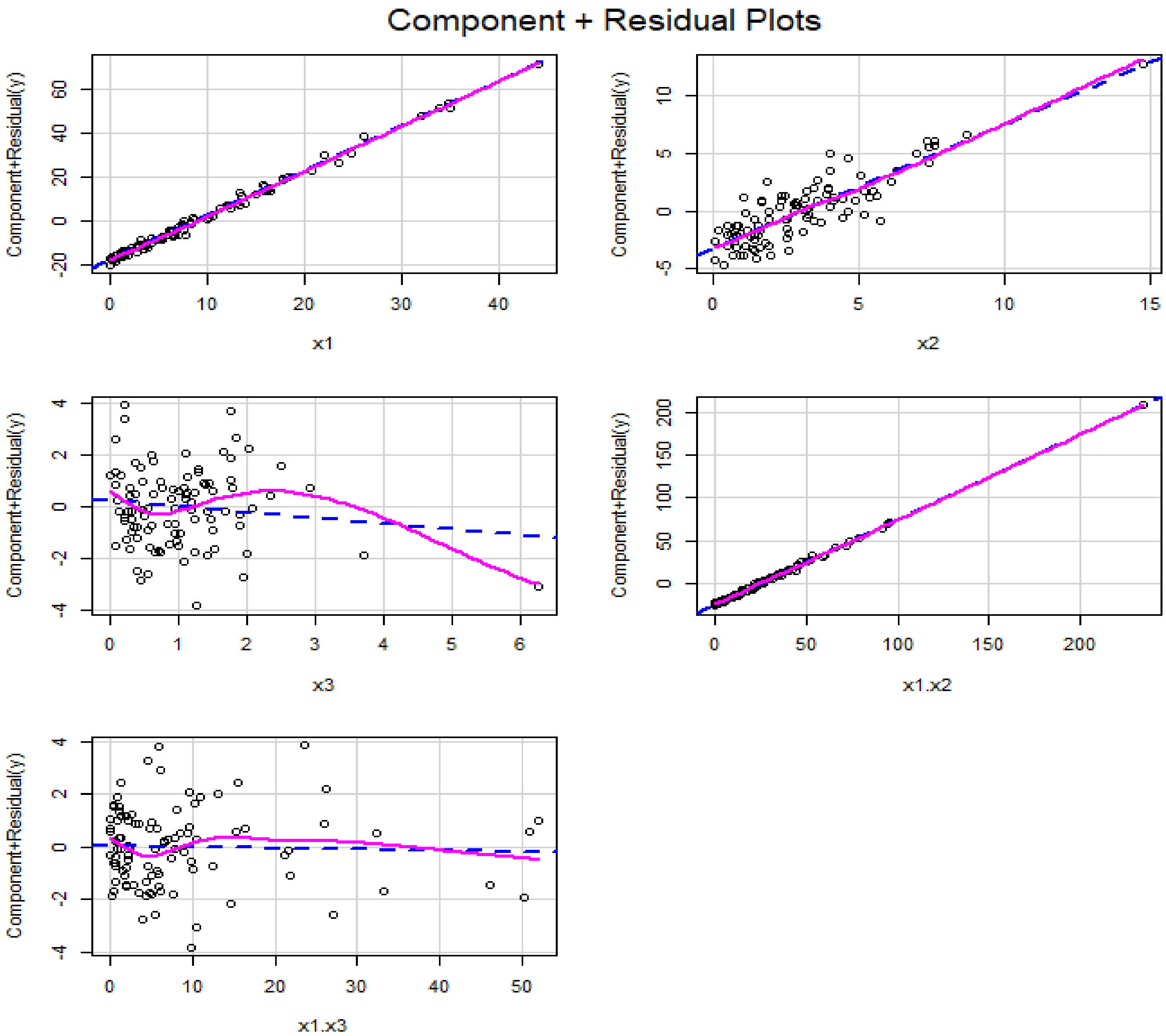
- Cohen, J, Cohen, P, West, S. G., and Aiken, L. S. 2003. Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences (3rd ed.). Hillsdale, NJ: Lawrence Erlbaum.
- Kutner, Nachtsheim, and Neter, 2004, Applied Linear Regression Models, Fourth Edition, McGraw-Hill Irwin.)

```
> vif(mlm)
      x1       x2       x3    x1:x2    x1:x3
3.296571 1.971189 1.703740 3.188911 3.233037
>
> set.seed(108) ; n= 100 ; beta0 = array( c(3, 2, 1, 1), c(1,4)  )
>
> x1=rgamma( n, 1, 1/10 ) ; x2= 1.5*x1 + rnorm( n, sd= 0.5 )  ;  x3=rexp( n )
> X= cbind( x1, x2, x3 )
> y= beta0[,1] + beta0[,2]*x1 + beta0[,3]*x2 + beta0[,4]*x1*x2 + rnorm( n, sd=1.5 )
>
> mlm2 = lm(y ~ x1 + x2 + x3 + x1:x2 + x1:x3 )
> vif(mlm2)
        x1         x2         x3      x1:x2      x1:x3
732.252036 753.418106   2.266356   7.620066   3.559061
>
```

(Non-) Linearity

- These functions construct component+residual plots, also called partial-residual plots, for linear and generalized linear models.
- df = data.frame(y,x1,x2,x3,x1.x2=x1*x2, x1.x3=x1*x3)

  lm.fit1 = lm(y ~ ., df) ; crPlots(lm.fit1)


Component + Residual Plots
Component+Residual(y)
x1
x2
x3
x1.x2
x1.x3

Global Validation of Linear Model Assumptions

- Pena, EA and Slate, EH (2006). Global validation of linear model assumptions, *J. Am. Stat. Assoc.*, **101**(473):341-354.
- library(gvlma)

  summary(gvlma(mlm))

```
Call:
lm(formula = y ~ x1 + x2 + x3 + x1:x2 + x1:x3)

Residuals:
    Min      1Q  Median      3Q     Max
-3.7979 -1.0429 -0.0360  0.9585  3.8980

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.770501   0.392648   7.056 2.89e-10 ***
x1           2.044134   0.031253  65.406  < 2e-16 ***
x2           1.075617   0.092715  11.601  < 2e-16 ***
x3          -0.225363   0.229452  -0.982    0.329
x1:x2        0.991471   0.008786 112.846  < 2e-16 ***
x1:x3       -0.003571   0.024526  -0.146    0.885
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.52 on 94 degrees of freedom
Multiple R-squared:  0.9989,    Adjusted R-squared:  0.9989
F-statistic: 1.731e+04 on 5 and 94 DF,  p-value: < 2.2e-16


ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
Level of Significance =  0.05

Call:
 gvlma(x = mlm)

                    Value p-value                Decision
Global Stat        0.90519  0.9238 Assumptions acceptable.
Skewness           0.37302  0.5414 Assumptions acceptable.
Kurtosis           0.00225  0.9622 Assumptions acceptable.
Link Function      0.49035  0.4838 Assumptions acceptable.
Heteroscedasticity 0.03957  0.8423 Assumptions acceptable.
```

Comparing models: ANOVA

```
> mlm_mainterms = lm( y~x1+x2+x3)
> anova(mlm, mlm_mainterms )
Analysis of Variance Table

Model 1: y ~ x1 + x2 + x3 + x1:x2 + x1:x3
Model 2: y ~ x1 + x2 + x3
  Res.Df   RSS Df Sum of Sq      F    Pr(>F)
1     94   217
2     96 31772 -2    -31555 6831.8 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> mlm = lm(y ~ x1 + x2 + x3 + x1:x2 + x1:x3 )
>
> mlm_correct = lm( y~x1+x2+x1:x2)
>
> anova(mlm, mlm_correct )
Analysis of Variance Table

Model 1: y ~ x1 + x2 + x3 + x1:x2 + x1:x3
Model 2: y ~ x1 + x2 + x1:x2
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     94 217.09
2     96 221.49 -2   -4.4042 0.9535 0.3891
>
```

Model selection

```
require(MASS)
mlm_all=lm(y~x1+x2+x3+x1:x2+x1:x3+x2:x3 )
data.input = data.frame( y, x1, x2, x3, x1*x2, x1*x3,
x2*x3 )

fit.full = lm(y ~ ., data.input) ; fit.null = lm(y ~ 1,
data.input)
```

```
Start:  AIC=762.14
y ~ 1

           Df Sum of Sq    RSS    AIC
+ x1...x2   1    182385  17712 521.68
+ x1        1    132044  68054 656.29
+ x1...x3   1     86219 113879 707.77
+ x2        1     21689 178408 752.67
+ x2...x3   1      9107 190990 759.48
<none>                  200097 762.14
+ x3        1        27 200070 764.13

Step:  AIC=521.68
y ~ x1...x2

           Df Sum of Sq     RSS    AIC
+ x1        1   17176.8   535.4 173.79
+ x1...x3   1    4219.1 13493.1 496.48
+ x2        1    3624.1 14088.1 500.79
+ x2...x3   1    1664.1 16048.2 513.82
<none>                  17712.2 521.68
+ x3        1     336.2 17376.0 521.77

Step:  AIC=173.79
y ~ x1...x2 + x1

           Df Sum of Sq    RSS     AIC
+ x2        1   313.921 221.49  87.521
+ x2...x3   1   128.132 407.28 148.433
<none>                  535.41 173.786
+ x1...x3   1     4.181 531.23 175.003
+ x3        1     0.032 535.38 175.780

Step:  AIC=87.52
y ~ x1...x2 + x1 + x2

           Df Sum of Sq    RSS    AIC
<none>                  221.49 87.521
+ x3        1    4.3552 217.13 87.535
+ x1...x3   1    2.1763 219.31 88.533
+ x2...x3   1    0.4981 220.99 89.296

Call:
lm(formula = y ~ x1...x2 + x1 + x2, data = data.input)

Coefficients:
(Intercept)      x1...x2           x1           x2
      2.571        0.991        2.043        1.062
```

stepAIC(fit.null,direction="forward",scope=list(upper=fit.full,lower=fit.null)) # lm(formula = y ~ x1...x2 + x1 + x2, data = data.input)

```
Start:  AIC=89.28
y ~ x1 + x2 + x3 + x1...x2 + x1...x3 + x2...x3

           Df Sum of Sq     RSS    AIC
- x1...x3   1       0.0   212.3  87.28
<none>                    212.3  89.28
- x2...x3   1       4.8   217.1  89.51
- x3        1       7.0   219.3  90.53
- x2        1      60.3   272.6 112.29
- x1        1    9404.1  9616.4 468.61
- x1...x2   1   28938.4 29150.7 579.51

Step:  AIC=87.28
y ~ x1 + x2 + x3 + x1...x2 + x2...x3

           Df Sum of Sq     RSS    AIC
<none>                    212.3  87.28
- x2...x3   1       4.8   217.1  87.53
- x3        1       8.7   221.0  89.30
- x2        1      60.4   272.7 110.31
- x1        1   13364.8 13577.1 501.10
- x1...x2   1   30734.8 30947.1 583.49

Call:
lm(formula = y ~ x1 + x2 + x3 + x1...x2 + x2...x3, data = data.input)

Coefficients:
(Intercept)           x1           x2           x3      x1...x2
     3.3680       2.0354       0.8701      -0.6948       0.9933
    x2...x3
     0.1460
```

stepAIC(fit.full,direction="backward")

```
Start:  AIC=762.14
y ~ 1

           Df Sum of Sq    RSS    AIC
+ x1...x2   1    182385  17712 521.68
+ x1        1    132044  68054 656.29
+ x1...x3   1     86219 113879 707.77
+ x2        1     21689 178408 752.67
+ x2...x3   1      9107 190990 759.48
<none>                  200097 762.14
+ x3        1        27 200070 764.13

Step:  AIC=521.68
y ~ x1...x2

           Df Sum of Sq    RSS    AIC
+ x1        1     17177    535 173.79
+ x1...x3   1      4219  13493 496.48
+ x2        1      3624  14088 500.79
+ x2...x3   1      1664  16048 513.82
<none>                   17712 521.68
+ x3        1       336  17376 521.77
- x1...x2   1    182385 200097 762.14

Step:  AIC=173.79
y ~ x1...x2 + x1

           Df Sum of Sq   RSS    AIC
+ x2        1       314   221  87.52
+ x2...x3   1       128   407 148.43
<none>                    535 173.79
+ x1...x3   1         4   531 175.00
+ x3        1         0   535 175.78
- x1        1     17177 17712 521.68
- x1...x2   1     67518 68054 656.29

Step:  AIC=87.52
y ~ x1...x2 + x1 + x2

           Df Sum of Sq   RSS    AIC
<none>                    221  87.52
+ x3        1       4.4   217  87.53
+ x1...x3   1       2.2   219  88.53
+ x2...x3   1       0.5   221  89.30
- x2        1     313.9   535 173.79
- x1        1   13866.6 14088 500.79
- x1...x2   1   31551.1 31773 582.12

Call:
lm(formula = y ~ x1...x2 + x1 + x2, data = data.input)

Coefficients:
(Intercept)      x1...x2           x1           x2
      2.571        0.991        2.043        1.062
```

stepAIC(fit.null,direction="both" ,scope=list(upper=fit.full,lower=fit.null
)) # lm(formula = y ~ x1...x2 + x1 + x2, data = data.input)

```
Start:  AIC=762.14
y ~ 1

            Df Sum of Sq    RSS    AIC
+ x1...x2    1    182385  17712 521.68
+ x1         1    132044  68054 656.29
+ x1...x3    1     86219 113879 707.77
+ x2         1     21689 178408 752.67
+ x2...x3    1      9107 190990 759.48
<none>                   200097 762.14
+ x3         1        27 200070 764.13

Step:  AIC=521.68
y ~ x1...x2

            Df Sum of Sq    RSS    AIC
+ x1         1     17177    535 173.79
+ x1...x3    1      4219  13493 496.48
+ x2         1      3624  14088 500.79
+ x2...x3    1      1664  16048 513.82
<none>                    17712 521.68
+ x3         1       336  17376 521.77
- x1...x2    1    182385 200097 762.14

Step:  AIC=173.79
y ~ x1...x2 + x1

            Df Sum of Sq   RSS    AIC
+ x2         1       314   221  87.52
+ x2...x3    1       128   407 148.43
<none>                     535 173.79
+ x1...x3    1         4   531 175.00
+ x3         1         0   535 175.78
- x1         1     17177 17712 521.68
- x1...x2    1     67518 68054 656.29

Step:  AIC=87.52
y ~ x1...x2 + x1 + x2

            Df Sum of Sq   RSS    AIC
<none>                     221  87.52
+ x3         1       4.4   217  87.53
+ x1...x3    1       2.2   219  88.53
+ x2...x3    1       0.5   221  89.30
- x2         1     313.9   535 173.79
- x1         1   13866.6 14088 500.79
- x1...x2    1   31551.1 31773 582.12

Call:
lm(formula = y ~ x1...x2 + x1 + x2, data = data.input)

Coefficients:
(Intercept)      x1...x2           x1           x2
      2.571        0.991        2.043        1.062
```

stepAIC(fit.null,direction="forward",scope=list(upper=fit.full,lower=fit.null), **k=log(n))** # lm(formula = y ~ x1...x2 + x1 + x2, data = data.input)

```
Start:  AIC=762.14
y ~ 1

              Df Sum of Sq    RSS    AIC
+ x1...x2    1    182385  17712 521.68
+ x1         1    132044  68054 656.29
+ x1...x3    1     86219 113879 707.77
+ x2         1     21689 178408 752.67
+ x2...x3    1      9107 190990 759.48
<none>                   200097 762.14
+ x3         1        27 200070 764.13

Step:  AIC=521.68
y ~ x1...x2

              Df Sum of Sq    RSS    AIC
+ x1         1     17177    535 173.79
+ x1...x3    1      4219  13493 496.48
+ x2         1      3624  14088 500.79
+ x2...x3    1      1664  16048 513.82
<none>                    17712 521.68
+ x3         1       336  17376 521.77
- x1...x2    1    182385 200097 762.14

Step:  AIC=173.79
y ~ x1...x2 + x1

              Df Sum of Sq   RSS    AIC
+ x2         1       314   221  87.52
+ x2...x3    1       128   407 148.43
<none>                     535 173.79
+ x1...x3    1         4   531 175.00
+ x3         1         0   535 175.78
- x1         1     17177 17712 521.68
- x1...x2    1     67518 68054 656.29

Step:  AIC=87.52
y ~ x1...x2 + x1 + x2

              Df Sum of Sq   RSS    AIC
<none>                     221  87.52
+ x3         1       4.4   217  87.53
+ x1...x3    1       2.2   219  88.53
+ x2...x3    1       0.5   221  89.30
- x2         1     313.9   535 173.79
- x1         1   13866.6 14088 500.79
- x1...x2    1   31551.1 31773 582.12

Call:
lm(formula = y ~ x1...x2 + x1 + x2, data = data.input)

Coefficients:
(Intercept)      x1...x2           x1           x2
      2.571        0.991        2.043        1.062
```

stepAIC(fit.full,direction="backward", **k=log(n)**) #

lm(formula = y ~ x1...x2 + x1 + x2, data = data.input)

```
Start:  AIC=762.14
y ~ 1

            Df Sum of Sq    RSS    AIC
+ x1...x2    1    182385  17712 521.68
+ x1         1    132044  68054 656.29
+ x1...x3    1     86219 113879 707.77
+ x2         1     21689 178408 752.67
+ x2...x3    1      9107 190990 759.48
<none>                   200097 762.14
+ x3         1        27 200070 764.13

Step:  AIC=521.68
y ~ x1...x2

            Df Sum of Sq    RSS    AIC
+ x1         1     17177    535 173.79
+ x1...x3    1      4219  13493 496.48
+ x2         1      3624  14088 500.79
+ x2...x3    1      1664  16048 513.82
<none>                    17712 521.68
+ x3         1       336  17376 521.77
- x1...x2    1    182385 200097 762.14

Step:  AIC=173.79
y ~ x1...x2 + x1

            Df Sum of Sq   RSS    AIC
+ x2         1       314   221  87.52
+ x2...x3    1       128   407 148.43
<none>                     535 173.79
+ x1...x3    1         4   531 175.00
+ x3         1         0   535 175.78
- x1         1     17177 17712 521.68
- x1...x2    1     67518 68054 656.29

Step:  AIC=87.52
y ~ x1...x2 + x1 + x2

            Df Sum of Sq   RSS    AIC
<none>                     221  87.52
+ x3         1       4.4   217  87.53
+ x1...x3    1       2.2   219  88.53
+ x2...x3    1       0.5   221  89.30
- x2         1     313.9   535 173.79
- x1         1   13866.6 14088 500.79
- x1...x2    1   31551.1 31773 582.12

Call:
lm(formula = y ~ x1...x2 + x1 + x2, data = data.input)

Coefficients:
(Intercept)      x1...x2           x1           x2
      2.571        0.991        2.043        1.062
```

stepAIC(fit.null,direction="both" ,scope=list(upper=fit.full,lower=fit.null
), **k=log(n))** # lm(formula = y ~ x1...x2 + x1 + x2, data = data.input)

Select on all subsets

library(leaps)

data.input = data.frame(y, x1, x2, x3, x1*x2, x1*x3, x2*x3)

model.com = regsubsets(y~., nvmax=30, data=data.input)

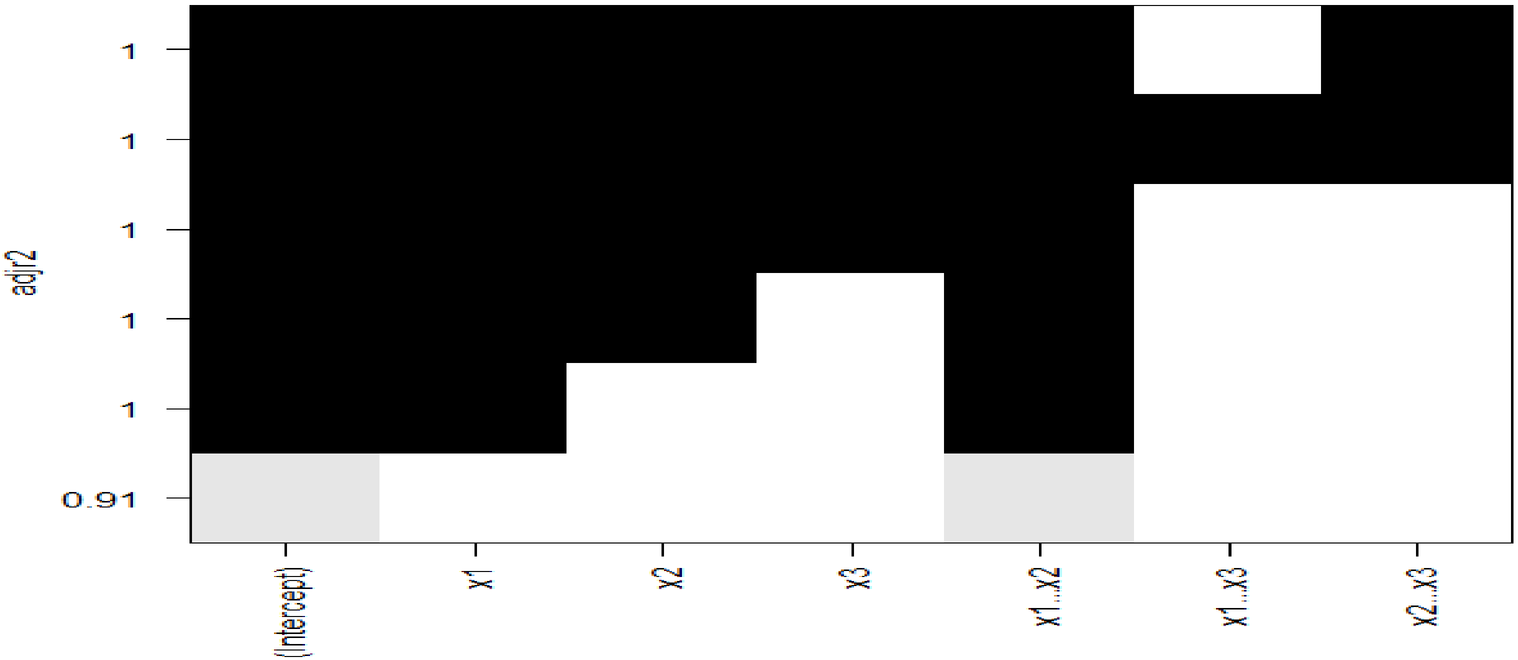
model.sum = summary(model.com)

names(model.sum) # "which" "rsq"   "rss"   "adjr2" "cp"   "bic"   "outmat" "obj"

```
         x1  x2  x3  x1...x2 x1...x3 x2...x3
1  ( 1 ) " " " " " " "*"     " "     " "
2  ( 1 ) "*" " " " " "*"     " "     " "
3  ( 1 ) "*" "*" " " "*"     " "     " "
4  ( 1 ) "*" "*" "*" "*"     " "     " "
5  ( 1 ) "*" "*" "*" "*"     " "     "*"
6  ( 1 ) "*" "*" "*" "*"     "*"     "*"
```

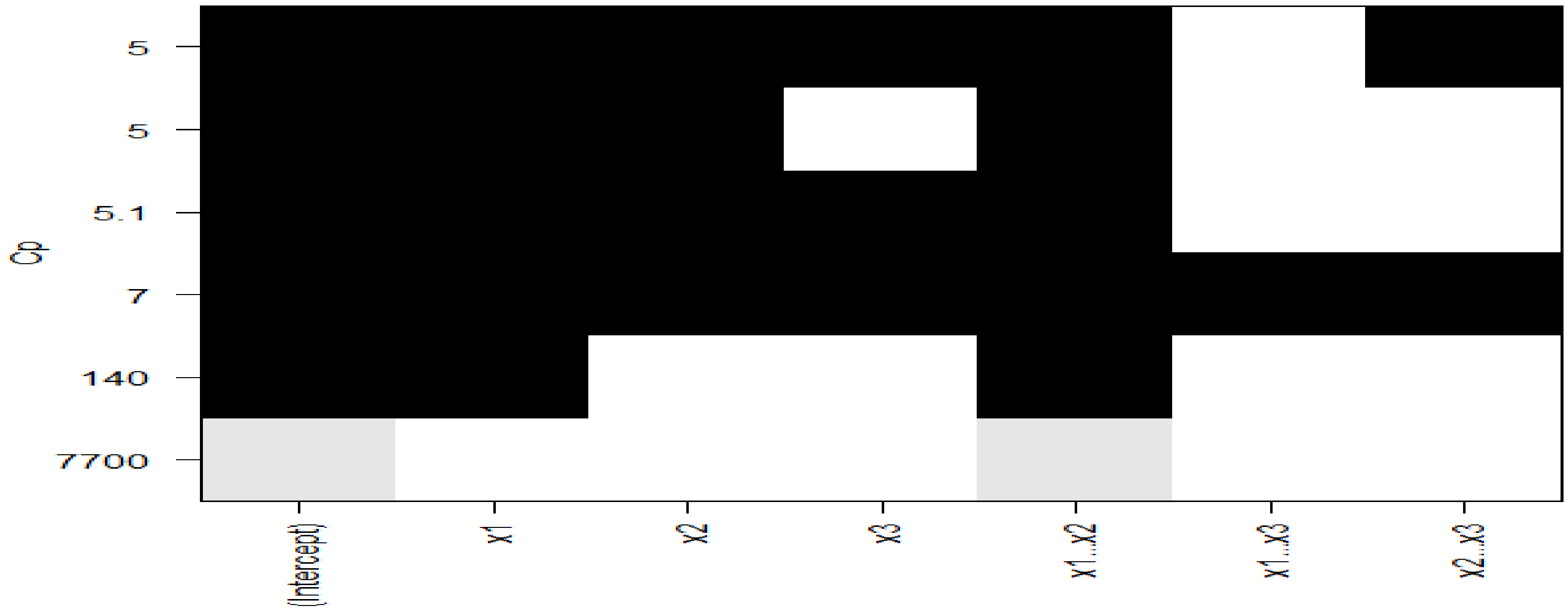
```
which.min(model.sum$rss)     # 6
which.min(model.sum$adjr2) # 1
which.min(model.sum$cp)     # 5
which.min(model.sum$bic)    # 3
```

Adjusted R-squared



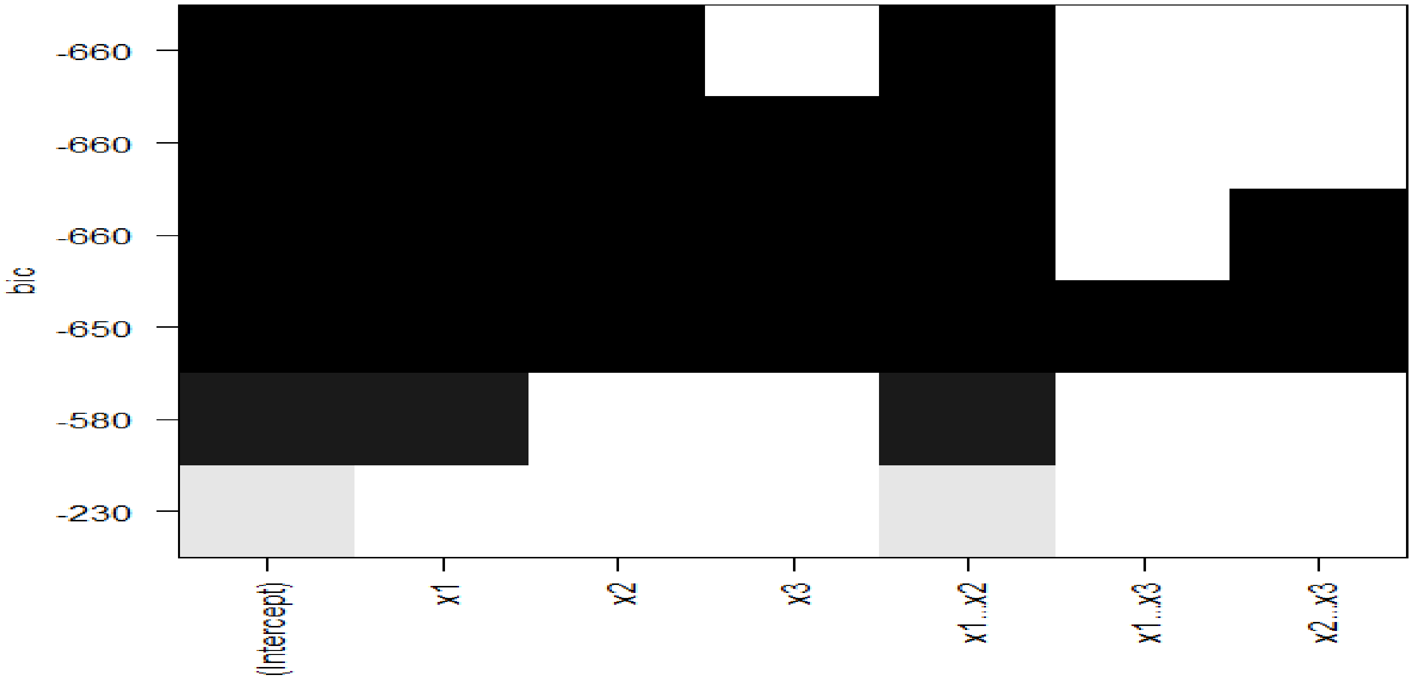
plot(model.com,scale="adjr2")

Cp (AIC)



plot(model.com,scale="Cp")

BIC



plot(model.com,scale=“bic")