MATH - 4360: Linear Statistical Models

Chapter 4, 6, and 9: Model Adequacy Checking, Diagnostic for Leverage and Influence & Multicollinearity

Suthakaran Ratnasingam

The fitting of the linear regression model, estimation of parameters testing of hypothesis properties of the estimator, is based on the following major assumptions:

- The relationship between the study variable and explanatory variables is linear, at least approximately.
- The error term, ϵ has zero mean.
- The error term, ϵ has a constant variance σ^2.
- The errors are uncorrelated.
- The errors are normally distributed.

```
rm(list = ls())
# I assume that you have installed the following R packages. If not, please install
# them using the R function: install.packages('package_name')
library(olsrr)
library(ggfortify)
library(ggplot2)
library(car)
library(Rcpp)
library(GGally)
library(asbio)
library(matlib) # enables function inv()
data1 = read.table("D:\\CSUSB\\Fall 2021\\MATH 4360\\RNotes\\ex31.txt", header = TRUE)
head(data1)
```

```
##    Time Cases Distance
## 1 16.68     7      560
## 2 11.50     3      220
## 3 12.03     3      340
## 4 14.88     4       80
## 5 13.75     6      150
## 6 18.11     7      330
```

```
n = nrow(data1)
Fit1 = lm(Time ~ Cases + Distance, data = data1)
p = length(coef(Fit1))
summary(Fit1)
```

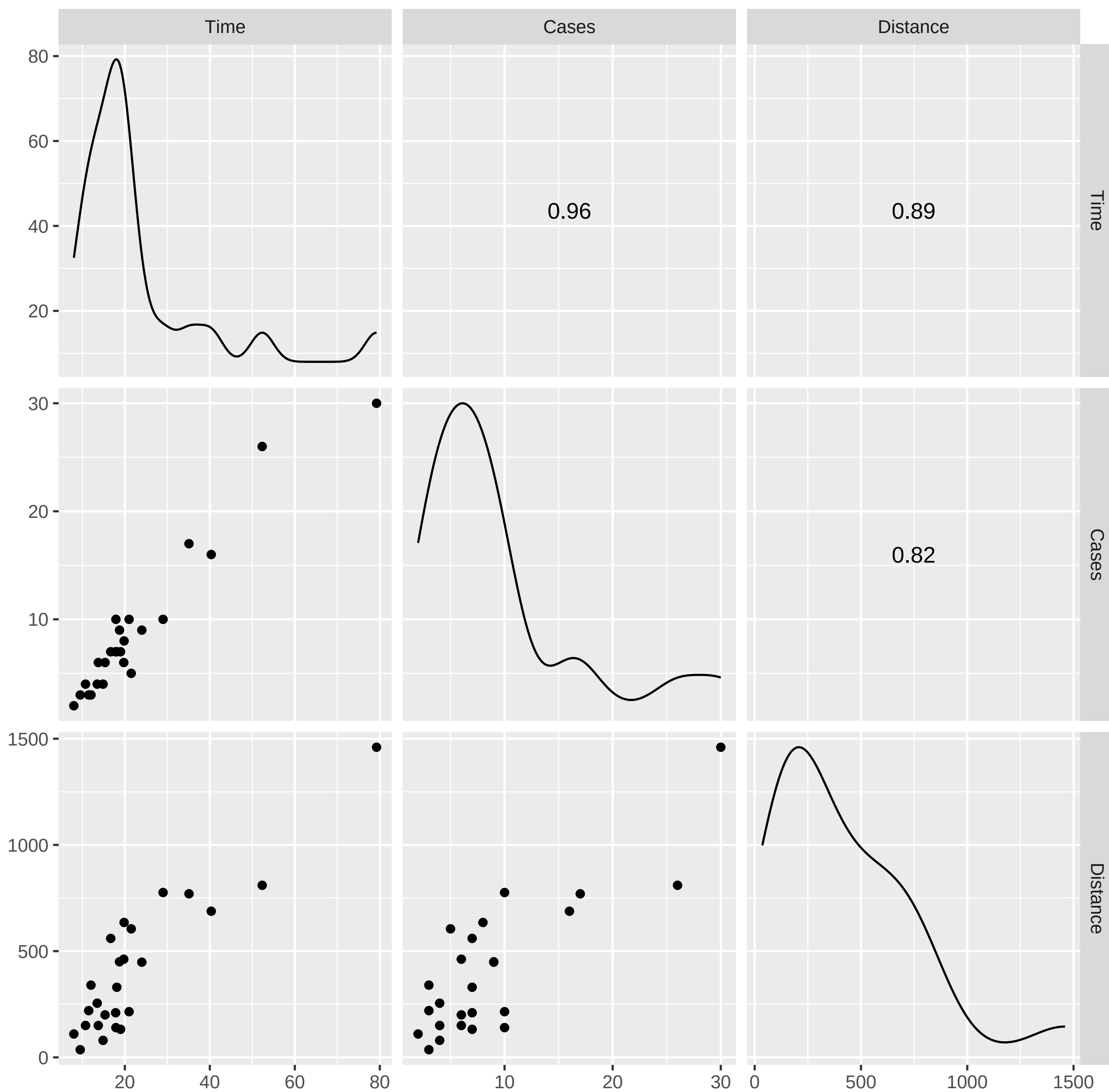
```
##
## Call:
## lm(formula = Time ~ Cases + Distance, data = data1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7880 -0.6629  0.4364  1.1566  7.4197
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.341231   1.096730   2.135 0.044170 *
## Cases       1.615907   0.170735   9.464 3.25e-09 ***
```

```
## Distance    0.014385   0.003613   3.981 0.000631 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.259 on 22 degrees of freedom
## Multiple R-squared:  0.9596, Adjusted R-squared:  0.9559
## F-statistic: 261.2 on 2 and 22 DF,  p-value: 4.687e-16
```

Scatterplot Matrix

A scatterplot matrix is a two-dimensional array of two-dimension plots where each form contains a scatter diagram except for the diagonal.

```
GGally::ggscatmat(data1, columns = c("Time", "Cases", "Distance"))
```



Residual Analysis

The residual is defined as the difference between the observed and fitted value of study variable. The ith residual is defined as

$$e_i = y_i - \hat{y}_i, \quad i = 1, 2, \ldots n$$

where y_i is an observation and $\hat{y}_i$ is the corresponding fitted value.

```
#Calculate Residual e_i
e_i = residuals(Fit1)
e_i
```

```
##          1          2          3          4          5          6          7
## -5.0280843  1.1463854 -0.0497937  4.9243539 -0.4443983 -0.2895743  0.8446235
##          8          9         10         11         12         13         14
##  1.1566049  7.4197062  2.3764129  2.2374930 -0.5930409  1.0270093  1.0675359
##         15         16         17         18         19         20         21
##  0.6712018 -0.6629284  0.4363603  3.4486213  1.7931935 -5.7879699 -2.6141789
##         22         23         24         25
## -3.6865279 -4.6075679 -4.5728535 -0.2125839
```

Methods for Scaling Residuals

Standardized Residuals

The residuals are standardized based on the concept of residual minus its mean and divided by its standard deviation. Since $E(e_i) = 0$ and MS_{Res} estimates the approximate average variance, so logically the scaling of residual is

$$d_i = \frac{e_i}{\sqrt{MS_{Res}}}, \quad i = 1, 2, \ldots, n$$

is called as standardized residual for which $E(d_i) = 0,\ Var(d_i) = 1$ (have mean zero and approximately unit variance). So a large value of $d_i (> 3$ say) potentially indicates an outlier.

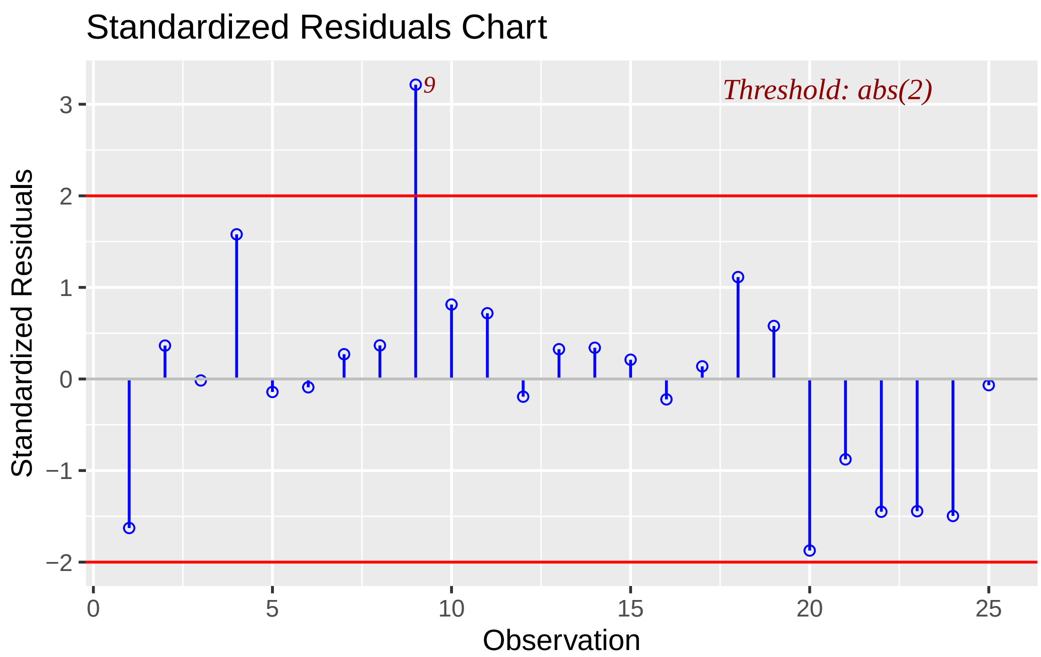
```
MS_res = (sigma(Fit1))^2
MS_res
```

```
## [1] 10.62417
```

```
# standardized residuals
d_i = e_i/sqrt(MS_res)
d_i
```

```
##           1           2           3           4           5           6
## -1.54260631  0.35170879 -0.01527661  1.51078203 -0.13634053 -0.08884082
##           7           8           9          10          11          12
##  0.25912883  0.35484408  2.27635117  0.72907878  0.68645843 -0.18194377
##          13          14          15          16          17          18
##  0.31508443  0.32751789  0.20592338 -0.20338513  0.13387449  1.05803019
##          19          20          21          22          23          24
##  0.55014821 -1.77573772 -0.80202492 -1.13101946 -1.41359270 -1.40294240
##          25
## -0.06522033
```

```
ols_plot_resid_stand(Fit1)  # library(olsrr)
```



Studentized Residuals

Internally studentized residuals has the form

$$r_i = \frac{e_i}{\sqrt{MS_{Res}(1 - h_{ii})}}$$

instead of e_i (or the standardized residuals d_i). For r_i, $E(r_i) = 0, Var(r_i) = 1$ regardless of the location of x_i when the form of the model is correct.

```
#Calculate h_ii
h_ii = lm.influence(Fit1)$hat
h_ii
```

```
##          1          2          3          4          5          6          7
## 0.10180178 0.07070164 0.09873476 0.08537479 0.07501050 0.04286693 0.08179867
##          8          9         10         11         12         13         14
## 0.06372559 0.49829216 0.19629595 0.08613260 0.11365570 0.06112463 0.07824332
##         15         16         17         18         19         20         21
## 0.04111077 0.16594043 0.05943202 0.09626046 0.09644857 0.10168486 0.16527689
##         22         23         24         25
## 0.39157522 0.04126005 0.12060826 0.06664345
```

```
# Calculate r_i
r_i = e_i/sqrt(MS_res*(1-h_ii))
r_i
```

```
##           1           2           3           4           5           6
## -1.62767993  0.36484267 -0.01609165  1.57972040 -0.14176094 -0.09080847
##           7           8           9          10          11          12
##  0.27042496  0.36672118  3.21376278  0.81325432  0.71807970 -0.19325733
##          13          14          15          16          17          18
##  0.32517935  0.34113547  0.21029137 -0.22270023  0.13803929  1.11295196
##          19          20          21          22          23          24
##  0.57876634 -1.87354643 -0.87784258 -1.44999541 -1.44368977 -1.49605875
##          25
## -0.06750861
```

R-Student

Externally studentized residual, usually called $R-$student, given by

$$t_i = \frac{e_i}{\sqrt{S_{(i)}^2(1-h_{ii})}}, \quad i = 1, 2, \ldots, n$$

where

$$S_{(i)}^2 = \frac{(n-p)MS_{Res} - e_i^2/(1-h_{ii})}{(n-p-1)}$$

If an observation has an externally studentized residual that is larger than 3 (in absolute value) we can call it an outlier.

```
as.vector(rstudent(Fit1))
```

```
##  [1] -1.69562881  0.35753764 -0.01572177  1.63916491 -0.13856493 -0.08873728
##  [7]  0.26464769  0.35938983  4.31078012  0.80677584  0.70993906 -0.18897451
## [13]  0.31846924  0.33417725  0.20566324 -0.21782566  0.13492400  1.11933065
## [19]  0.56981420 -1.99667657 -0.87308697 -1.48962473 -1.48246718 -1.54221512
## [25] -0.06596332
```

```
ols_plot_resid_stud(Fit1)
```

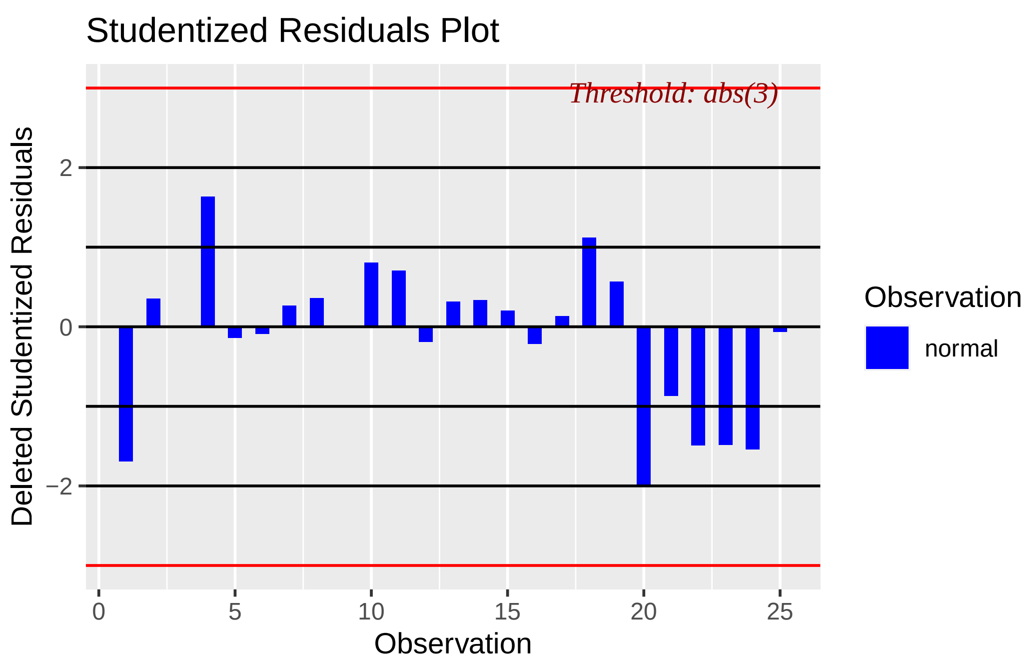


TABLE 4.1 Scaled Residuals for Example 4.1

Here, I reproduced a Table 4.1 from your textbook.

```
# Calculate Residual e_i
e_i = residuals(Fit1)

MS_res = (sigma(Fit1))^2
# standardized residuals
d_i = e_i/sqrt(MS_res)

#Calculate h_ii
```

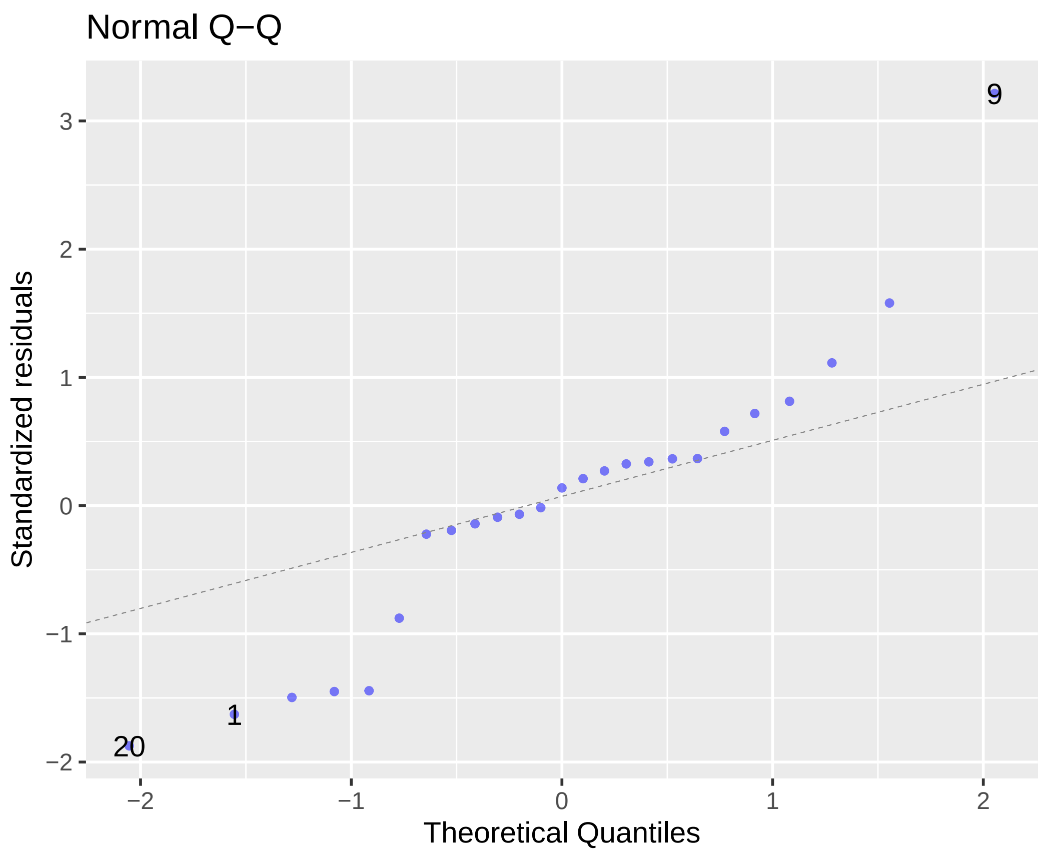
```r
h_ii = lm.influence(Fit1)$hat

# Calculate r_i
r_i = e_i/sqrt(MS_res*(1-h_ii))

#Calculate e_deleted_i = e_(i)
e_deleted_i = e_i/(1-h_ii)

#Calculate Studentized residuals, t_i
t_i <- rstudent(Fit1)

#Calculate e_deleted_i (e_(i)) squared
e_deleted_i_squared = (e_i/(1-h_ii))^2

# Table Scaled Residuals
Table1 = data.frame(Obs = 1:n, e_i = e_i, d_i = d_i, h_ii = h_ii, r_i = r_i,
                    e_deleted_i = e_deleted_i, t_i = t_i, e_deleted_i_squared = e_deleted_i_squared )
Table = round(Table1, 4)
Table
```

```
##    Obs     e_i     d_i   h_ii     r_i e_deleted_i     t_i e_deleted_i_squared
## 1    1 -5.0281 -1.5426 0.1018 -1.6277     -5.5980 -1.6956             31.3372
## 2    2  1.1464  0.3517 0.0707  0.3648      1.2336  0.3575              1.5218
## 3    3 -0.0498 -0.0153 0.0987 -0.0161     -0.0552 -0.0157              0.0031
## 4    4  4.9244  1.5108 0.0854  1.5797      5.3840  1.6392             28.9876
## 5    5 -0.4444 -0.1363 0.0750 -0.1418     -0.4804 -0.1386              0.2308
## 6    6 -0.2896 -0.0888 0.0429 -0.0908     -0.3025 -0.0887              0.0915
## 7    7  0.8446  0.2591 0.0818  0.2704      0.9199  0.2646              0.8462
## 8    8  1.1566  0.3548 0.0637  0.3667      1.2353  0.3594              1.5260
## 9    9  7.4197  2.2764 0.4983  3.2138     14.7889  4.3108            218.7115
## 10  10  2.3764  0.7291 0.1963  0.8133      2.9568  0.8068              8.7428
## 11  11  2.2375  0.6865 0.0861  0.7181      2.4484  0.7099              5.9946
## 12  12 -0.5930 -0.1819 0.1137 -0.1933     -0.6691 -0.1890              0.4477
## 13  13  1.0270  0.3151 0.0611  0.3252      1.0939  0.3185              1.1966
## 14  14  1.0675  0.3275 0.0782  0.3411      1.1582  0.3342              1.3413
## 15  15  0.6712  0.2059 0.0411  0.2103      0.7000  0.2057              0.4900
## 16  16 -0.6629 -0.2034 0.1659 -0.2227     -0.7948 -0.2178              0.6317
## 17  17  0.4364  0.1339 0.0594  0.1380      0.4639  0.1349              0.2152
## 18  18  3.4486  1.0580 0.0963  1.1130      3.8159  1.1193             14.5614
## 19  19  1.7932  0.5501 0.0964  0.5788      1.9846  0.5698              3.9387
## 20  20 -5.7880 -1.7757 0.1017 -1.8735     -6.4431 -1.9967             41.5140
## 21  21 -2.6142 -0.8020 0.1653 -0.8778     -3.1318 -0.8731              9.8081
## 22  22 -3.6865 -1.1310 0.3916 -1.4500     -6.0591 -1.4896             36.7131
## 23  23 -4.6076 -1.4136 0.0413 -1.4437     -4.8059 -1.4825             23.0963
## 24  24 -4.5729 -1.4029 0.1206 -1.4961     -5.2000 -1.5422             27.0402
## 25  25 -0.2126 -0.0652 0.0666 -0.0675     -0.2278 -0.0660              0.0519
```

Normal Probability Plot

- The assumption of normality of disturbances/error is very much needed for the validity of the results for testing of hypothesis, confidence intervals and prediction intervals.
- The normal probability plots help in verifying the assumption of normal distribution. If errors are coming from a distribution with thicker and heavier tails than normal, then the least-squares fit may be sensitive to a small set of data.

```r
# Use the R package "ggfortify"
ggplot2::autoplot(Fit1, which = 2, colour = "blue", alpha = 0.5, size = 1, ncol = 1)
```



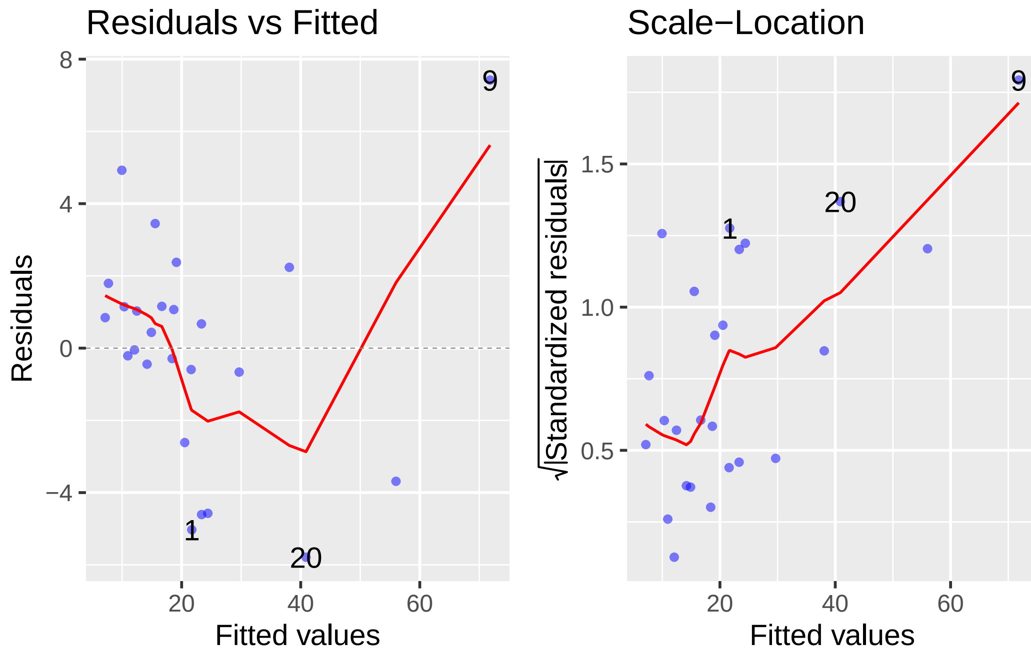

```
# alternatively, plot(Fit1, which = 1)
```

Plots of residuals against the fitted value $\hat{y}_i$

fitted value $\hat{y}_i$} A plot of residuals (e_i) or any of the scaled residuals (d_i, r_i, t_i) versus the corresponding fitted values $\hat{y}_i$ is helpful in detecting several common types of model inadequacies. These plots can help us identify:

- Non-constant variance
- Violation of the assumption of linearity
- Potential outliers

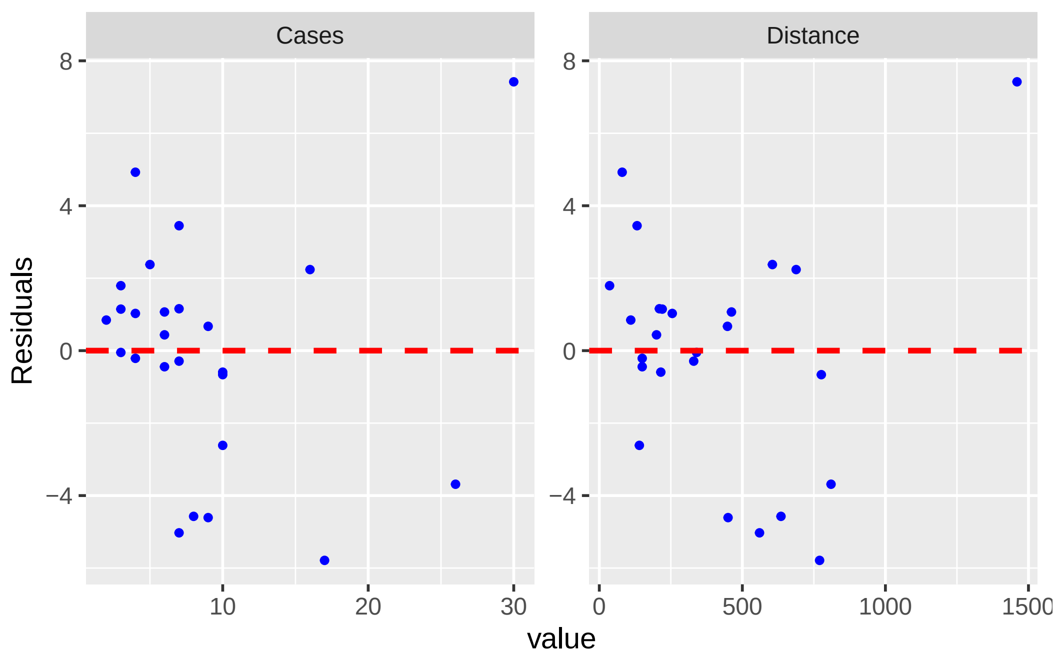
```
ggplot2::autoplot(Fit1, which = c(1, 3),   colour = "blue",
                  smooth.colour = "red",  alpha = 0.5,  size = 1)
```



Plots of residuals against explanatory variable (the Regressor)

Plotting of residuals against the corresponding values of each explanatory variable can also be helpful.

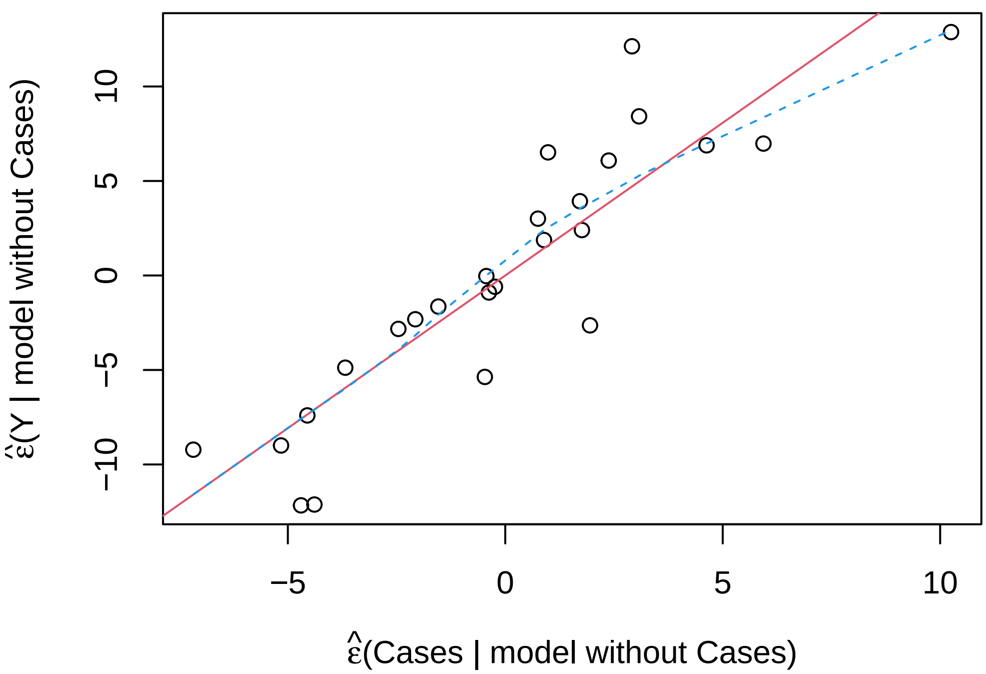
```
data1$resid = residuals(Fit1)
library(reshape2)
mydf = melt(data1[, c("Cases", "Distance", "resid")], id="resid")
library(ggplot2)
ggplot(mydf, aes(x=value, y=resid)) +
  geom_point(col = "blue", size = 1) + facet_wrap(~ variable, scales = "free") +
  geom_hline(yintercept=0, linetype="dashed", color = "red", size = 1) +
  labs(y = "Residuals")
```



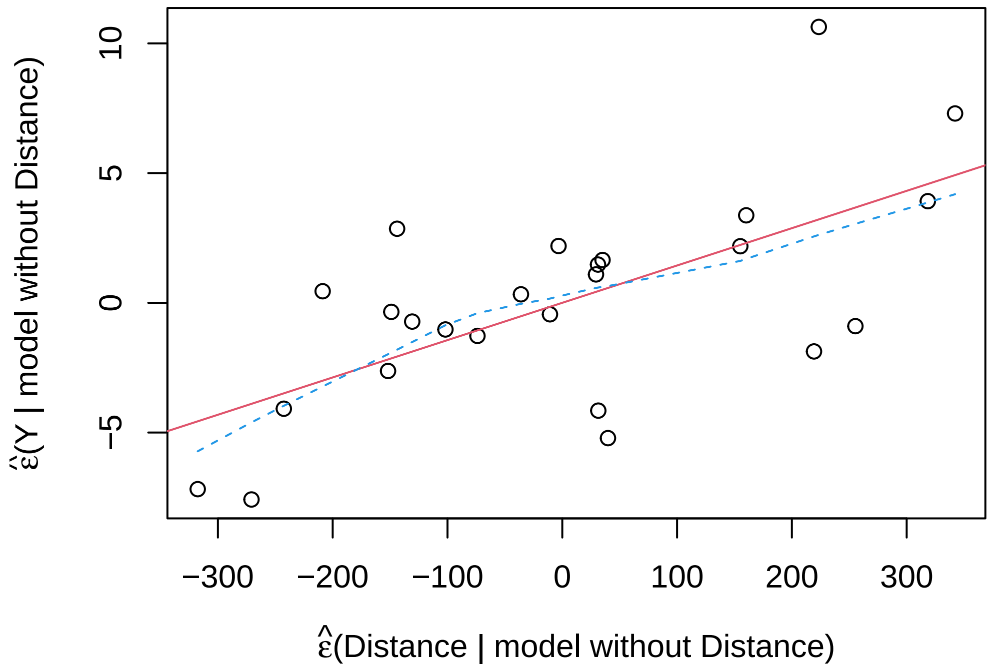
Partial Regression and Partial Residual Plots

- Partial regression plot (also called as added variable plot or adjusted variable plot) is a variation of the plot of residuals versus the predictor.
- It helps better to study the marginal relationship of an explanatory variable given the other variables that are in the model.
- Also called the added variable plot or the adjusted variable plot.
- In partial regression plot
  - Regress y on all the explanatory variable except the jth explanatory variables X_j and obtain the residuals $e\big[y/X_{(j)}\big]$, say where $X_{(j)}$ denotes the $X-$ matrix with X_j removed.
  - Regress X_j on all other explanatory variables and obtain the residuals $e\big[X_j/X_{(j)}\big]$, say
  - Plot $e\big[y/X_{(j)}\big]$ against $e\big[X_j/X_{(j)}\big]$

```
Fit1 = lm(Time ~ Cases + Distance, data = data1)
partial.resid.plot(Fit1) # r package asbio
```




```
## Press return for next plot
```




```
## Press return for next plot
```

PRESS Statistics

The PRESS residuals are defined as

$$e_{(i)} = y_i - \hat{y}_{(i)}, \quad i = 1, 2, \ldots, n$$

where $\hat{y}_{(i)}$ is the predicted value of the ith observed study variable based on a model fit to the remaining $(n-1)$ points. The large residuals are useful in identifying those observations where the model does not fit well or the observations for which the model is likely to provide poor predictions for future values. The prediction sum of squares is defined as the sum of squared PRESS residuals and is called as PRESS statistic as

$$PRESS = \sum_{i=1}^{n} \left[y_i - \hat{y}_{(i)} \right]^2 = \sum_{i=1}^{n} \left(\frac{e_i}{1 - h_{ii}} \right)^2$$

```
#PRESS Statistics
PRESS_stat = sum(Table$e_deleted_i_squared)
PRESS_stat
```

```
## [1] 459.0393
```

Test for Lack of Fit of a Regression Model

The test for lack of fit of a regression model is based on the assumptions of normality, independence and constant variance which are satisfied.

Let y_i be the mean of n_i observations on x_i. Then the (i, j)th residual is

$$
\begin{aligned}
(y_{ij} - \hat{y}_i) &= (y_{ij} - \bar{y}_i) + (\bar{y}_i - \hat{y}_i) \\
\sum_{i=1}^{m}\sum_{i=1}^{n_i}(y_{ij} - \hat{y}_i)^2 &= \sum_{i=1}^{m}\sum_{i=1}^{n_i}(y_{ij} - \bar{y}_i)^2 + \sum_{i=1}^{m}\sum_{i=1}^{n_i}(\bar{y}_i - \hat{y}_i)^2 \\
&\quad \text{(obtained by squaring and summing over } i \text{ and } j \text{)} \\
SS_{Res} &= SS_{PE} + SS_{LOF} \\
\text{Residual sum of squares} &= \text{Sum of squares due to pure error} + \text{sum of squares due to lack of fit} \\
&= \text{Measures pure error} + \text{Measures of lack of fit}
\end{aligned}
$$

Example 4.8 Testing for Lack of Fit Let's use R to reproduce Analysis of Variance table for Example 4.8.

```
LOF_data = data.frame(x = c(1.0, 1.0, 2.0, 3.3, 3.3, 4.0, 4.0, 4.0,
                            4.7, 5.0, 5.6, 5.6, 5.6, 6.0, 6.0, 6.5, 6.9),
                      y = c(10.84, 9.30, 16.35, 22.88, 24.35, 24.56, 25.86, 29.16, 24.59, 22.25,
                            25.90, 27.20, 25.61, 25.45, 26.56, 21.03, 21.46))

Ori_model = lm(y ~ x, data = LOF_data)
summary(Ori_model)
```

```
##
## Call:
## lm(formula = y ~ x, data = LOF_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.4536 -1.6158  0.5638  2.6358  7.4246
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  13.2139     2.6649   4.959 0.000172 ***
## x             2.1304     0.5645   3.774 0.001839 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.084 on 15 degrees of freedom
## Multiple R-squared:  0.487,  Adjusted R-squared:  0.4528
## F-statistic: 14.24 on 1 and 15 DF,  p-value: 0.001839
```

```
anova(Ori_model)
```

```
## Analysis of Variance Table
##
## Response: y
##           Df Sum Sq Mean Sq F value   Pr(>F)
## x          1 237.48 237.479  14.241 0.001839 **
## Residuals 15 250.13  16.676
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
LOF_data$fac_x = as.factor(LOF_data$x)
anova_fit = anova(update(Ori_model, . ~ . + factor(x)))
as.table(cbind(
  'SS' = c('SSR'   =     anova_fit[1,   2],
           'SSE'   = sum(anova_fit[2:3, 2]),
           'SSLF'  =     anova_fit[2,   2],
           'SSPE'  =     anova_fit[3,   2],
           'Total' = sum(anova_fit[1:3, 2])),
```

```
  'Df' = c(              anova_fit[1,   1],
                         sum(anova_fit[2:3, 1]),
                         anova_fit[2,   1],
                         anova_fit[3,   1],
                         sum(anova_fit[1:3, 1])),

  'MS' = c(              anova_fit[1,   3],
                         sum(anova_fit[2:3, 2]) / sum(anova_fit[2:3, 1]),
                         anova_fit[2,   3],
                         anova_fit[3,   3],
                         NA),

  'F-Test' = c(          NA,
                         NA,
                         anova_fit[2,   3]/anova_fit[3,   3],
                         NA,
                         NA)
))
```

```
##              SS        Df        MS   F-Test
## SSR   237.47877   1.00000 237.47877
## SSE   250.13383  15.00000  16.67559
## SSLF  234.57080   8.00000  29.32135 13.18827
## SSPE   15.56303   7.00000   2.22329
## Total 487.61260  16.00000
```

Chapter - 6: Diagnostics Leverage and Influence

A leverage point is an observation that has an unusual predictor value (very different from the bulk of the observations)

- If $h_{ii} > 2\bar{h} = \dfrac{2p}{n} \implies$ the point is remote enough from rest of the data to be considered as a leverage point.

```
data1 = read.table("D:\\CSUSB\\Fall 2021\\MATH 4360\\RNotes\\ex31.txt", header = TRUE)
head(data1)
```

```
##    Time Cases Distance
## 1 16.68     7      560
## 2 11.50     3      220
## 3 12.03     3      340
## 4 14.88     4       80
## 5 13.75     6      150
## 6 18.11     7      330
```

```
n = nrow(data1)
X = matrix(c(rep(1, length(data1$Time)),  data1$Cases, data1$Distance),  ncol = 3)
# Lets find the hat matrix
H = X %*% solve(t(X) %*% X) %*% t(X)
data.frame(Cases = data1$Cases, Distance = data1$Distance, H = diag(H))
```

```
##    Cases Distance          H
## 1      7      560 0.10180178
## 2      3      220 0.07070164
## 3      3      340 0.09873476
## 4      4       80 0.08537479
## 5      6      150 0.07501050
## 6      7      330 0.04286693
```

```
## 7      2      110 0.08179867
## 8      7      210 0.06372559
## 9     30     1460 0.49829216
## 10     5      605 0.19629595
## 11    16      688 0.08613260
## 12    10      215 0.11365570
## 13     4      255 0.06112463
## 14     6      462 0.07824332
## 15     9      448 0.04111077
## 16    10      776 0.16594043
## 17     6      200 0.05943202
## 18     7      132 0.09626046
## 19     3       36 0.09644857
## 20    17      770 0.10168486
## 21    10      140 0.16527689
## 22    26      810 0.39157522
## 23     9      450 0.04126005
## 24     8      635 0.12060826
## 25     4      150 0.06664345
```

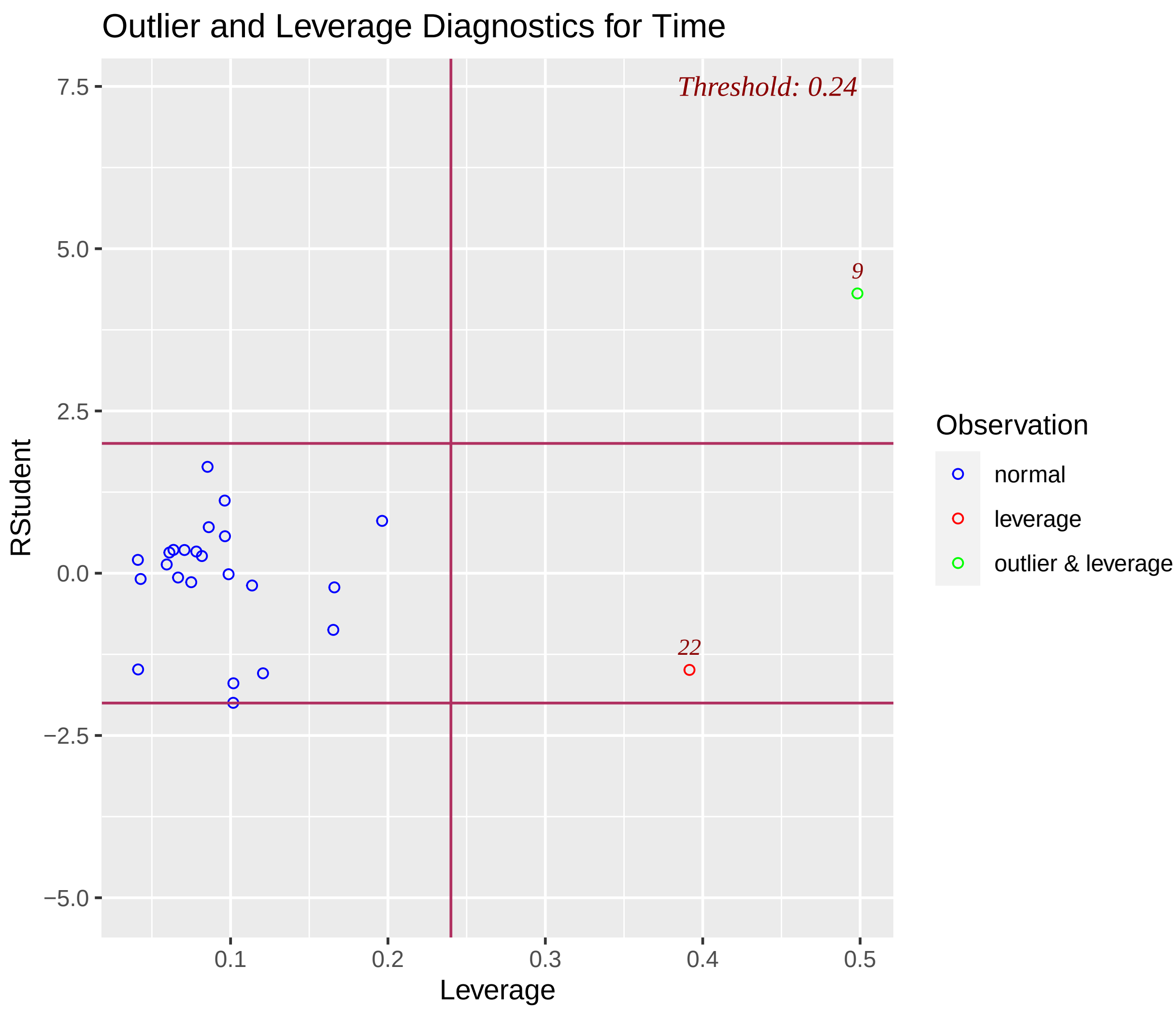
```
sum(diag(H)) # number of predictor variables
```

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

```
Fit1 = lm(Time ~ Cases + Distance, data = data1)
Leverage_point = hatvalues(Fit1) > 2 * mean(hatvalues(Fit1))
data1[Leverage_point,] # look at the high leverage point(s)
```

```
##     Time Cases Distance
## 9  79.24    30     1460
## 22 52.32    26      810
```

```
ols_plot_resid_lev(Fit1)
```




```r
# ols_plot_diagnostics(Fit1) # This function gives all diagnostic plots
```

Measures of Influence

If data set is small, then the deletion of values greatly affects the fit and statistical conclusions. In measuring influence, it is desirable to consider both

- the location of point is $x-$space and
- the response variable.

The Cook's distance statistics denoted as, Cook's $D-$statistic is a measure of the distance between the least-squares estimate based on all n observations in $\hat{\beta}$ and the estimate obtained by deleting the ith point, say $\hat{\beta}_i$. It is given by

$$D_i = \frac{(\hat{y} - \hat{y}_i)'(\hat{y} - \hat{y}_i)}{p\,MS_{Res}}, \quad i = 1, 2, \ldots, n$$

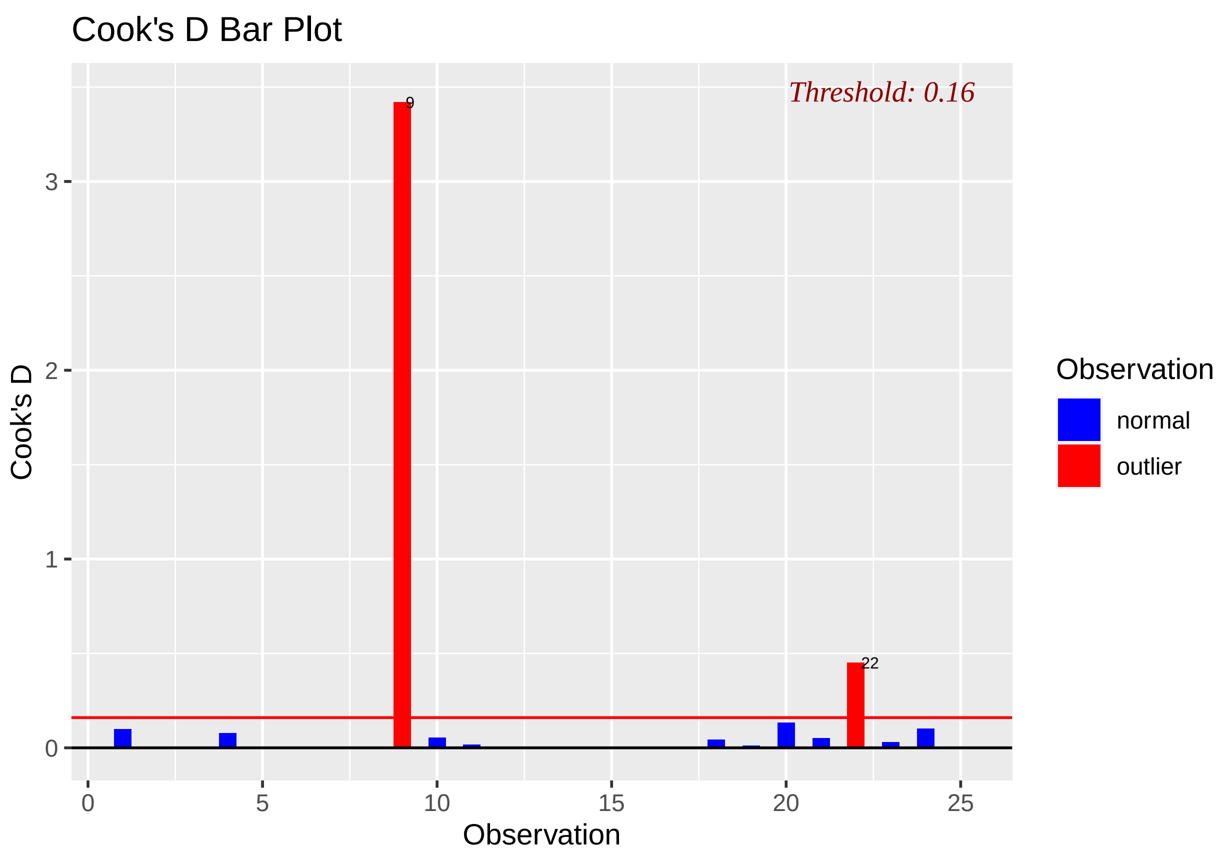
where $\hat{y} = X\hat{\beta},\ \hat{y}_i = X\hat{\beta}_i, \hat{\beta} = (X'X)^{-1}X'y$

```r
cooks.distance(Fit1)
```

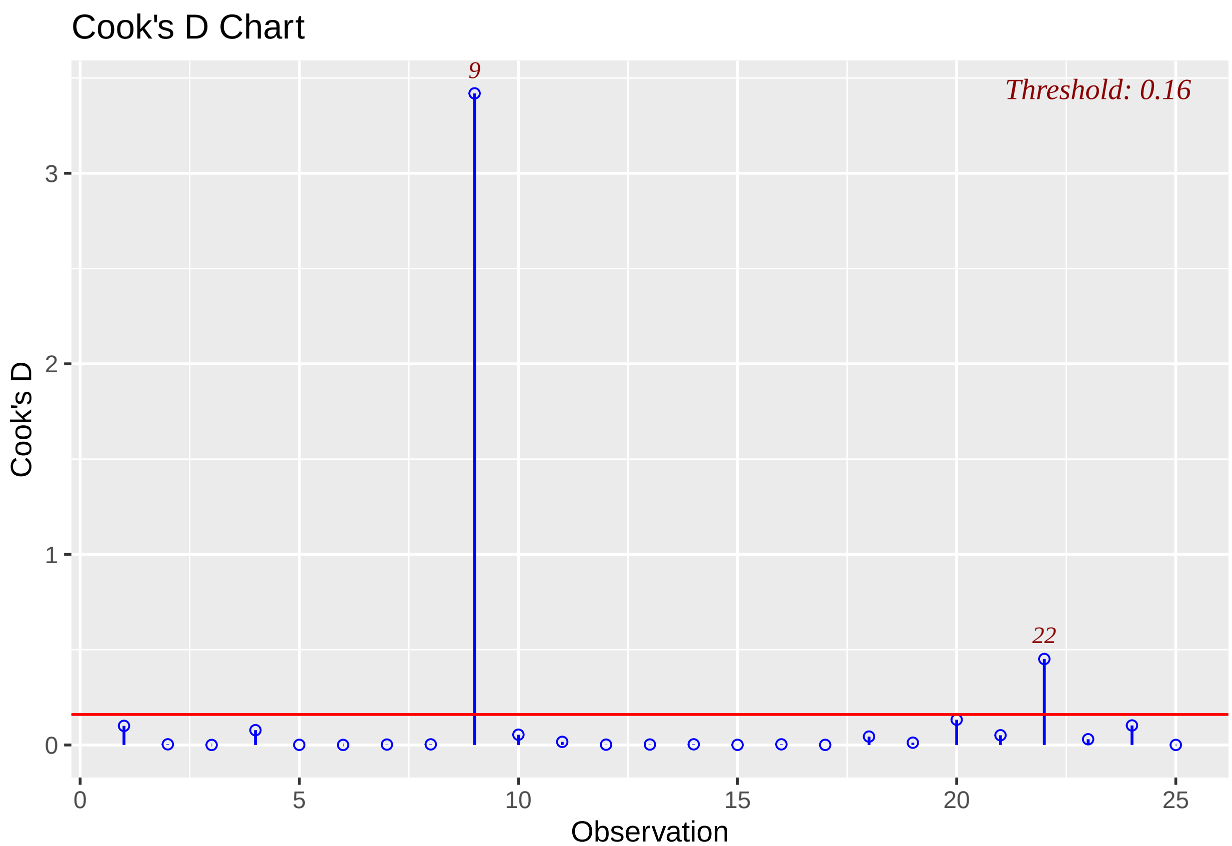
```
##            1            2            3            4            5            6
## 1.000921e-01 3.375704e-03 9.455785e-06 7.764718e-02 5.432217e-04 1.231067e-04
##            7            8            9           10           11           12
## 2.171604e-03 3.051135e-03 3.419318e+00 5.384516e-02 1.619975e-02 1.596392e-03
```

```
##            13            14            15            16            17            18
## 2.294737e-03 3.292786e-03 6.319880e-04 3.289086e-03 4.013419e-04 4.397807e-02
##            19            20            21            22            23            24
## 1.191868e-02 1.324449e-01 5.086063e-02 4.510455e-01 2.989892e-02 1.023224e-01
##            25
## 1.084694e-04
```

```
ols_plot_cooksd_bar(Fit1)    # threshold = 4/n = 4/25 = 0.16
```




```
ols_plot_cooksd_chart(Fit1)
```



Cook's distance measure is a deletion diagnostic, i.e., it measures the influence of ith observation if it is removed from the sample.

DFBETAS

DFBETAS which indicates how much the regression coefficient changes if the ith observation were deleted. Such change is measured in terms of standard deviation units. This statistic is

$$DFBETAS_{j,i} = \frac{(\hat{\beta}_j - \hat{\beta}_{j(i)})}{\sqrt{S^2_{(i)} C_{jj}}}$$

where C_{jj} is the jth diagonal element of $(X'X)^{-1}$ and $\hat{\beta}_{j(i)}$ regression coefficient computed without the use of ith observation.

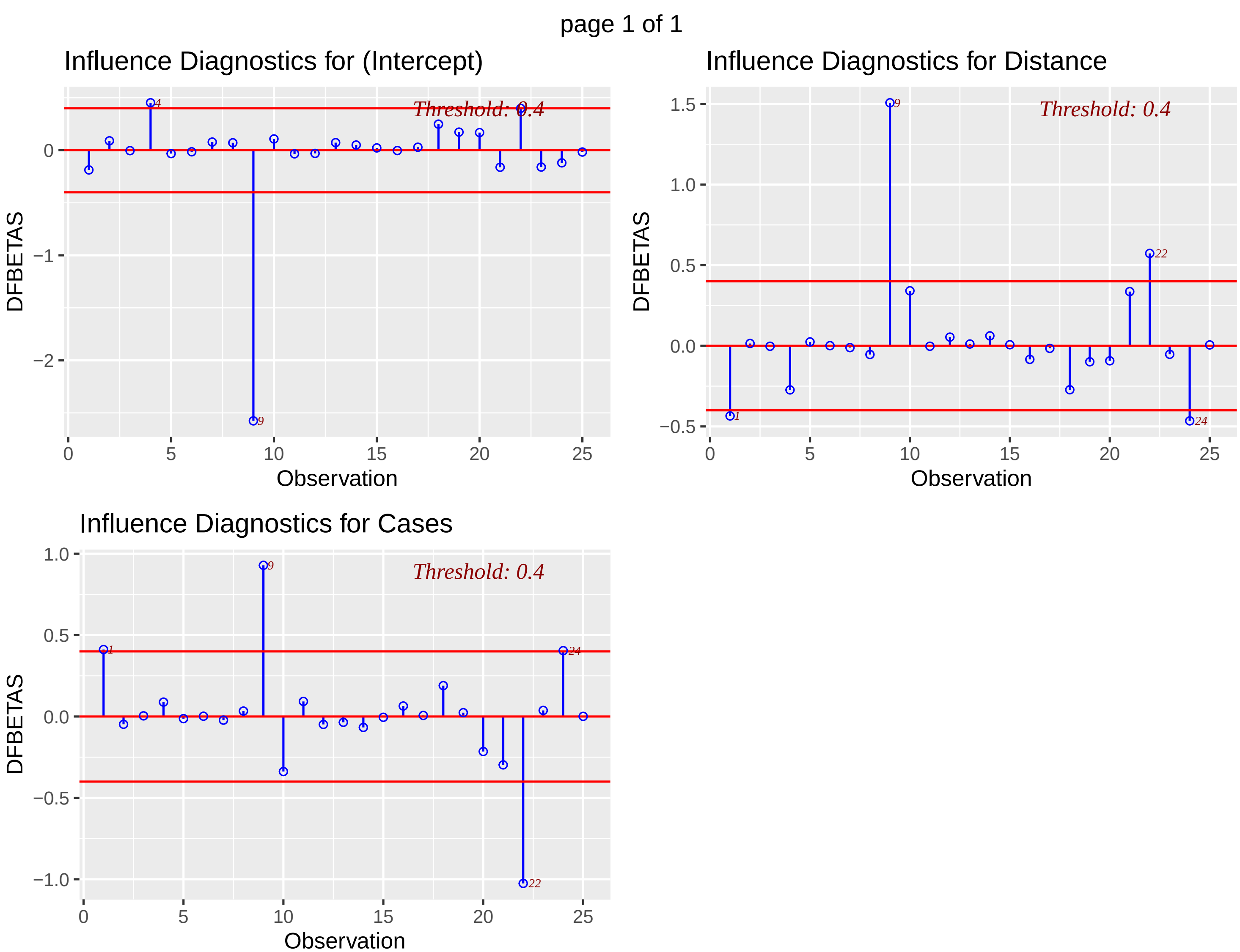
- If $|DFBETAS_{j,i}| > \frac{2}{\sqrt{n}}$, then ith observation warrants examination.

```
dfbetas(Fit1)
```

```
##     (Intercept)         Cases     Distance
## 1  -0.187267279  0.4113118750 -0.434862094
## 2   0.089793299 -0.0477642427  0.014414155
## 3  -0.003515177  0.0039483525 -0.002846468
## 4   0.451964743  0.0882802920 -0.273373097
## 5  -0.031674102 -0.0133001129  0.024240457
## 6  -0.014681480  0.0017921068  0.001078986
## 7   0.078071285 -0.0222783194 -0.011018802
## 8   0.071202807  0.0333823324 -0.053823961
## 9  -2.575739806  0.9287433421  1.507550618
## 10  0.107919369 -0.3381628707  0.341326746
## 11 -0.034274535  0.0925271540 -0.002686252
## 12 -0.030268935 -0.0486664488  0.053973390
```

```
## 13  0.072366473 -0.0356212226  0.011335105
## 14  0.049516699 -0.0670868604  0.061816778
## 15  0.022279094 -0.0047895025  0.006838236
## 16 -0.002693186  0.0644208340 -0.084187552
## 17  0.028855555  0.0064876499 -0.015696507
## 18  0.248558020  0.1897331043 -0.272430555
## 19  0.172558506  0.0235737344 -0.098968842
## 20  0.168036548 -0.2149950233 -0.092915080
## 21 -0.161928685 -0.2971750929  0.336406248
## 22  0.398566309 -1.0254140704  0.573140240
## 23 -0.159852248  0.0372930389 -0.052651959
## 24 -0.119720216  0.4046225960 -0.465446949
## 25 -0.016816024  0.0008498979  0.005592192
```

```
ols_plot_dfbetas(Fit1)
```



DFFITS

The deletion influence of ith observation on the predicted or fitted value can be investigated by using diagnostic as

$$DFFITS_i = \frac{(\hat{y} - \hat{y}_i)}{\sqrt{S_{(i)}^2 h_{jj}}}$$

where $\hat{y}_i$ is the fitted value of y_i obtained without the use of the ith observation. The denominator is just a standardization, since $Var(\hat{y}_i) = \sigma^2 h_{ii}$

- Thus $DFFITS_i$ is affected by both leverage and prediction error

$$|DFFITS_i| > 2\sqrt{\frac{p}{n}}$$

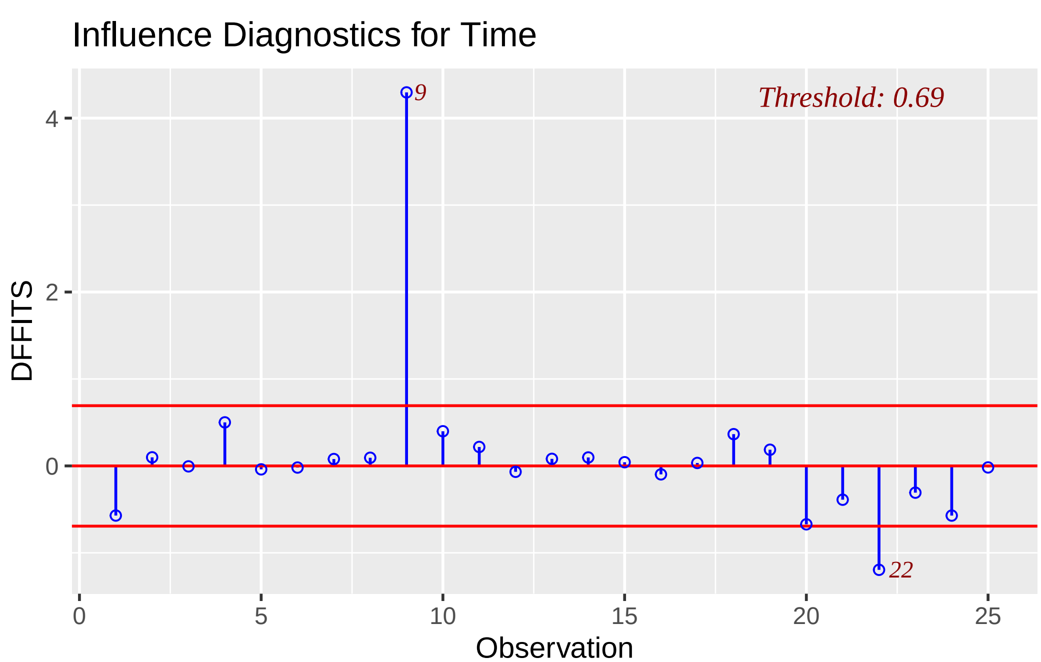
```
dffits(Fit1)
```

```
##            1            2            3            4            5            6
## -0.570850478  0.098618619 -0.005203676  0.500801817 -0.039458989 -0.018779374
##            7            8            9           10           11           12
##  0.078990030  0.093760764  4.296080927  0.398713071  0.217953207 -0.067670223
##           13           14           15           16           17           18
##  0.081259033  0.097362643  0.042584374 -0.097159801  0.033915978  0.365309285
##           19           20           21           22           23           24
##  0.186167873 -0.671771402 -0.388501185 -1.195036104 -0.307538544 -0.571139627
##           25
## -0.017626149
```

```
check_obs_dffits = abs(dffits(Fit1)) > 2 * sqrt(p/n)
data1[check_obs_dffits,]
```

```
##     Time Cases Distance
## 9  79.24    30     1460
## 22 52.32    26      810
```

```
ols_plot_dffits(Fit1) # Threshold 2 * sqrt(p/n)
```



A Measure of Model Performance

- The diagnostics D_i, $DFBETAS_{j,i}$, and $DFFITS_i$ provide insight about the effect of observations on the estimated coefficients $\hat{\beta}_j$ and fitted values $\hat{y}_j$
- To express the role of the ith observation on the precision of estimation, we could define

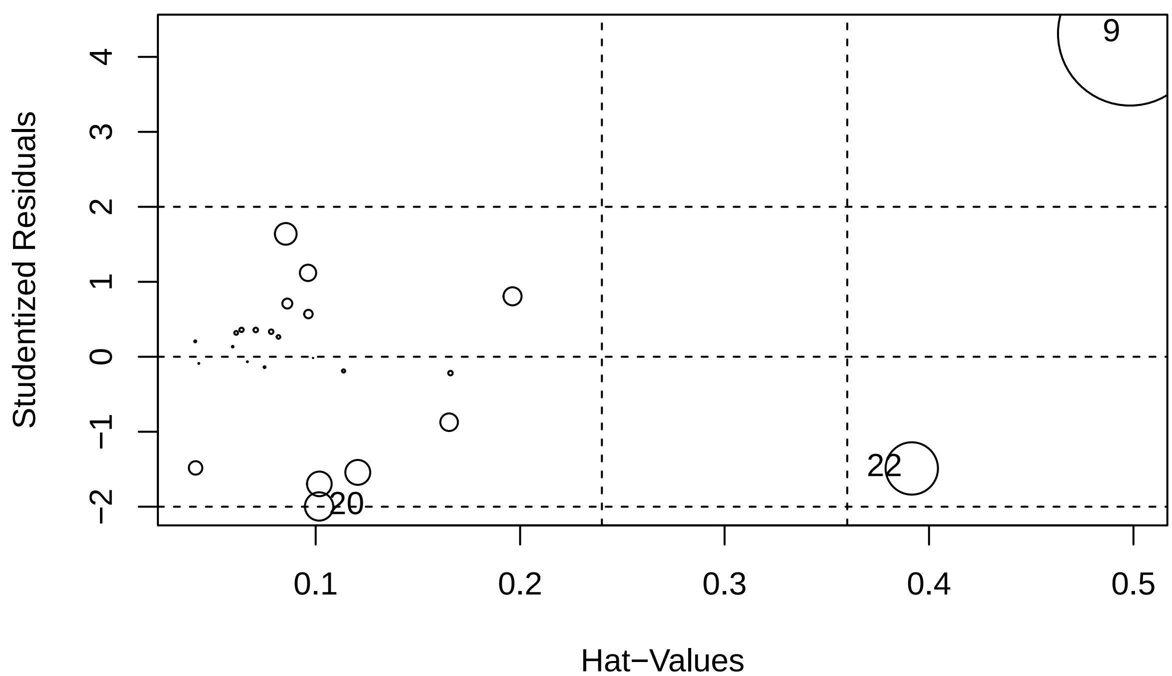
$$COVRATIO_i = \frac{|(X'_{(i)}X_{(i)})^{-1}S^2_{(i)}|}{|(X'X)^{-1}MS_{Res}|}, \quad i = 1, 2, \ldots, n$$

- If $COVRATIO_i > 1$, removing the ith observation observation degrades precision (and including it improves the precision of estimation)
- If $COVRATIO_i < 1$, removing the ith observation improves precision (and including it degrades the precision of estimation)

```
covratio(Fit1)
```

```
##         1         2         3         4         5         6         7         8 
## 0.8710782 1.2149209 1.2756813 0.8759964 1.2396032 1.1999120 1.2397501 1.2056413 
##         9        10        11        12        13        14        15        16 
## 0.3422132 1.3054035 1.1717266 1.2906069 1.2070490 1.2276758 1.1918460 1.3692181 
##        17        18        19        20        21        22        23        24 
## 1.2192451 1.0692145 1.2152541 0.7598217 1.2376914 1.3980787 0.8896761 0.9476321 
##        25 
## 1.2310981
```

```
influencePlot(Fit1)
```




```
##      StudRes       Hat     CookD
## 9   4.310780 0.4982922 3.4193184
## 20 -1.996677 0.1016849 0.1324449
## 22 -1.489625 0.3915752 0.4510455
```

```
influence.measures(Fit1)
```

```
## Influence measures of
##   lm(formula = Time ~ Cases + Distance, data = data1) :
```

```
##
##       dfb.1_ dfb.Cass dfb.Dstn   dffit cov.r   cook.d    hat inf
## 1  -0.18727  0.41131 -0.43486 -0.5709 0.871 1.00e-01 0.1018
## 2   0.08979 -0.04776  0.01441  0.0986 1.215 3.38e-03 0.0707
## 3  -0.00352  0.00395 -0.00285 -0.0052 1.276 9.46e-06 0.0987
## 4   0.45196  0.08828 -0.27337  0.5008 0.876 7.76e-02 0.0854
## 5  -0.03167 -0.01330  0.02424 -0.0395 1.240 5.43e-04 0.0750
## 6  -0.01468  0.00179  0.00108 -0.0188 1.200 1.23e-04 0.0429
## 7   0.07807 -0.02228 -0.01102  0.0790 1.240 2.17e-03 0.0818
## 8   0.07120  0.03338 -0.05382  0.0938 1.206 3.05e-03 0.0637
## 9  -2.57574  0.92874  1.50755  4.2961 0.342 3.42e+00 0.4983   *
## 10  0.10792 -0.33816  0.34133  0.3987 1.305 5.38e-02 0.1963
## 11 -0.03427  0.09253 -0.00269  0.2180 1.172 1.62e-02 0.0861
## 12 -0.03027 -0.04867  0.05397 -0.0677 1.291 1.60e-03 0.1137
## 13  0.07237 -0.03562  0.01134  0.0813 1.207 2.29e-03 0.0611
## 14  0.04952 -0.06709  0.06182  0.0974 1.228 3.29e-03 0.0782
## 15  0.02228 -0.00479  0.00684  0.0426 1.192 6.32e-04 0.0411
## 16 -0.00269  0.06442 -0.08419 -0.0972 1.369 3.29e-03 0.1659
## 17  0.02886  0.00649 -0.01570  0.0339 1.219 4.01e-04 0.0594
## 18  0.24856  0.18973 -0.27243  0.3653 1.069 4.40e-02 0.0963
## 19  0.17256  0.02357 -0.09897  0.1862 1.215 1.19e-02 0.0964
## 20  0.16804 -0.21500 -0.09292 -0.6718 0.760 1.32e-01 0.1017
## 21 -0.16193 -0.29718  0.33641 -0.3885 1.238 5.09e-02 0.1653
## 22  0.39857 -1.02541  0.57314 -1.1950 1.398 4.51e-01 0.3916   *
## 23 -0.15985  0.03729 -0.05265 -0.3075 0.890 2.99e-02 0.0413
## 24 -0.11972  0.40462 -0.46545 -0.5711 0.948 1.02e-01 0.1206
## 25 -0.01682  0.00085  0.00559 -0.0176 1.231 1.08e-04 0.0666
```

Chapter - 9: Multicolinearity

- The use and interpretation of a multiple regression model often depends explicitly or implicitly on the estimates of the individual regression coefficients.
- When there are near - linear dependencies among the regressors, the problem of multicollinearity is said to exist.
- Several techniques have been proposed for detecting multicollinearity. We will now discuss and illustrate some of these diagnostic measures.

Multicollinearity Diagnostics

Several techniques have been proposed for detecting multicollinearity. We will now discuss and illustrate some of these diagnostic measures.

- Scatterplot/correlation matrix
- Variance inflation factors (VIFs)
- Condition number of the correlation matrix

Variance Inflation Factors

$$VIF_j = \frac{1}{1 - R_j^2}$$

One or more large VIFs indicate multicollinearity. Practical experience indicates that if any of the VIFs exceeds 5 or 10, it is an indication that the associated regression coefficients are poorly estimated because of multicollinearity.

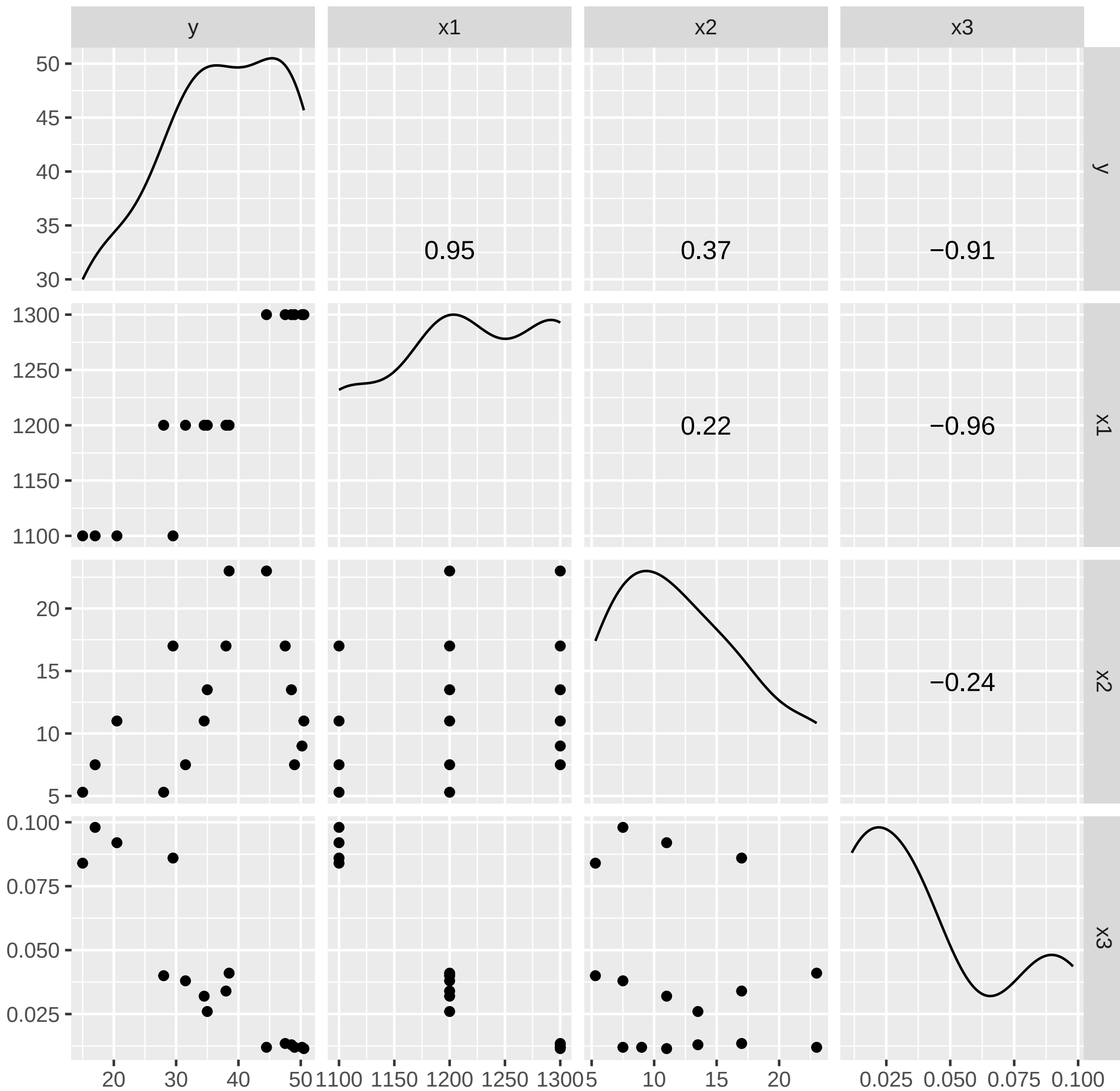
```r
data3 = read.table("D:\\CSUSB\\Fall 2021\\MATH 4360\\RNotes\\ex91.txt", header = TRUE)
head(data3)
```

```
##       y   x1   x2     x3
## 1 49.0 1300  7.5 0.0120
## 2 50.2 1300  9.0 0.0120
## 3 50.5 1300 11.0 0.0115
## 4 48.5 1300 13.5 0.0130
## 5 47.5 1300 17.0 0.0135
## 6 44.5 1300 23.0 0.0120
```

```
names(data3)
```

```
## [1] "y"  "x1" "x2" "x3"
```

```
n = nrow(data3)
GGally::ggscatmat(data3, columns = c("y", "x1", "x2", "x3"))
```




```
Fit3 = lm(y ~ x1 + x2 + x3 + x1:x2 + x1:x3 + x2:x3 + I(x1^2) + I(x2^2) + I(x3^2), data = data3)
summary(Fit3)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x1:x2 + x1:x3 + x2:x3 + I(x1^2) +
##     I(x2^2) + I(x3^2), data = data3)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.3499 -0.3411  0.1297  0.5011  0.6720
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.617e+03  3.136e+03  -1.153  0.29260
## x1           5.324e+00  4.879e+00   1.091  0.31706
## x2           1.924e+01  4.303e+00   4.472  0.00423 **
## x3           1.377e+04  1.045e+04   1.318  0.23572
## I(x1^2)     -1.927e-03  1.896e-03  -1.016  0.34874
## I(x2^2)     -3.034e-02  1.168e-02  -2.597  0.04084 *
## I(x3^2)     -1.158e+04  7.699e+03  -1.504  0.18318
## x1:x2       -1.414e-02  3.212e-03  -4.404  0.00455 **
## x1:x3       -1.058e+01  8.241e+00  -1.283  0.24666
## x2:x3       -2.103e+01  9.241e+00  -2.276  0.06312 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9014 on 6 degrees of freedom
## Multiple R-squared:  0.9977, Adjusted R-squared:  0.9943
## F-statistic: 289.7 on 9 and 6 DF,  p-value: 3.225e-07
```

```
VIF_All =  car::vif(Fit3)
VIF_All
```

```
##           x1           x2           x3      I(x1^2)      I(x2^2)      I(x3^2)
## 2.856749e+06 1.095614e+04 2.017163e+06 2.501945e+06 6.573359e+01 1.266710e+04
##        x1:x2        x1:x3        x2:x3
## 9.802903e+03 1.428092e+06 2.403594e+02
```

```
max(VIF_All)
```

```
## [1] 2856749
```

Condition number of the correlation matrix - Eigensystem Analysis of $X'X$

The condition indices of the $X'X$ matrix are

$$K_j = \frac{\lambda_{max}}{\lambda_j}, \quad j = 1, 2, \dots, p$$

Generally,

- If the condition number is less than 100, there is no serious problem with multicollinearity.
- Condition numbers between 100 and 1000 imply moderate to strong multicollinearity.
- Condition numbers bigger than 1000 indicate severe multicollinearity

```
n = nrow(data3)
X = cbind(data3$x1, data3$x2, data3$x3, data3$x1^2, data3$x2^2, data3$x3^2,
          data3$x1*data3$x2, data3$x1*data3$x3, data3$x2*data3$x3)
Y = data3$y
Xt_X = t(X) %*% X # X'X matrix
ev = eigen(Xt_X)
ev
```

```
## eigen() decomposition
## $values
## [1] 3.543840e+13 6.843691e+08 1.133770e+05 1.156081e+04 1.371600e+03
## [6] 1.793483e+00 1.076084e-02 7.164943e-06 1.849621e-06
##
## $vectors
##                [,1]          [,2]          [,3]          [,4]          [,5]
##  [1,] -8.148094e-04 -7.871574e-04  8.969851e-01  4.680698e-02 -0.4394125924
##  [2,] -8.397435e-06  8.100881e-04  8.743648e-03  2.413187e-03 -0.0089528890
##  [3,] -2.439565e-08 -4.874603e-07  4.054838e-04  1.140686e-05  0.0009012669
##  [4,] -9.999464e-01 -1.032341e-02 -7.482441e-04 -1.508866e-04  0.0003289572
##  [5,] -1.258936e-04  2.308876e-02  4.180254e-02 -9.986334e-01 -0.0211358051
##  [6,] -1.462126e-09 -5.839829e-08  3.831237e-05  7.412505e-06  0.0001527700
##  [7,] -1.032391e-02  9.996793e-01 -4.841350e-05  2.309762e-02  0.0006132590
##  [8,] -2.839847e-05 -5.127936e-04  4.399719e-01 -5.988617e-04  0.8979652118
##  [9,] -2.833037e-07  2.486543e-05  4.213321e-03  8.609170e-04  0.0068451226
##                [,6]          [,7]          [,8]          [,9]
##  [1,] -1.108801e-02 -4.369690e-03  0.000000e+00  0.000000e+00
##  [2,]  9.031693e-01  4.290718e-01  4.367635e-03 -7.291443e-04
##  [3,] -1.160328e-04 -6.978564e-03  5.738794e-01 -8.189095e-01
##  [4,]  9.299151e-06  3.306628e-06  3.673860e-09 -1.542755e-09
##  [5,]  2.014085e-03  4.036014e-05  1.590512e-05 -5.970149e-06
##  [6,] -1.399365e-04 -7.063391e-03  8.188810e-01  5.739198e-01
##  [7,] -7.975254e-04 -3.249414e-04 -3.670913e-06  6.875008e-07
##  [8,]  3.557584e-04  9.033753e-03 -6.045318e-04  7.054369e-04
##  [9,]  4.291357e-01 -9.031600e-01 -8.769610e-03  1.499745e-03
```

```r
max_ev = max(ev$values) # Find maximum
min_ev = min(ev$values) # Find minimum
K_j = max_ev/ev$values
K_j
```

```
## [1] 1.000000e+00 5.178259e+04 3.125714e+08 3.065390e+09 2.583727e+10
## [6] 1.975955e+13 3.293275e+15 4.946083e+18 1.915982e+19
```

```r
## The condition number is
K = max_ev/min_ev
K
```

```
## [1] 1.915982e+19
```

References

- Introduction to Linear Regression Analysis, 5th Edition, by Douglas C. Montgomery, Elizabeth A. Peck, G. Geoffrey Vining (Wiley), ISBN: 978-0-470-54281-1.
- R Core Team (2020). R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing.
- RStudio Team (2020). RStudio: Integrated Development Environment for R. Boston, MA: RStudio, PBC.