Regression 3

Diagnostic Procedures for aptness of model

Consider the simple linear regression model

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i.$$

This model is based on a series of assumptions which may not be met in practice. Departures from the simple linear regression model with normal errors happen when:

1. The regression function is not linear,
   ($\Rightarrow \hat{\epsilon}_i$ vs. $\hat{Y}_i$ plot)
2. The $\mathrm{Var}(\epsilon_i)$ is not constant,
   ($\Rightarrow \hat{\epsilon}_i^2$ vs. $\hat{Y}_i$ plot)
3. The errors ϵ_i are not independent,
   ($\Rightarrow \hat{\epsilon}_i$ vs. time-order plot)
4. The errors ϵ_i are not normally distributed,
   ($\Rightarrow$ histogram, normal probability plot or Q–Q plot)
5. Other important predictor variables have been omitted,
   ($\Rightarrow \hat{\epsilon}_i$ vs. other predictors)
6. The model fits all but one or a few outliers.
   ($\Rightarrow \hat{\epsilon}_i/\sqrt{\mathrm{MSE}}$ vs. $\hat{Y}_i$ plot)

1 Residuals

Direct diagnostic plots for the response Y are ordinarily not so useful in regression analysis because the values of the observations on the response are a function of the predictor X. Instead, diagnostics for the response Y are usually carried out indirectly through an examination of the residuals, $\hat{\epsilon}_i$. The residuals $\hat{\epsilon}_i$ is defined as

$$\hat{\epsilon}_i = Y_i - \hat{Y}_i = Y_i - (\hat{\beta}_0 + \hat{\beta}_1 X_i).$$

They have the following properties:

1. Sample mean: $\bar{\hat{\epsilon}} = \frac{1}{n}\sum \hat{\epsilon}_i = 0$.
2. Sample variance: $\text{MSE} = \frac{1}{n-2}\sum(\hat{\epsilon}_i - \bar{\hat{\epsilon}})^2 = \frac{1}{n-2}\sum(\hat{\epsilon}_i)^2 = \frac{1}{n-2}\text{SSE}$.
   $E(\text{MSE}) = \sigma^2$ (unbiased).
3. The error terms ϵ_i are *iid* $N(0, \sigma^2)$. But the residuals $\hat{\epsilon}_i$ are *not* fully independent because there are two constraints from the normal equation:
   $$\text{(i)} \sum \hat{\epsilon}_i = 0 \quad \text{and} \quad \text{(ii)} \sum X_i \hat{\epsilon}_i = 0.$$
   Let n = sample size and p = the # of parameters in the regression model, for example $p = 2$ for the simple linear regression. If $n \gg p$, then we can usually ignore the dependencies of $\hat{\epsilon}_i$.

2 Diagnostic for residuals

`Minitab` and R offer a convenient informal graphic analysis of residuals. The `plot()` command in R gives four graphics, which are a scatter plot of $\hat{\epsilon}_i$ vs. $\hat{Y}_i$, a norm Q-Q plot, absolute value of standardized $\hat{\epsilon}_i$ vs. $\hat{Y}_i$, and standardized $\hat{\epsilon}_i$ vs. leverages.

The `Minitab` macro command `%resplots` (old version) also gives four graphics, which are a normal probability plot, a time-series plot, a histogram, and a scatter plot of $\hat{\epsilon}_i$ vs. $\hat{Y}_i$.

Example 3.1. Graphic analysis of residuals.

`Minitab`

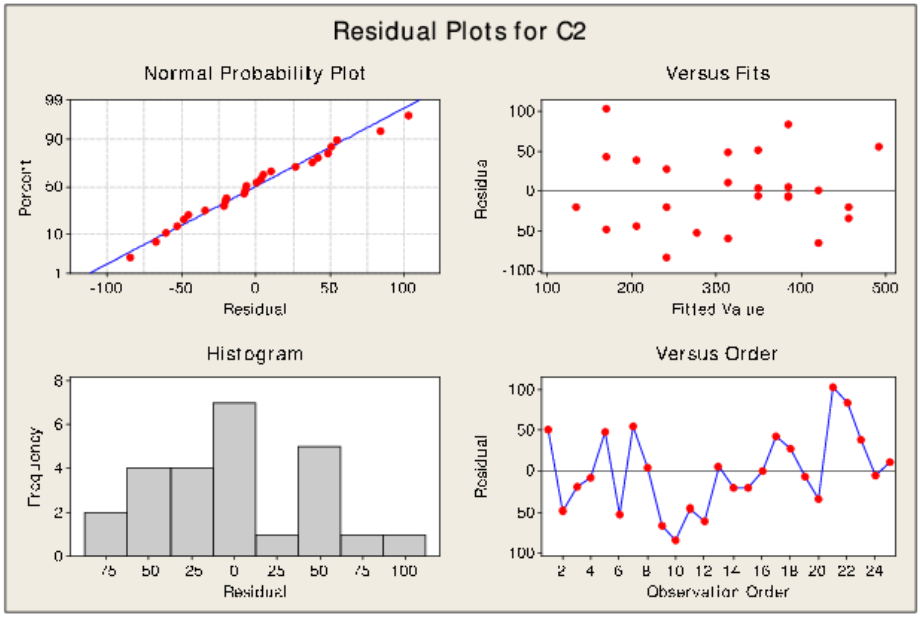
Read Data

```
SUBC> file "S:\LM\CH01TA01.txt" .

Entering data from file: S:\LM\CH01TA01.TXT
25 rows read.
```

The `Minitab` macro command `%resplots` (old version) also gives four graphics, which are a normal probability plot, a time-series plot, a histogram, and a scatter plot of $\hat{\epsilon}_i$ vs. $\hat{Y}_i$.

Resicual Plots

```
## store residuals into c3 and fitted Y into c21
 regr c2 1 c1;
 fits c21;
 resid c3.

## Call resplots.mac (older version)
 %resplots c3 c21.

## New version
## Stat -> Regression -> Graphs... -> (click four in one) -> OK
```



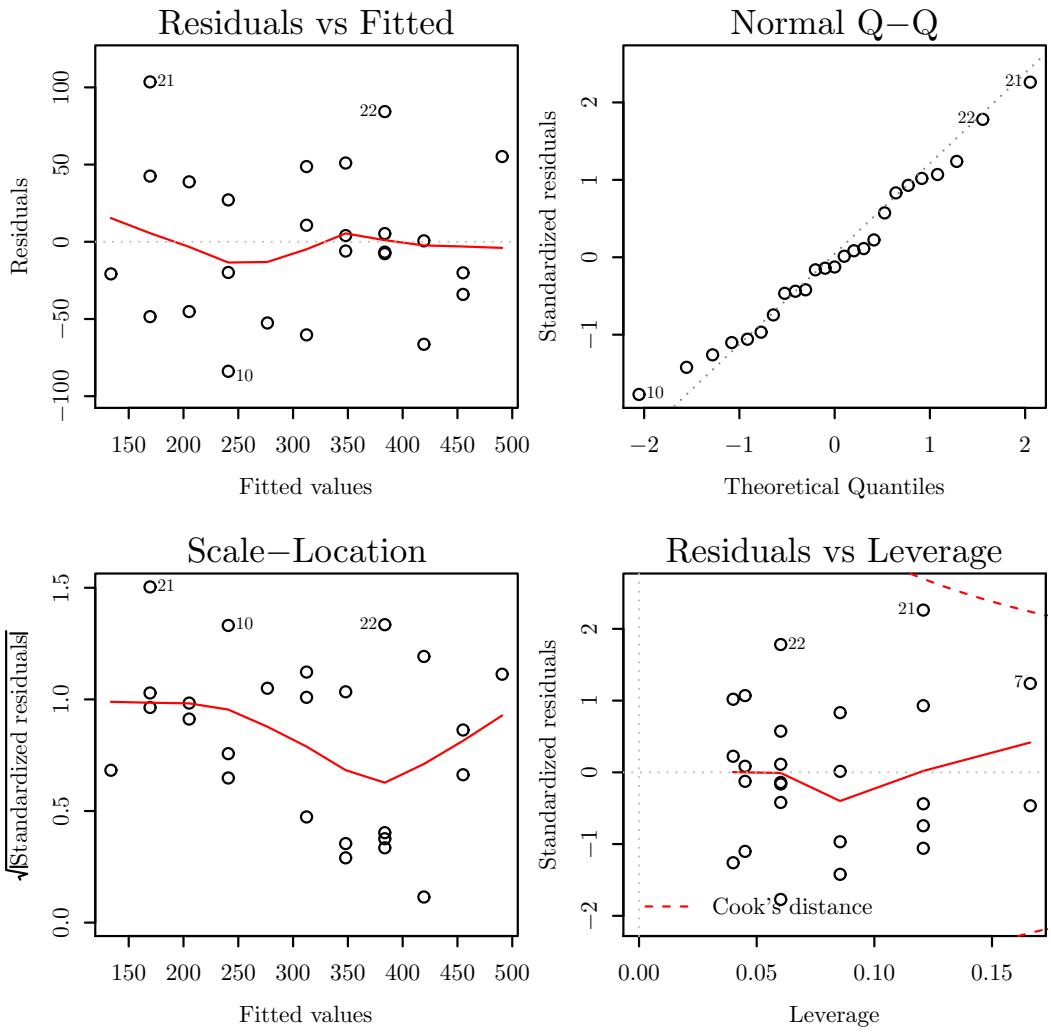
R

Read the data set

```r
# If you have " CH01TA01.txt " in your current computer .
# mydata = read.table("S:\\LM\\CH01TA01.txt")

# If PC is connected to Internet, then the following works.
  mydata =
      read.table("https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt")
```

Residuals Plot

```r
  x = mydata[,1]
  y = mydata[,2]
  LM = lm( y ~ x )

  par ( mfrow=c(2,2) )  ## Put four plots into one sheet
  plot(LM)
```



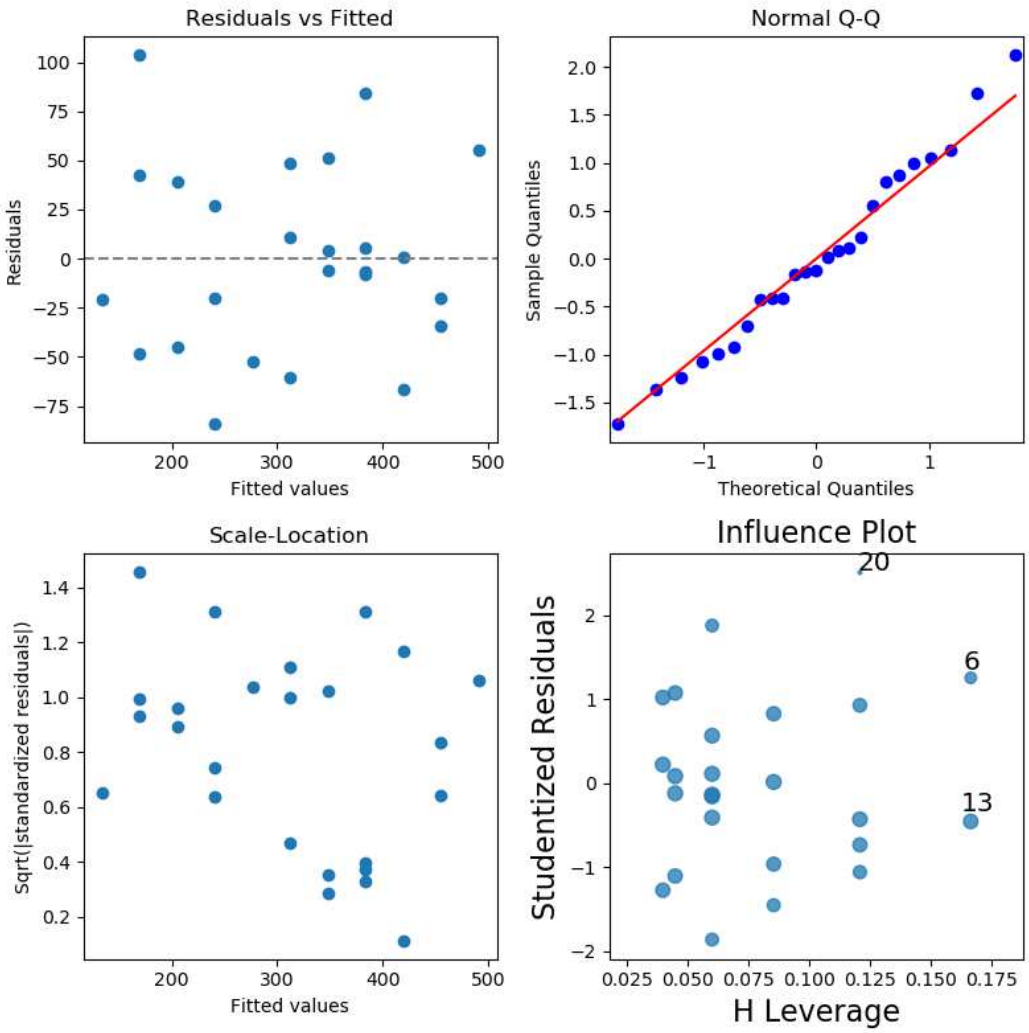
Python

Read the data set

```
#!/usr/bin/python3

# If you have " CH01TA01.txt " in your current computer .
# mydata = open("S:/LM/CH01TA01.txt", "r")
# file = mydata.read().splitlines()

# If PC is connected to Internet, then the following works.
from urllib.request import urlopen
link = "https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt"
url = urlopen(link)
file = url.read().splitlines()
url.close()

x = []
y = []
for line in file:
    p = line.split()
    x.append(float(p[0]))
    y.append(float(p[1]))
```

Residuals Plot

```
import statsmodels.formula.api as smf
import pandas as pd

data1 = pd.DataFrame({"x":x, "y":y})
lm = smf.ols(formula = 'y ~ x', data = data1).fit()

import matplotlib.pyplot as plt
import statsmodels.api as sm

fitted = lm.predict()
resids = lm.resid
std_resids = lm.resid_pearson

# 4 plots in one window
fig = plt.figure(figsize = (8, 8), dpi = 100)

## raw residuals vs. fitted
ax1 = fig.add_subplot(2, 2, 1)
ax1.plot(fitted, resids,  'o')
l = plt.axhline(y = 0, color = 'grey', linestyle = 'dashed')
ax1.set_xlabel('Fitted values')
ax1.set_ylabel('Residuals')
ax1.set_title('Residuals vs Fitted')

## q-q plot
ax2 = fig.add_subplot(2, 2, 2)
sm.qqplot(std_resids, line='s', ax = ax2)
ax2.set_title('Normal Q-Q')

## scale-location
ax3 = fig.add_subplot(2, 2, 3)
ax3.plot(fitted, abs(std_resids)**.5,  'o')
ax3.set_xlabel('Fitted values')
ax3.set_ylabel('Sqrt(|standardized residuals|)')
ax3.set_title('Scale-Location')

## residuals vs. leverage
ax4 = fig.add_subplot(2, 2, 4)
sm.graphics.influence_plot(lm, criterion = 'Cooks', size = 2, ax = ax4)

plt.tight_layout()
fig.savefig('regplots.png')
```


Residuals vs Fitted
Residuals
Fitted values
Normal Q-Q
Sample Quantiles
Theoretical Quantiles
Scale-Location
Sqrt(|standardized residuals|)
Fitted values
Influence Plot
Studentized Residuals
H Leverage
20
6
13

We will study some informal diagnostic plots of residuals to provide information on whether any of the six types of departures from the simple linear regression model are present. Graphic analysis of residuals provides very useful and attractive information. One has to be careful with looking at these plots, however, as sometimes they are difficult to interpret. Unless the effect is very strong, one usually needs a lot of points, say 100 or more, to really notice the effect. Graphic analysis of residuals is inherently subjective. We introduce some informal diagnostic plots of residuals and some objective tests to check departures from the simple linear regression model.

2.1 Non-linearity of regression function

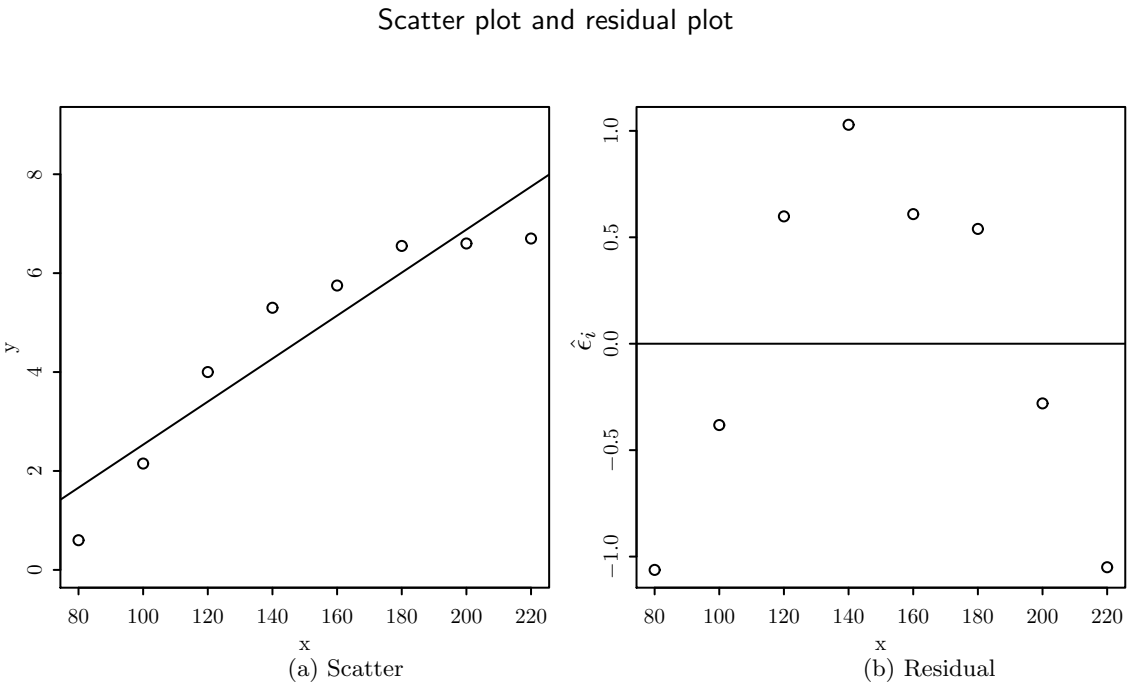
Plot

Nonlinearity of the regression function can be investigated from:

(a) $\hat{\epsilon}_i$ vs. $\hat{Y}_i \Rightarrow$ recommended.

(b) $\hat{\epsilon}_i$ vs. $X_i \Rightarrow$ essentially equivalent to (a).

(c) (X_i, Y_i) scatter plot $\Rightarrow$ not always effective.

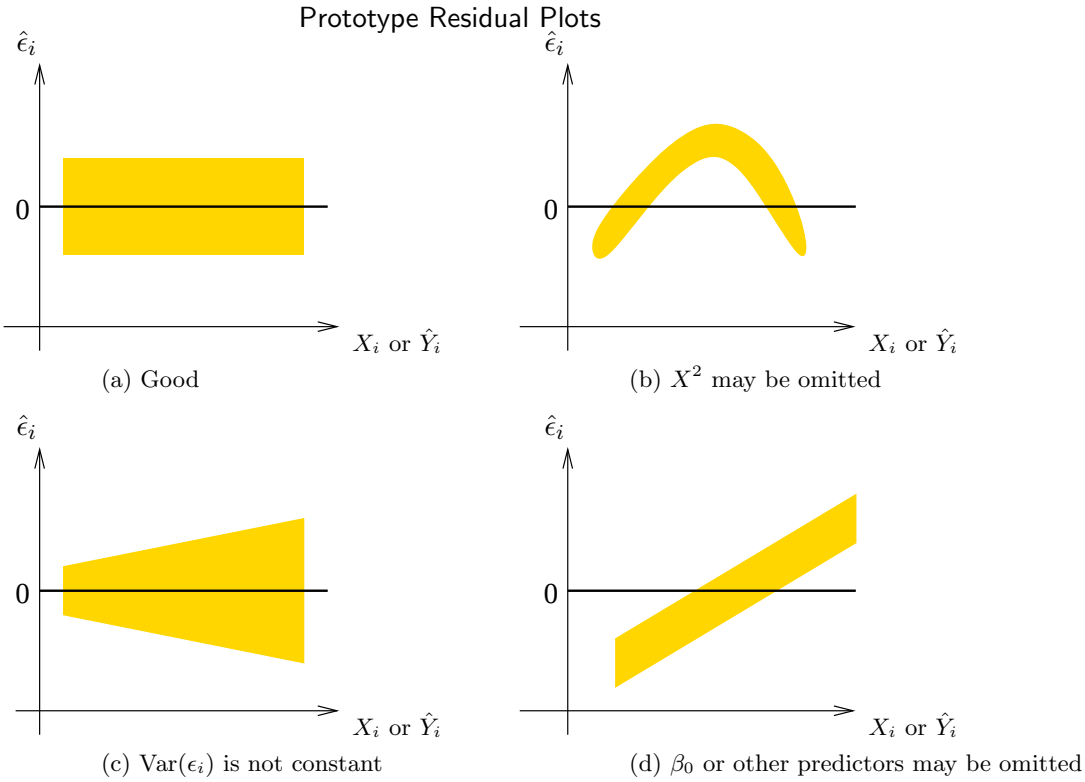


(a) Scatter (b) Residual

The above plots were made by R using the data in Table 3.1 on Page 105 of the text.

R

```r
x = c(80, 220, 140, 120, 180, 100, 200, 160)
y = c(0.60,6.70,5.30,4.00,6.55,2.15,6.60,5.75)

LM = lm(y~x)
y1 = fitted(LM)
r  = y - y1

par(mfrow=c(1,2))   ## two plots into one sheet
plot(x,y, ylim=c(0,9), sub="(a)")
abline(coef(LM))

plot(x,r,  sub="(b)")
abline(h=0)
```



(a) Good (b) X^2 may be omitted

(c) Var(ϵ_i) is not constant (d) β_0 or other predictors may be omitted

Test

Regress $\hat{\epsilon}_i$ on $\hat{Y}_i$ and $\hat{Y}_i^2$:

$$\hat{\epsilon}_i = \gamma_0 + \gamma_1 \hat{Y}_i + \gamma_2 \hat{Y}_i^2.$$

If the coefficient of $\hat{Y}_i^2$ (*i.e.*, γ_2) is significant (usually when p-value is less than $\alpha = 0.05$), then this suggests that the model should include a quadratic term.

Example 3.2.

Minitab

```
## Data set from Table 3.1 pg. 105
MTB > set c1
DATA> 80 220 140 120 180 100 200 160
DATA> end
MTB > set c2
DATA> .60 6.70 5.30 4.0 6.55 2.15 6.6 5.75
DATA> end
MTB > regr c2 1 c1;
SUBC> fits c3;
SUBC> resid c5.

Regression Analysis: C2 versus C1

The regression equation is
C2 = - 1.82 + 0.0435 C1

Predictor       Coef   SE Coef       T       P
Constant     -1.816     1.052   -1.73   0.135
C1         0.043482  0.006706    6.48   0.001

S = 0.869241   R-Sq = 87.5%   R-Sq(adj) = 85.4%

Analysis of Variance
Source           DF      SS      MS      F       P
Regression        1  31.764  31.764  42.04   0.001
Residual Error    6   4.533   0.756
Total             7  36.297

MTB > let c4 = c3**2.
MTB > regr c5 2 c3 c4.

Regression Analysis: C5 versus C3, C4

The regression equation is
C5 = - 3.87 + 2.00 C3 - 0.213 C4

Predictor       Coef  SE Coef        T       P
Constant    -3.8702   0.3896    -9.93   0.000
C3           2.0039   0.1843    10.87   0.000
C4          -0.21290  0.01925   -11.06   0.000

S = 0.188738   R-Sq = 96.1%   R-Sq(adj) = 94.5%

Analysis of Variance
Source           DF      SS      MS      F       P
Regression        2  4.3554  2.1777   61.13   0.000
Residual Error    5  0.1781  0.0356
Total             7  4.5335

Source  DF  Seq SS
C3       1  0.0000
C4       1  4.3554
```

R

```
> x = c(80, 220, 140, 120, 180, 100, 200, 160)
> y = c(0.60, 6.70, 5.30, 4.00, 6.55, 2.15, 6.60, 5.75)

> LM = lm(y~x)
> c3 = fitted(LM)
> c4 = c3^2
> c5 = resid(LM)

> LM2 = lm(c5 ~ c3 + c4)
> summary(LM2)

Call:
lm(formula = c5 ~ c3 + c4)

Residuals:
       1        2        3        4        5        6        7        8
 0.06458  0.07708  0.22351  0.11518  0.05625 -0.22113 -0.11935 -0.19613

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.87017    0.38961  -9.933 0.000177 ***
c3           2.00392    0.18430  10.873 0.000114 ***
c4          -0.21290    0.01925 -11.057 0.000105 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1887 on 5 degrees of freedom
Multiple R-Squared: 0.9607,     Adjusted R-squared: 0.945
F-statistic: 61.13 on 2 and 5 DF,  p-value: 0.0003059
```

Python

```
#!/usr/bin/python3

import pandas as pd
import statsmodels.formula.api as smf

x = [80,  220, 140, 120, 180, 100, 200, 160]
y = [0.60, 6.70, 5.30, 4.00, 6.55, 2.15, 6.60, 5.75]

data1 = pd.DataFrame({"x":x, "y":y})
LM = smf.ols(formula = 'y ~ x', data = data1).fit()
c3 = LM.fittedvalues
c4 = c3 ** 2
c5 = LM.resid #Residuals

data2 = pd.DataFrame({"c3":c3, "c4":c4, "c5":c5})
LM2 = smf.ols(formula = 'c5 ~ c3 + c4', data = data2).fit()
print(LM2.summary())
```

‖

Remark 3.1. **Note**: From the `Minitab` result, the coefficient of $\hat{Y}^2$ (`C4`) is significant, *i.e.*, p-value for two-sided hypothesis test ($H_0 : \gamma_2 = 0$ vs. $H_1 : \gamma_2 \neq 0$) is 0 from the `Minitab` result and $0.000105 \approx 0$ from the R result both of which are less than $\alpha = 0.05$. This result suggests that $\hat{\epsilon}_i$ have a quadratic term and so the model should include a quadratic term. △

2.2 Non-constancy of variance of error

Plot

(i) $\hat{\epsilon}_i^2$ *or* $|\hat{\epsilon}_i|$ vs. $\hat{Y}_i \Rightarrow$ recommended.

(ii) $\hat{\epsilon}_i^2$ *or* $|\hat{\epsilon}_i|$ vs. $X_i \Rightarrow$ essentially equivalent to (a).

Test

(a) Regress $\hat{\epsilon}_i^2$ on $\hat{Y}_i$. If the coefficient of $\hat{Y}_i$ is significant, *i.e.*, p-value is less than $\alpha = 0.05$, then this suggests that variance of error is not constant.

(b) Brown-Forsythe (Modified Levene) Test.

The modified Levene test is the test of the equality of variances of two groups. This test can be used to test the constant error variance. To conduct this test, we divide the data set into two groups, according to the level of X, so that one group consists of cases where the X level is low and the other group consists of cases where the X level is high. We shall use $\hat{\epsilon}_{i1}$ to denote the ith residual for group I and $\hat{\epsilon}_{i2}$ to denote the ith residual for group II. Also we denote n_1 and n_2 to be the sample sizes of the two groups. Denote

$$d_{i1} = |\hat{\epsilon}_{i1} - \tilde{\epsilon}_1| \quad \text{and} \quad d_{i2} = |\hat{\epsilon}_{i2} - \tilde{\epsilon}_2|,$$

where $\tilde{\epsilon}_1 = \text{median}_i(\hat{\epsilon}_{i1})$ and $\tilde{\epsilon}_2 = \text{median}_i(\hat{\epsilon}_{i2})$. Note that the original Levene test uses the mean instead of the median.

The two-sample t test statistic is given as

$$t_L^* = \frac{\bar{d}_1 - \bar{d}_2}{s\sqrt{1/n_1 + 1/n_2}},$$

where $\bar{d}_1$ and $\bar{d}_2$ are the sample means of d_{i1} and d_{i2}, respectively and the pooled variance s^2 is

$$s^2 = \frac{\sum(d_{i1} - \bar{d}_1)^2 + \sum(d_{i2} - \bar{d}_2)^2}{n_1 + n_2 - 2}.$$

The decision rule is:

- If $|t_L^*| \le t(1 - \frac{\alpha}{2}; n_1 + n_2 - 2)$, conclude the error variance is constant.
- If $|t_L^*| > t(1 - \frac{\alpha}{2}; n_1 + n_2 - 2)$, conclude the error variance is not constant.

It should be noted that if the usual ANOVA F statistic for testing equality of means applied to the absolute deviations of k samples $(d_{i_1 1}, d_{i_2 2}, \ldots, d_{i_k k})$, we can perform the homogeneity of variances of k populations.

(c) Breusch-Pagan Test.

This test assumes that the error terms are independent and normally distributed and the variance of the error term ϵ_i, denoted by σ_i^2 is related to the levels of X in the following way:

$$\ln \sigma_i^2 = \gamma_0 + \gamma_1 X_i.$$

The test of $H_0 : \gamma_1 = 0$ is carried out by means of regressing the squared residuals $\hat{\epsilon}_i^2$ on X_i in the usual manner and obtaining the regression sum of squares SSR^*. The test statistic X^2_{BP} is as follows:

$$\text{X}^2_{\text{BP}} = \frac{\text{SSR}^*}{2} \div \left(\frac{\text{SSE}}{n}\right)^2 \dot{\sim} \chi^2_{\text{df}=p-1},$$

where

SSR^* is the regression sum of squares when regressing $\hat{\epsilon}_i^2$ on X_i

and

SSE is the error sum of squares when regressing Y_i on X_i.

The test statistic X^2_{BP} follows approximately the χ^2 distribution with $p-1$ degree of freedom (p is the number of parameters. So $p = 2$.) Large values of X^2_{BP} lead to H_1: non-constancy of error variance.

(d) Other tests of homogeneity of variances.

In general, F-test is used for comparing two variances, where the test statistic given by the ratio of two sample variances. For the modified Levene test, we divided the data set into two groups, according to the level of X, so that one group consists of cases where the X level is low and the other group consists of cases where the X level is high. We denoted the residuals for group I by $\hat{\epsilon}_{i1}$ and the residuals for group II by $\hat{\epsilon}_{j2}$, where $i = 1, 2, \ldots, n_1$ and $j = 1, 2, \ldots, n_2$. Thus, using two samples for groups I and II, we can easily perform the F-test and its test statistic is given by

$$F = \frac{S_1^2}{S_2^2} \sim F(n_1 - 1, n_2 - 1),$$

where S_1^2 is the sample variance of the first sample and S_1^2 is the sample variance of the second. This is easily performed by using the R function, `var.test`. It should

be noted that this test is very sensitive to the departure from the normality assumption. For robust alternative to this test, one can refer to the Ansari-Bradley Test (Hollander and Wolfe, 2013) and the R has the function `ansari.test` for this test.

If there are more than k populations, the above tests can not be applied. For homoscedasticity or homogeneity of variances of k populations, one can refer to Brown-Forsythe test, Bartlett, Fligner-Killeen, Hartley's F-max test, and Cochran's C test. The R program provides `bartlett.test` and `fligner.test`. Note that Bartlett, Hartley's F-max and Cochran's C tests are sensitive to departure from normality.

Example 3.3. (a) **Regress $\hat{\epsilon}_i^2$ on $\hat{Y}_i$.**

Minitab

```
MTB >  READ C1 C2;
SUBC>  file "S:\LM\CH01TA01.txt" .
Entering data from file: S:\LM\CH01TA01.TXT
25 rows read.
MTB > regr c2 1 c1;
SUBC> resid c3;
SUBC> fits  c4.

MTB > let c5 = c3**2
MTB > regr c5 1 c4.

Regression Analysis: C5 versus C4

The regression equation is
C5 = 3940 - 5.59 C4

Predictor     Coef  SE Coef      T      P
Constant      3940     1756   2.24  0.035
C4          -5.593    5.354  -1.04  0.307

S = 2689.72   R-Sq = 4.5%   R-Sq(adj) = 0.4%

Analysis of Variance
Source           DF          SS        MS     F      P
Regression        1     7896142   7896142  1.09  0.307
Residual Error   23   166395896   7234604
Total            24   174292038

Unusual Observations
Obs    C4      C5    Fit  SE Fit  Residual  St Resid
 21   169   10718   2992     935      7726      3.06R
 22   384    7109   1794     660      5316      2.04R

R denotes an observation with a large standardized residual.
```

R

Read the data set

```
# If you have "CH01TA01.txt" in your current computer.
> mydata = read.table("S:\\LM\\CH01TA01.txt")

# If your computer is connected to Internet
```

```
> mydata =
    read.table("https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt")
```

Regression analysis

```
> c1 = mydata[,1]
> c2 = mydata[,2]
> LM = lm(c2 ~ c1)

> c3 = resid(LM)
> c4 = fitted(LM)
> c5 = c3^2

> LM2 = lm(c5 ~ c4)
> summary(LM2)

Call:
lm(formula = c5 ~ c4)

Residuals:
    Min      1Q  Median      3Q     Max
-2760.1 -1765.4  -990.7   609.5  7726.3

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 3939.750   1756.371   2.243   0.0348 *
c4            -5.593      5.354  -1.045   0.3070
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2690 on 23 degrees of freedom
Multiple R-Squared: 0.0453,     Adjusted R-squared: 0.003796
F-statistic: 1.091 on 1 and 23 DF,  p-value: 0.307
```

Python

Read the data set

```
#!/usr/bin/python3

# If you have " CH01TA01.txt " in your current computer .
# mydata = open("S:/LM/CH01TA01.txt", "r")
# file = mydata.read().splitlines()

# If PC is connected to Internet, then the following works.
from urllib.request import urlopen
link = "https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt"
url = urlopen(link)
file = url.read().splitlines()
url.close()

c1 = []
c2 = []
for line in file:
    p = line.split()
    c1.append(float(p[0]))
    c2.append(float(p[1]))
```

Regression analysis

```
import pandas as pd
import statsmodels.formula.api as smf

data1 = pd.DataFrame({"c1":c1, "c2":c2})
LM = smf.ols(formula = 'c2 ~ c1', data = data1).fit()

c3 = LM.resid
```

```
c4 = LM.fittedvalues
c5 = c3 ** 2

data2 = pd.DataFrame({"c4":c4, "c5":c5})
LM2 = smf.ols(formula = 'c5 ~ c4', data = data2).fit()
print(LM2.summary())
```

‖

Example 3.4. (b) **Modified Levene Test.**

Minitab The Minitab macro for the Levene test (file: levene.MAC) is available at

https://github.com/AppliedStat/LM

```
MTB >  READ C1 C2;
SUBC>  file "S:\LM\CH01TA01.txt" .
Entering data from file: S:\LM\CH01TA01.TXT
25 rows read.

MTB > regr c2 1 c1;
SUBC> resid c3.

MTB > sort c1 c3 c4 c5;
SUBC> by c1.
MTB > print c1 c3 c4 c5

Data Display
Row    C1        C3    C4        C5
  1    80    51.018    20   -20.770
  2    30   -48.472    30   -48.472
  3    50   -19.876    30    42.528
  ..............................

MTB > copy c5 c11
MTB > copy c5 c12
MTB > delete 14:25 c11
MTB > delete  1:13 c12

MTB > %S:\LM\levene c11 c12 k1
Executing from file: S:\LM\levene.MAC

Data Display

K1     1.31648

MTB > invcdf 0.975;
SUBC>  t 23.

Inverse Cumulative Distribution Function

Student's t distribution with 23 DF

P( X <= x )         x
      0.975   2.06866
```

Remark 3.2. From the Minitab result above, we have $|t_L^*| = 1.31648 < 2.06866$. Hence we conclude that the error variance is constant. △

R The R function for the Levene test (file: `levene.R`) is available at

https://github.com/AppliedStat/LM

```
> # First, save the levene.R file at your current directory.
> # source("S:/LM/levene.R")
>
> # If your PC is connected to Internet, the following will work:
> source("https://raw.githubusercontent.com/AppliedStat/LM/master/levene.R")
>
> mydata =
    read.table("https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt")
>
> c1 = mydata[,1]
> c2 = mydata[,2]
>
> LM = lm(c2 ~ c1)
>
> c3 = resid(LM)
>
> c1.order = order(c1)
> c4 = c1 [ c1.order ]
> c5 = c3 [ c1.order ]
>
> print( cbind(c1,c3,c4,c5) )
    c1          c3  c4          c5
1   80  51.0179798  20 -20.7698990
2   30 -48.4719192  30 -48.4719192
3   50 -19.8759596  30  42.5280808
4   90  -7.6840404  30 103.5280808
    ................
>
> gr1 = c5[1:13]
> gr2 = c5[14:25]
>
> levene.test(gr1, gr2)
$t.test.stat
[1] 1.316482

$df
[1] 23

$p.value
[1] 0.2009812
```

Python

```
#!/usr/bin/python3

# If you have " CH01TA01.txt " in your current computer .
# mydata = open("S:/LM/CH01TA01.txt", "r")
# file = mydata.read().splitlines()

# If PC is connected to Internet, then the following works.
from urllib.request import urlopen
link = "https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt"
url = urlopen(link)
file = url.read().splitlines()
url.close()

c1 = []
c2 = []
for line in file:
```

```
    p = line.split()
    c1.append(int(p[0]))
    c2.append(int(p[1]))

import numpy as np
import pandas as pd
import scipy.stats as stats
import statsmodels.formula.api as smf
data1 = pd.DataFrame({"c1":c1, "c2":c2})
LM = smf.ols(formula = 'c2 ~ c1', data = data1).fit()

c3 = LM.resid

c1 = np.asarray(c1)
c3 = np.asarray(c3)
c1_order = np.argsort(c1, kind = 'stable')
c4 = c1[c1_order]
c5 = c3[c1_order]

# cbind = np.vstack((c1, c3, c4, c5))
cbind = pd.DataFrame({"c1":c1, "c2":c2, "c3":c3, "c4":c4})
print(cbind)

gr1 = c5[0:13]
gr2 = c5[13:25]

w, p = stats.levene(gr1, gr2)
print(p)
```

‖

Remark 3.3. Note that the t-distribution critical value for the modified Levene's test at the significance level α with `df` can be found in R as follows:

$$> \texttt{qt(1 - } \alpha\texttt{/2, df)}.$$

To test with $\alpha = 0.05$ and 23 degrees of freedom, we use `qt(1-0.05/2, df=23)` which results in `2.068658`. △

Example 3.5. (c) **Breusch-Pagan Test.**

Minitab The Minitab macro for the Breusch-Pagan test (file: BPtest.MAC) is available at

https://github.com/AppliedStat/LM

```
MTB >#  1. Read the data
MTB >  READ C1 C2;
SUBC>       file "S:\LM\CH01TA01.TXT" .
Entering data from file: S:\LM\CH01TA01.TXT
25 rows read.

MTB > # 2. RUN BPtest Macro
MTB >  %S:\LM\BPtest C2 C1 .
Executing from file: S:\LM\BPtest.MAC

Data Display

Breusch-Pagan Test Statistic:      0.82092
Degrees of Freedom:                1
p-value:                           0.36491
```

R

```
> mydata =
    read.table("https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt")
>
> c1 = mydata[,1]
> c2 = mydata[,2]
>
> LM = lm(c2 ~ c1)
>
> e  = resid(LM)
>
> sigma2 = e^2
>
> LM2 = lm ( sigma2 ~ c1 )
>
> SSR.star = sum( (fitted(LM2)-mean(sigma2))^2 )
>
> SSE = sum( (fitted(LM)-c2)^2 )
>
> n = length(c2)
>
> cbind(SSR.star, SSE, n)
     SSR.star      SSE  n
[1,]  7896142 54825.46 25
>
> X.BP = SSR.star/2 / ( (SSE/n)^2 )
>
> X.BP
[1] 0.8209192
>
> qchisq(0.95, df = 1) ## chi-square critical value
[1] 3.841459
```

Python

```python
#!/usr/bin/python3

# If you have " CH01TA01.txt " in your current computer .
# mydata = open("S:/LM/CH01TA01.txt", "r")
# file = mydata.read().splitlines()

# If PC is connected to Internet, then the following works.
from urllib.request import urlopen
link = "https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt"
url = urlopen(link)
file = url.read().splitlines()
url.close()

c1 = []
c2 = []
for line in file:
    p = line.split()
    c1.append(int(p[0]))
    c2.append(int(p[1]))

import numpy as np
import pandas as pd
import scipy.stats as stats
import statsmodels.formula.api as smf

data1 = pd.DataFrame({"c1":c1, "c2":c2})
LM = smf.ols(formula = 'c2 ~ c1', data = data1).fit()

e = LM.resid
sigma2 = e**2
data2 = pd.DataFrame({"c1":c1, "sigma2":sigma2})
LM2 = smf.ols(formula = 'sigma2 ~ c1', data = data2).fit()
SSR_star = sum((LM2.fittedvalues-np.mean(sigma2))**2)
SSE = sum((LM.fittedvalues-c2)**2)
n = len(c2)

cbind = pd.DataFrame({"SSR_star":SSR_star, "SSE":SSE, "n":n}, index = [0])
print('\nData Display of SSR_star,SSE and n:')
print(cbind)

X_BP = SSR_star/2/((SSE/n)**2)
print('\nX_BP:')
print(X_BP)

qchisq = stats.chi2.ppf(0.95, df=1)
print('\n qchisq:')
print(qchisq)
```

Remark 3.4. Note that the χ^2 critical value for the Breusch-Pagan test at the significance level α with `df` can be found in R as follows:

$$> \texttt{qchisq(1 - } \alpha\texttt{, df)}.$$

Since $X^2_{\text{BP}} = 0.8209192$ is less than $\chi^2(0.95; 1) = 3.841459$, we conclude H_0: constant error variance.

△

The R function for the Breusch-Pagan test (file: `Breusch-Pagan.R`) is also available at

https://github.com/AppliedStat/LM

```
> source("https://raw.githubusercontent.com/AppliedStat/LM/master/Breusch-Pagan.R")

> BP.test ( c2 ~ c1)
$test.stat
[1] 0.8209192

$df
[1] 1

$p.value
[1] 0.3649116
```

2.3 Presence of outliers

Outliers are extreme observations.

Plot

It is convenient to use *semi-Studentized residuals* which are defined as

$$\hat{\epsilon}_i^* = \frac{\hat{\epsilon}_i - \bar{\hat{\epsilon}}}{\sqrt{\text{MSE}}} = \frac{\hat{\epsilon}_i}{\sqrt{\text{MSE}}},$$

since they do not depend on the unit of Y.

(a) $\hat{\epsilon}_i^*$ vs. $\hat{Y}_i$. A rule of thumb is to identify Y_i as outliers if $|\hat{\epsilon}_i^*| > 4$

(b) Use the *Studentized deleted residuals* defined as

$$t_i = \frac{\hat{\epsilon}_i}{\sqrt{\text{MSE}_{(i)}(1 - h_{ii})}}.$$

It is better than (a). We will study this later.

(c) Residuals can also be identified from box plot, stem-and-leaf plot, and histogram or dot plot.

- `Minitab`: Use `BOXPLOT`, `STEM-AND-LEAF`, `HISTOGRAM`.
- R: Use `boxplot( )`, `stem( )`, `hist( )`.

Test

We will discuss this later (textbook: chapter 9).

2.4 Non-independence of error terms

Plot

$\hat{\epsilon}_i$ vs. time-order (sequence) plot.

Test

Runs test and Durbin-Watson test are frequently used to test for lack of randomness in the residuals arranged in time or sequence order. We will discuss this later (textbook: chapter 12).

2.5 Non-normality of error terms

Plot

(a) Normal probability plot ($\hat{\epsilon}_{[k]}$ vs. $E(\hat{\epsilon}_{[k]})$) or Q-Q plot $\Rightarrow$ recommended.

Normal probability plot of the residuals is a plot of $\hat{\epsilon}_{[k]}$ vs. $E(\hat{\epsilon}_{[k]})$, where $\hat{\epsilon}_{[k]}$ is the k-th smallest among the n residuals. A good approximation of $E(\hat{\epsilon}_{[k]})$ is

$$E(\hat{\epsilon}_{[k]}) \approx \sqrt{\text{MSE}}\,\Phi^{-1}\Big(\frac{k-0.375}{n+0.25}\Big),$$

where $\Phi^{-1}(\cdot)$ is the inverse cdf of $N(0,1)$.

Note: the R function `qqnorm()` gives the Q-Q plot.

(b) Distribution plots such as box plot, histogram, dot plot, stem-and-leaf.

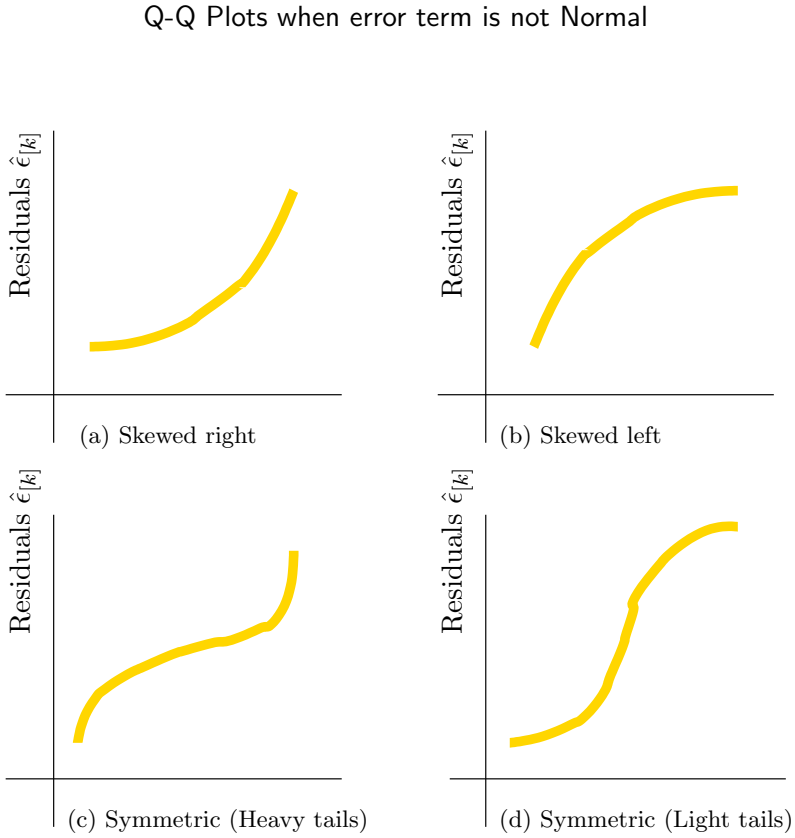
(c) Comparison of frequencies:

$100(1-\alpha)\%$ of the residuals $\hat{\epsilon}_i$ fall between $\pm\sqrt{\text{MSE}}\cdot t(1-\frac{\alpha}{2}; \text{df}=n-p)$, where p is the number of parameters.

Test

Calculate the sample correlation coefficient between $\hat{\epsilon}_{[k]}$ and $E(\hat{\epsilon}_{[k]}) \approx \sqrt{\text{MSE}}\,\Phi^{-1}\Big(\frac{k-0.375}{n+0.25}\Big)$. Find the critical value for n with α from Table B.6 of the textbook or Table 2 of Looney and Gulledge (1985) in *The American Statistician* 39, pp. 75–79. If the sample correlation coefficient is at least as large as the critical value from Table B.6 (textbook) or Table 2 (Looney and Gulledge), then one can conclude that the error terms are reasonably normally distributed.

Remark 3.5. Personally, I am more concerned with getting the variance constant. And residuals may appear to be not normal because an inappropriate regression function is used or because the variance of error terms is not constant. △

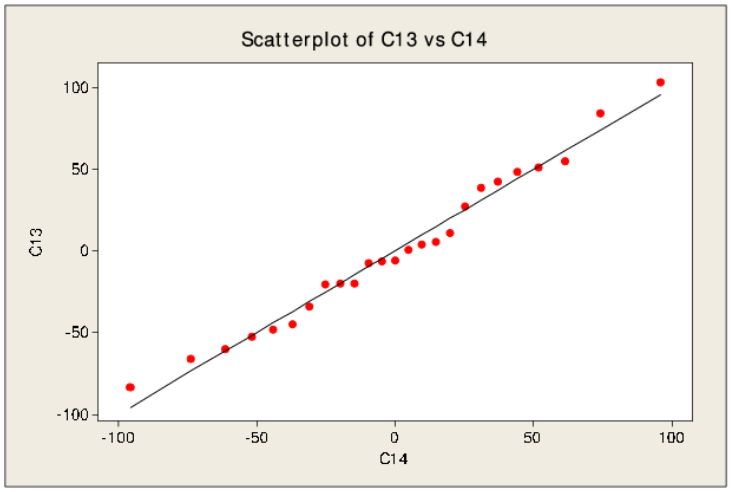


Example 3.6. Normal Probability Plot (based on the textbook).

Note that the normal probability plot using the `Minitab` built-in function, `%resplot`, uses the horizontal axis for the residuals.

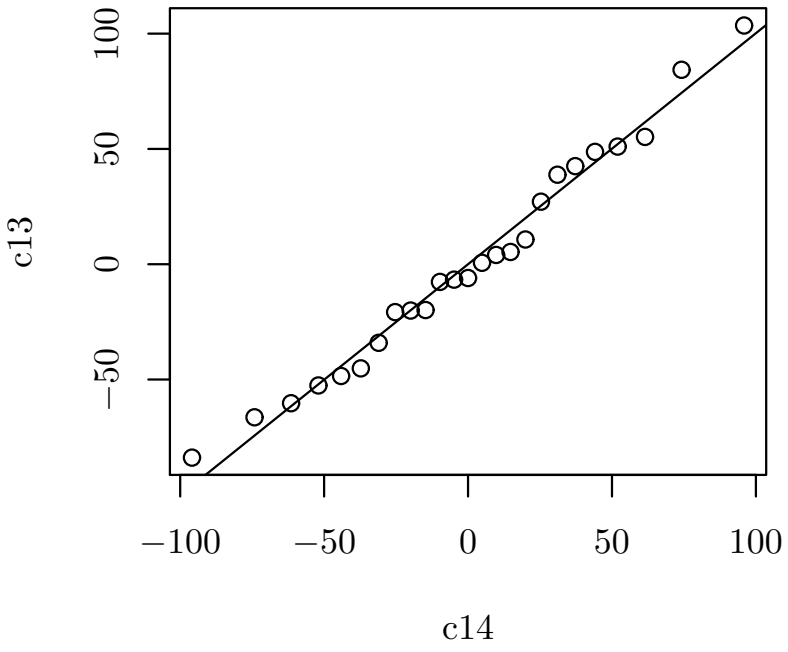
`Minitab`

```
MTB > READ C1 C2;
SUBC>    file "S:\LM\CH01TA01.txt" .
Entering data from file: S:\LM\CH01TA01.TXT
25 rows read.

MTB > regr c2 1 c1;
SUBC> resid c3;
SUBC> mse   k2.

MTB >let k1 = count(c3)

MTB > set c10
DATA>     1:k1
DATA> end .

MTB >sort c3 c13
MTB > let c11 = (c10 - .375) / (k1+.25)

MTB >invcdf c11 c12;
SUBC> normal 0 1.
MTB >let c14 = sqrt(k2)*c12
MTB >plot c13*c14 ;
SUBC> line c14 c14 .
```



R

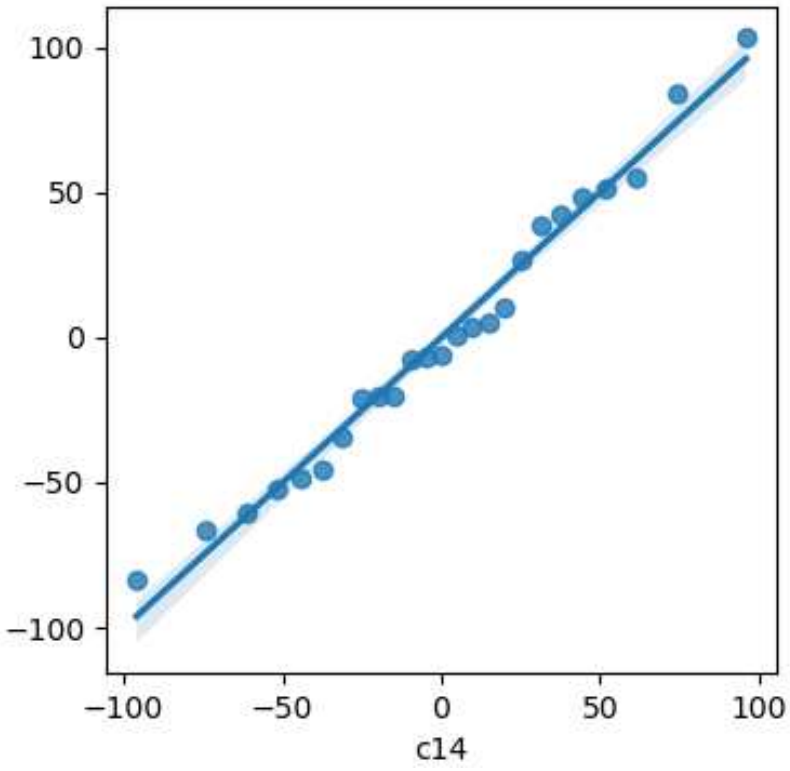
```
> mydata =
    read.table("https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt")

> c1 = mydata[,1]
> c2 = mydata[,2]
>
> LM = lm(c2 ~ c1)
>
> c3 = resid(LM)
>
> LM.sum = summary(LM)
>
> attributes(LM.sum)
$names
 [1] "call"          "terms"         "residuals"     "coefficients"
 [5] "aliased"       "sigma"         "df"            "r.squared"
 [9] "adj.r.squared" "fstatistic"    "cov.unscaled"

$class
[1] "summary.lm"

> s = LM.sum[[ "sigma" ]]
>
> c13 = sort(c3)
>
> n = length(c3)
>
> k = 1:n
>
> c11 = (k - 0.375) / (n+0.25)
>
> c12 = qnorm( c11 )
>
> c14 = s * c12
>
> postscript( "ex4a.ps", width=4, height=4)
>
> plot ( c14, c13)
```




```
> abline(lm(c13~c14))
```

Python

```
#!/usr/bin/python3

# If you have " CH01TA01.txt " in your current computer .
# mydata = open("S:/LM/CH01TA01.txt", "r")
# file = mydata.read().splitlines()

# If PC is connected to Internet, then the following works.
from urllib.request import urlopen
link = "https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt"
url = urlopen(link)
file = url.read().splitlines()
url.close()

c1 = []
c2 = []
for line in file:
    p = line.split()
    c1.append(int(p[0]))
    c2.append(int(p[1]))

import seaborn as sns
# The below may be needed for install seaborn and pandas (numexpr), etc
# sudo apt-get update -y
# sudo apt-get install -y python3-seaborn
# sudo apt-get install -y python3-pandas
# sudo apt-get install -y python3-numexpr   # latest numexpr is needed for pandas
import numpy as np
```




```
import scipy.stats as stats
import pandas as pd

import statsmodels.formula.api as smf
import matplotlib.pyplot as plt

data1 = pd.DataFrame({"c1":c1, "c2":c2})
LM = smf.ols(formula = 'c2 ~ c1', data = data1).fit()

c3 = LM.resid
LM_sum = LM.summary()


# get the residual standard error
squared_sigma = LM.mse_resid
s = squared_sigma ** 0.5

c13 = np.sort(c3)
n = len(c3)

c14 = []
for k in range(1,n+1):
    c11 = (k - 0.375)/(n + 0.25)
    c12 = stats.norm.ppf(c11)
    c14.append(s * c12)

c14 = np.asarray(c14)

fig = plt.figure(figsize = (4,4), dpi = 100)
data2 = pd.DataFrame({"c13":c13, "c14":c14})
sns.regplot(x="c14",y="c13",data=data2)
#sns.lmplot(x="c14",y="c13",data=data2)
plt.savefig('ex4a.png')
```

Example 3.7. Correlation test for normality.

Minitab

```
MTB > READ C1 C2;
SUBC>    file "S:\LM\CH01TA01.txt" .
Entering data from file: S:\LM\CH01TA01.TXT
25 rows read.

MTB >regr c2 1 c1;
SUBC> resid c3;
SUBC> mse   k2.

MTB > let k1 = count(c3)

MTB > set c10
DATA>     1:k1
DATA> end .

MTB > sort c3 c13

MTB > let c11 = (c10 - .375) / (k1+.25)

MTB >invcdf c11 c12;
SUBC> normal 0 1.

MTB >let c14 = sqrt(k2)*c12

MTB > correlation c13 c14 .

Correlations: C13, C14

Pearson correlation of C13 and C14 = 0.992
P-Value = 0.000
```

Don't use the above p-value for the normality test.

R

```
> # Using Normal Probability Plot
> mydata =
    read.table("https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt")
> c1 = mydata[,1]
> c2 = mydata[,2]
> LM = lm(c2 ~ c1)
> c3 = resid(LM)
> LM.sum = summary(LM)
> s = LM.sum[[ "sigma" ]]
> c13 = sort(c3)
> n = length(c3)
> k = 1:n
> c11 = (k - 0.375) / (n+0.25)
> c12 = qnorm( c11 )
> c14 = s * c12
>
> cor(c13, c14)
[1] 0.9915055

>
> ## Using Q-Q plot
```

```
> pp = ppoints(c13, a=3/8)    # option "a=3/8=0.375" gives (k-0.375)/(n+0.25)
> qq = qnorm(pp)
> cor(c13, qq)
[1] 0.9915055
```

Python

```
#!/usr/bin/python3

# If you have " CH01TA01.txt " in your current computer .
# mydata = open("S:/LM/CH01TA01.txt", "r")
# file = mydata.read().splitlines()


# If PC is connected to Internet, then the following works.
from urllib.request import urlopen
link = "https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt"
url = urlopen(link)
file = url.read().splitlines()
url.close()

c1 = []
c2 = []
for line in file:
    p = line.split()
    c1.append(int(p[0]))
    c2.append(int(p[1]))

import numpy as np
import scipy.stats as stats
import pandas as pd
import statsmodels.formula.api as smf

data1 = pd.DataFrame({"c1":c1, "c2":c2})
LM = smf.ols(formula = 'c2 ~ c1', data = data1).fit()

c3 = LM.resid
LM_sum = LM.summary()

squared_sigma = LM.mse_resid
s = squared_sigma ** 0.5

c13 = np.sort(c3)
n = len(c3)

c11 = [(k - 0.375)/(n + 0.25) for k in range(1,n+1)]

c12 = stats.norm.ppf(c11)
c14 = s * c12

cor1 = np.corrcoef(c13, c14)
print(cor1)

a = 3/8
m = len(c13)


# pp is written by ourselves, equivalent to ppints in R
pp = [(i-a)/(m+1/4) for i in range(1, m+1)]

qq = stats.norm.ppf(pp)
cor2 = np.corrcoef(c13, qq)
print(cor2)
```

From Table B.6 of the textbook or Table 2 of Looney and Gulledge (1985), the critical value for $n = 25$ and $\alpha = 0.05$ is 0.959. Since the sample correlation coefficient exceeds this critical value, we can conclude that the distribution the error terms does not depart from a normal distribution.

‖

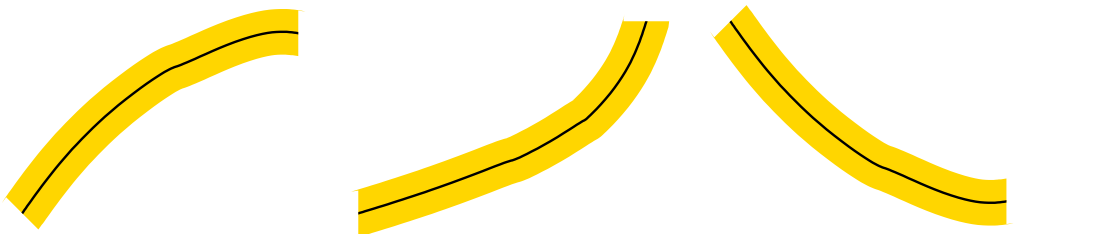
3 Transformations

We consider transformations of the predictor X and the response Y.

3.1 Transformations of the predictor X

If the distribution of the error terms is reasonably close to *normal distribution* and the error terms have approximately *constant variance*, the transformations on X should be attempted. The transformations on Y is not desirable because the transformations on Y may change the shape of the distribution of the error terms from the normal distribution and may also leads to substantially differing error term variances.

Prototype Nonlinear regression pattern with constant error variance.

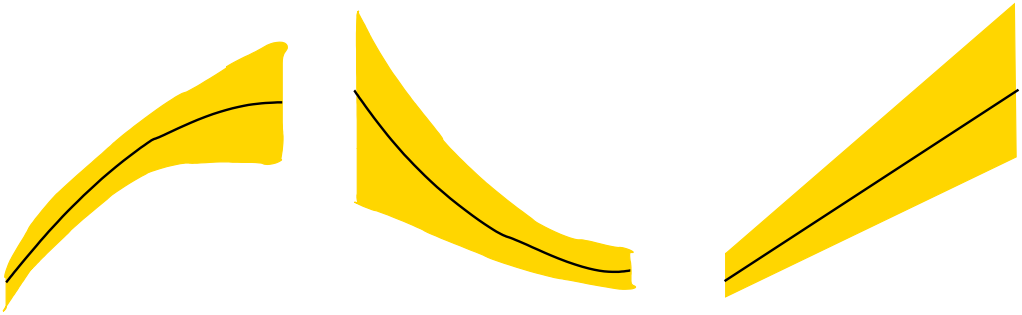


(a) $X' = \log_{10} X$, $X' = \sqrt{X}$ (b) $X' = X^2$, $X' = \exp(X)$ (c) $X' = 1/X$, $X' = \exp(-X)$

3.2 Transformations of the response Y

Non-constant error variances and non-normality of the error terms frequently appear together. To remedy these departures from the simple linear regression model, we need a transformation of Y. We recommend the following transformations.

Prototype Nonlinear regression pattern with non-constant error variance.



Possible transformations: $Y' = \sqrt{Y}$, $Y' = \log_{10} Y$, or $Y' = 1/Y$.

3.3 Box-Cox transformations

It is often difficult to determine from diagnostic plots which transformation of Y is most appropriate for correcting unequal error variances and nonlinearity of the regression function. The Box-Cox procedure automatically identifies a transformation from the family of power transformations on Y. The family of power transformations is of the form

$$Y' = \frac{Y^\lambda - 1}{\lambda} \approx Y^\lambda,$$

for some λ and if $\lambda = 0$, use $Y' = \ln Y$. The normal error regression model of the above power transformations become

$$Y_i^\lambda = \beta_0 + \beta_1 X_i + \epsilon_i.$$

The Box-Cox procedure shows how to estimate $\hat{\lambda}$, the maximum likelihood estimate of λ to use in the power transformation. Note that the textbook use λ which has the smallest SSE.

Example 3.8. Box-Cox Transforms

The `Minitab` macro for the Box-Cox Transform (`bxcx.MAC`) is available at

https://github.com/AppliedStat/LM

`Minitab`

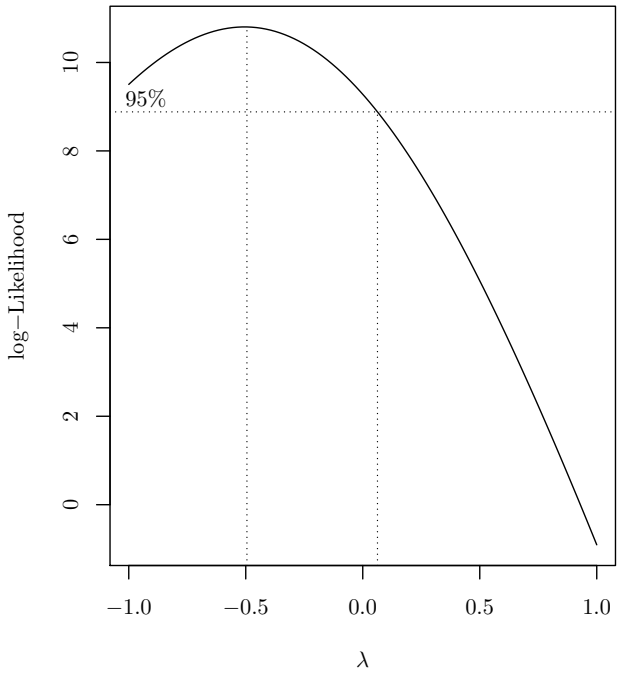
```
## See Table 3.9 (Section 3.9)
MTB > READ c1 c2 c3;
SUBC>   file "S:\LM\CH03TA08.txt" .
Entering data from file: S:\LM\CH03TA08.TXT
25 rows read.

## Generate 1.0, 0.9, ..., -1.0
MTB >set c3
DATA>   10:-10
DATA> end

MTB >let c3 = c3/10

MTB >%S:\LM\bxcx c2 c1 c3 c4
Executing from file: S:\LM\bxcx.MAC

MTB >print c3 c4
Data Display
Row     C3        C4
  1    1.0   77.9831
  2    0.9   70.3505
  3    0.8   63.6693
  4    0.7   57.8369
  5    0.6   52.7634
  6    0.5   48.3707
  7    0.4   44.5905
  8    0.3   41.3634
  9    0.2   38.6379
 10    0.1   36.3694
 11    0.0   34.5195
 12   -0.1   33.0552
 13   -0.2   31.9487
 14   -0.3   31.1763
 15   -0.4   30.7186
 16   -0.5   30.5596
 17   -0.6   30.6868
 18   -0.7   31.0907
 19   -0.8   31.7645
 20   -0.9   32.7044
 21   -1.0   33.9089
```

The R function for the Box-Cox Transform (`bxcx.R`) is available at

https://github.com/AppliedStat/LM

R

```
> ## Using web link:
> source("https://raw.githubusercontent.com/AppliedStat/LM/master/bxcx.R")
>
> mydata =
    read.table("https://raw.githubusercontent.com/AppliedStat/LM/master/CH03TA08.txt")
>
> c1 = mydata[,1]
> c2 = mydata[,2]
>
> lam = seq(1.0, -1.0, by=-0.1)
> sse = bxcx(c2, c1, lambda=lam)
>
> cbind(lam, sse)
        lam      sse
 [1,]   1.0 77.98306
 [2,]   0.9 70.35050
 [3,]   0.8 63.66932
 [4,]   0.7 57.83686
 [5,]   0.6 52.76343
 [6,]   0.5 48.37072
 [7,]   0.4 44.59051
 [8,]   0.3 41.36342
 [9,]   0.2 38.63791
[10,]   0.1 36.36939
[11,]   0.0 34.51945
[12,]  -0.1 33.05520
[13,]  -0.2 31.94867
[14,]  -0.3 31.17631
[15,]  -0.4 30.71859
[16,]  -0.5 30.55961
[17,]  -0.6 30.68680
[18,]  -0.7 31.09066
[19,]  -0.8 31.76453
[20,]  -0.9 32.70442
[21,]  -1.0 33.90887

> # ================================================================
> # Using MASS library
> # For help, use
> # > library(help="MASS") or help("boxcox")
> # ----------------------------------------------------------------
> library("MASS")
>
> # Note: 1. find the max. instead of the min.
> #       2. log-likelihood is used instead of SSE
>
> lam = seq(1.0, -1.0, by=-0.1)
> boxcox( c2 ~ c1 , lambda = lam )
```



Python

```python
#!/usr/bin/python3

# If you have " CH01TA01.txt " in your current computer .
# mydata = open("S:/LM/CH01TA01.txt", "r")
# file = mydata.read().splitlines()

# If PC is connected to Internet, then the following works.
from urllib.request import urlopen
link = "https://raw.githubusercontent.com/AppliedStat/LM/master/CH03TA08.txt"
url = urlopen(link)
file = url.read().splitlines()
url.close()

c1 = []
c2 = []
for line in file:
    p = line.split()
    c1.append(float(p[0]))
    c2.append(float(p[1]))

import numpy as np
import pandas as pd
import statsmodels.formula.api as smf

# Implement the boxcox function
def bxcx(y, x, lam):
    n = len(lam)
    sse = []
    k2 = np.exp(np.mean(np.log(y)))
    for i in range(0, n):
        if abs(lam[i])>0.0001:
            k1 = 1/(lam[i]* k2**(lam[i]-1))
```

```
            w = k1 * (y**lam[i]-1)
        else:
            w = k2 * np.log(y)
        data = pd.DataFrame({"x":x, "w":w})
        LM = smf.ols(formula = 'w ~ x', data = data).fit()
        res = LM.resid
        sse.append(sum(res**2))
    return sse

lam = np.arange(1.0, -1.1, -0.1)
sse = bxcx(c2,c1,lam)

#cbind = np.vstack((lam, sse))
cbind = pd.DataFrame({"lam":lam, "sse":sse})
print(cbind)

#
               lam         sse
0    1.000000e+00   77.983064
1    9.000000e-01   70.350503
2    8.000000e-01   63.669322
3    7.000000e-01   57.836861
4    6.000000e-01   52.763425
5    5.000000e-01   48.370723
6    4.000000e-01   44.590514
7    3.000000e-01   41.363420
8    2.000000e-01   38.637906
9    1.000000e-01   36.369386
10   2.220446e-16   34.519452
11  -1.000000e-01   33.055203
12  -2.000000e-01   31.948669
13  -3.000000e-01   31.176310
14  -4.000000e-01   30.718590
15  -5.000000e-01   30.559611
16  -6.000000e-01   30.686803
17  -7.000000e-01   31.090661
18  -8.000000e-01   31.764528
19  -9.000000e-01   32.704418
20  -1.000000e+00   33.908874
```

Note:

1. Always check the plot of residuals after transformation.
2. Use the Box-Cox transformation only as a rough guide to selecting λ. It is better to use *nice* values like $\lambda = 0, 1/2, 1/3, -1$, etc. than *weird* values like $\lambda = 0.3645$.

References

Hollander, M. and Wolfe, D. A. (2013). *Nonparametric Statistical Methods*. Wiley, New York, 3rd edition.