

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 $\text{Var}(\epsilon_i)$ 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.
($\Rightarrow \hat{\epsilon}_i/\sqrt{\text{MSE}}$ 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:

$$(i) \sum \hat{\epsilon}_i = 0 \quad \text{and} \quad (ii) \sum X_i \hat{\epsilon}_i = 0.$$

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

```
1 SUBC> file "S:\LM\CH01TA01.txt" .
2
3 Entering data from file: S:\LM\CH01TA01.TXT
4 25 rows read.
```

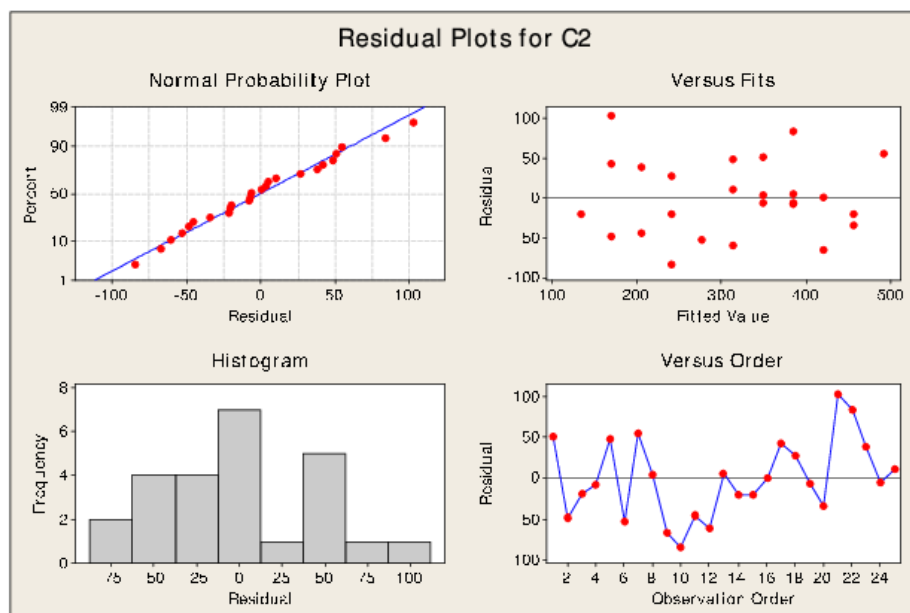
The Minitab macro command `%resplots` (old version) also gives four graphics, which are a normal probability plot, a time-series plot, a histogram, and a scatter plot of $\hat{\epsilon}_i$ vs. \hat{Y}_i .

Residual Plots

```

1  ## store residuals into c3 and fitted Y into c21
2  regr c2 1 c1;
3  fits c21;
4  resid c3.
5
6  ## Call resplots.mac (older version)
7  %resplots c3 c21.
8
9  ## New version
10 ## Stat -> Regression -> Graphs... -> (click four in one) -> OK

```



R

Read the data set

```

1 # If you have " CH01TA01.txt " in your current computer .
2 # mydata = read.table("S:\\LM\\CH01TA01.txt")
3
4 # If PC is connected to Internet, then the following works.
5 mydata =
  read.table("https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt")

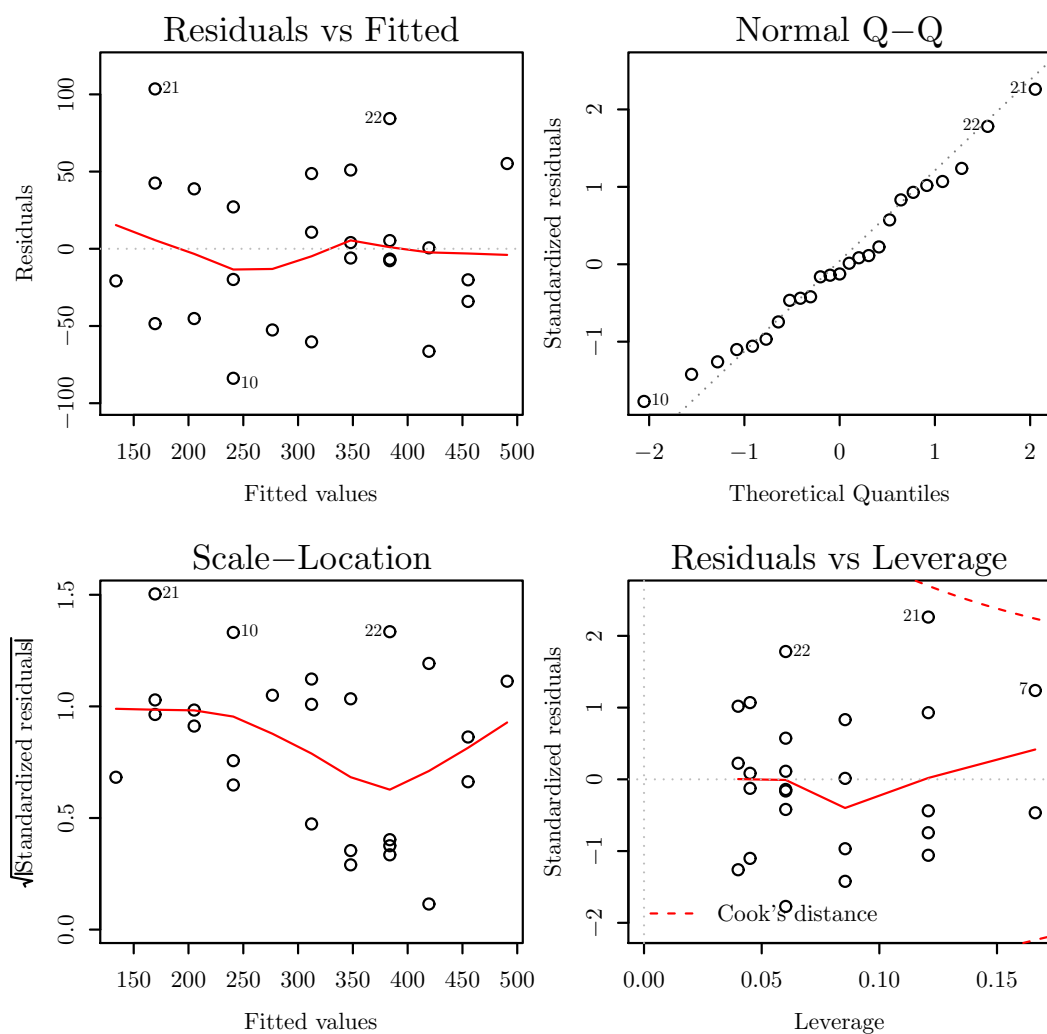
```

Residuals Plot

```

1 x = mydata[,1]
2 y = mydata[,2]
3 LM = lm( y ~ x )
4
5 par ( mfrow=c(2,2) ) ## Put four plots into one sheet
6 plot(LM)

```



Python

Read the data set

```

1  #!/usr/bin/python3
2
3  # If you have " CH01TA01.txt " in your current computer .
4  # mydata = open("S:/LM/CH01TA01.txt", "r")
5  # file = mydata.read().splitlines()
6
7  # If PC is connected to Internet, then the following works.
8  from urllib.request import urlopen
9  link = "https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt"
10 url = urlopen(link)
11 file = url.read().splitlines()
12 url.close()
13
14 x = []
15 y = []
16 for line in file:
17     p = line.split()
18     x.append(float(p[0]))
19     y.append(float(p[1]))

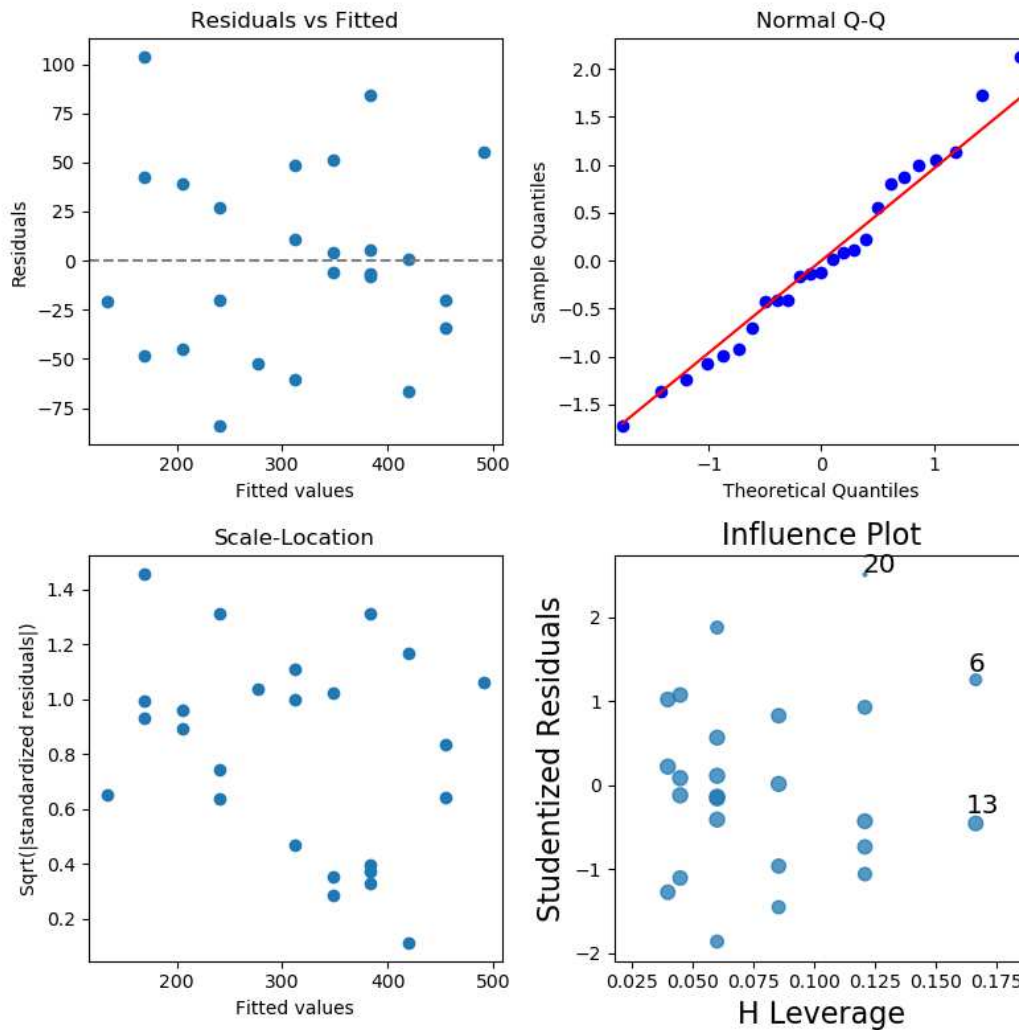
```

Residuals Plot

```

1  import statsmodels.formula.api as smf
2  import pandas as pd
3
4  data1 = pd.DataFrame({"x":x, "y":y})
5  lm = smf.ols(formula = 'y ~ x', data = data1).fit()
6
7  import matplotlib.pyplot as plt
8  import statsmodels.api as sm
9
10 fitted = lm.predict()
11 resids = lm.resid
12 std_resids = lm.resid_pearson
13
14 # 4 plots in one window
15 fig = plt.figure(figsize = (8, 8), dpi = 100)
16
17 ## raw residuals vs. fitted
18 ax1 = fig.add_subplot(2, 2, 1)
19 ax1.plot(fitted, resids, 'o')
20 l = plt.axhline(y = 0, color = 'grey', linestyle = 'dashed')
21 ax1.set_xlabel('Fitted values')
22 ax1.set_ylabel('Residuals')
23 ax1.set_title('Residuals vs Fitted')
24
25 ## q-q plot
26 ax2 = fig.add_subplot(2, 2, 2)
27 sm.qqplot(std_resids, line='s', ax = ax2)
28 ax2.set_title('Normal Q-Q')
29
30 ## scale-location
31 ax3 = fig.add_subplot(2, 2, 3)
32 ax3.plot(fitted, abs(std_resids)**.5, 'o')
33 ax3.set_xlabel('Fitted values')
34 ax3.set_ylabel('Sqrt(|standardized residuals|)')
35 ax3.set_title('Scale-Location')
36
37 ## residuals vs. leverage
38 ax4 = fig.add_subplot(2, 2, 4)
39 sm.graphics.influence_plot(lm, criterion = 'Cooks', size = 2, ax = ax4)
40
41 plt.tight_layout()
42 fig.savefig('regplots.png')

```



||

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.

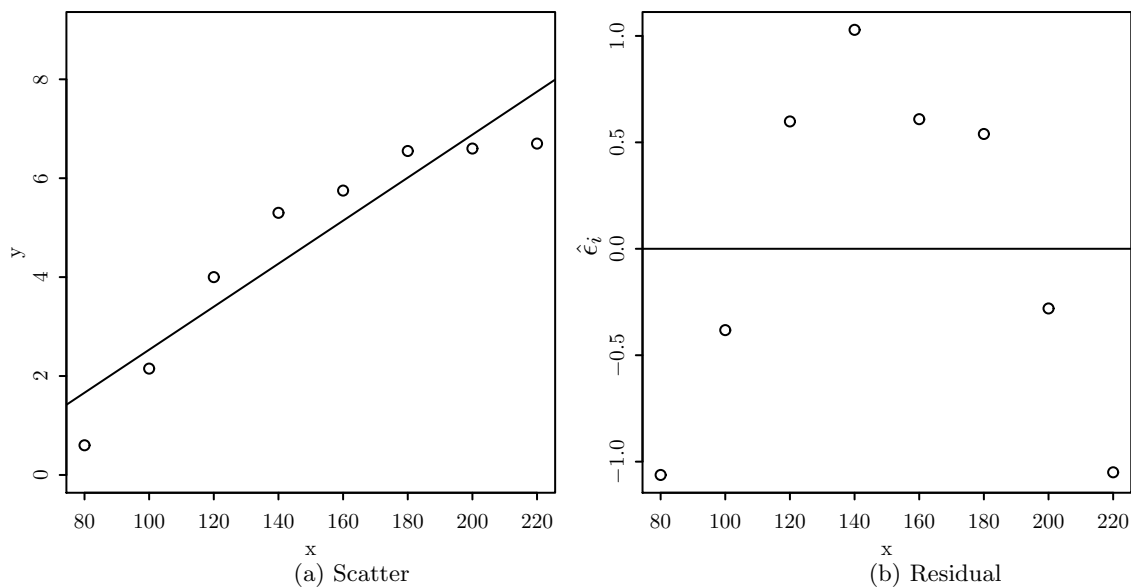
2.1 Non-linearity of regression function

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.

Scatter plot and residual plot



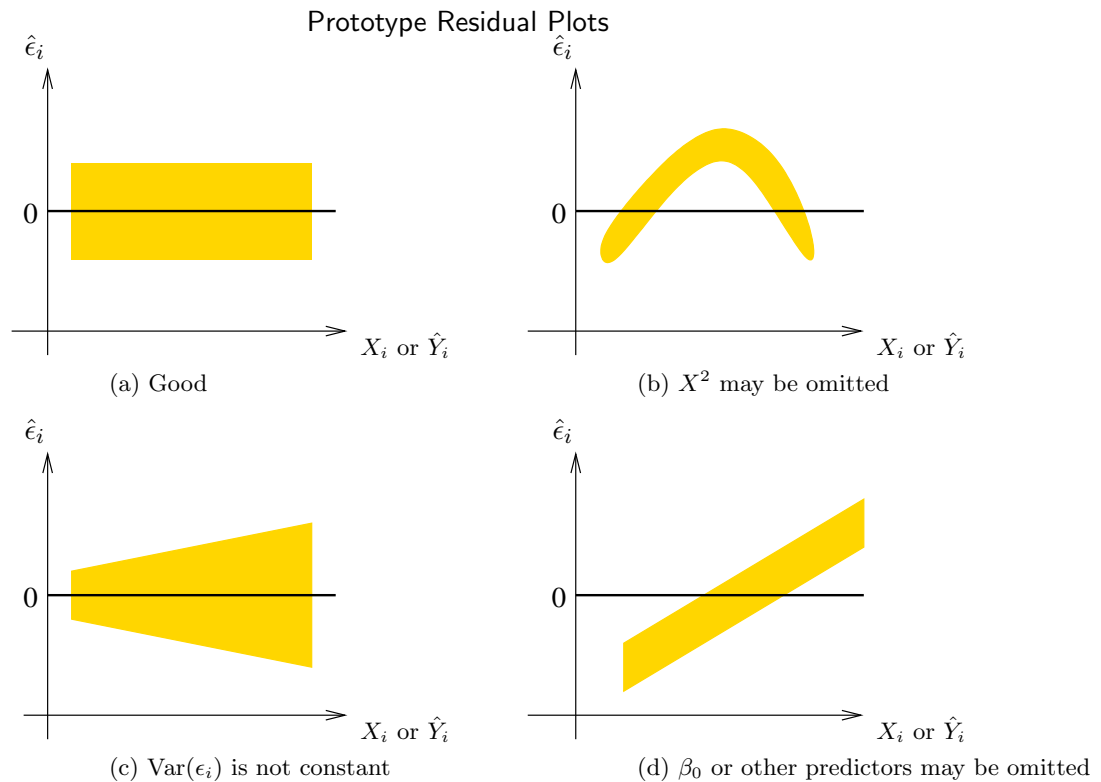
The above plots were made by R using the data in Table 3.1 on Page 105 of the text.

R

```

1 x = c(80, 220, 140, 120, 180, 100, 200, 160)
2 y = c(0.60, 6.70, 5.30, 4.00, 6.55, 2.15, 6.60, 5.75)
3
4 LM = lm(y~x)
5 y1 = fitted(LM)
6 r = y - y1
7
8 par(mfrow=c(1,2)) ## two plots into one sheet
9 plot(x,y, ylim=c(0,9), sub="(a)")
10 abline(coef(LM))
11
12 plot(x,r, sub="(b)")
13 abline(h=0)

```



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

```

1  ## Data set from Table 3.1 pg. 105
2  MTB > set c1
3  DATA> 80 220 140 120 180 100 200 160
4  DATA> end
5  MTB > set c2
6  DATA> .60 6.70 5.30 4.0 6.55 2.15 6.6 5.75
7  DATA> end
8  MTB > regr c2 1 c1;
9  SUBC> fits c3;
10 SUBC> resid c5.
11
12 Regression Analysis: C2 versus C1
13
14 The regression equation is
15 C2 = - 1.82 + 0.0435 C1
16
17 Predictor      Coef    SE Coef      T      P

```



```

18 Constant      -1.816      1.052    -1.73    0.135
19 C1             0.043482   0.006706    6.48    0.001
20
21 S = 0.869241    R-Sq = 87.5%    R-Sq(adj) = 85.4%
22
23 Analysis of Variance
24 Source         DF         SS         MS         F         P
25 Regression      1    31.764    31.764    42.04    0.001
26 Residual Error  6     4.533     0.756
27 Total           7    36.297
28
29 MTB > let c4 = c3**2.
30 MTB > regr c5 2 c3 c4.
31
32 Regression Analysis: C5 versus C3, C4
33
34 The regression equation is
35 C5 = - 3.87 + 2.00 C3 - 0.213 C4
36
37 Predictor      Coef    SE Coef      T      P
38 Constant     -3.8702    0.3896   -9.93  0.000
39 C3             2.0039    0.1843   10.87  0.000
40 C4            -0.21290   0.01925  -11.06  0.000
41
42 S = 0.188738    R-Sq = 96.1%    R-Sq(adj) = 94.5%
43
44 Analysis of Variance
45 Source         DF         SS         MS         F         P
46 Regression      2     4.3554    2.1777   61.13  0.000
47 Residual Error  5     0.1781    0.0356
48 Total           7     4.5335
49
50 Source  DF  Seq SS
51 C3       1  0.0000
52 C4       1  4.3554

```

R

```

1 > x = c(80, 220, 140, 120, 180, 100, 200, 160)
2 > y = c(0.60, 6.70, 5.30, 4.00, 6.55, 2.15, 6.60, 5.75)
3
4 > LM = lm(y~x)
5 > c3 = fitted(LM)
6 > c4 = c3^2
7 > c5 = resid(LM)
8
9 > LM2 = lm(c5 ~ c3 + c4)
10 > summary(LM2)
11
12 Call:
13 lm(formula = c5 ~ c3 + c4)
14
15 Residuals:
16      1      2      3      4      5      6      7      8
17  0.06458  0.07708  0.22351  0.11518  0.05625 -0.22113 -0.11935 -0.19613
18
19 Coefficients:
20             Estimate Std. Error t value Pr(>|t|)
21 (Intercept) -3.87017    0.38961   -9.933  0.000177 ***
22 c3           2.00392    0.18430   10.873  0.000114 ***
23 c4          -0.21290    0.01925  -11.057  0.000105 ***
24 ---
25 Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
26
27 Residual standard error: 0.1887 on 5 degrees of freedom
28 Multiple R-Squared:  0.9607,    Adjusted R-squared:  0.945
29 F-statistic: 61.13 on 2 and 5 DF,  p-value: 0.0003059

```

Python

```

1  #!/usr/bin/python3
2
3  import pandas as pd
4  import statsmodels.formula.api as smf
5
6  x = [80, 220, 140, 120, 180, 100, 200, 160]
7  y = [0.60, 6.70, 5.30, 4.00, 6.55, 2.15, 6.60, 5.75]
8
9  data1 = pd.DataFrame({"x":x, "y":y})
10 LM = smf.ols(formula = 'y ~ x', data = data1).fit()
11 c3 = LM.fittedvalues
12 c4 = c3 ** 2
13 c5 = LM.resid #Residuals
14
15 data2 = pd.DataFrame({"c3":c3, "c4":c4, "c5":c5})
16 LM2 = smf.ols(formula = 'c5 ~ c3 + c4', data = data2).fit()
17 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. \triangle

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 i th residual for group I and $\hat{\epsilon}_{i2}$ to denote the i th 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 $|t_L^*| \leq t(1 - \frac{\alpha}{2}; n_1 + n_2 - 2)$, 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 $(d_{i11}, d_{i22}, \dots, d_{i_kk})$, 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_{BP}^2 is as follows:

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

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_{BP}^2 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_{BP}^2 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 $i = 1, 2, \dots, n_1$ and $j = 1, 2, \dots, n_2$. 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_2^2 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

```

1 MTB > READ C1 C2;
2 SUBC> file "S:\LM\CH01TA01.txt" .
3 Entering data from file: S:\LM\CH01TA01.TXT
4 25 rows read.
5 MTB > regr c2 1 c1;
6 SUBC> resid c3;
7 SUBC> fits c4.
8
9 MTB > let c5 = c3**2
10 MTB > regr c5 1 c4.
11
```

```

12 Regression Analysis: C5 versus C4
13
14 The regression equation is
15 C5 = 3940 - 5.59 C4
16
17 Predictor      Coef    SE Coef      T      P
18 Constant      3940      1756      2.24  0.035
19 C4             -5.593     5.354     -1.04  0.307
20
21 S = 2689.72    R-Sq = 4.5%    R-Sq(adj) = 0.4%
22
23 Analysis of Variance
24 Source          DF          SS          MS          F          P
25 Regression       1       7896142    7896142     1.09    0.307
26 Residual Error   23    166395896    7234604
27 Total            24    174292038
28
29 Unusual Observations
30 Obs    C4      C5    Fit    SE Fit    Residual    St Resid
31   21  169   10718   2992      935       7726       3.06R
32   22  384    7109   1794      660       5316       2.04R
33
34 R denotes an observation with a large standardized residual.

```

R

Read the data set

```

1 # If you have "CH01TA01.txt" in your current computer.
2 > mydata = read.table("S:\\LM\\CH01TA01.txt")
3
4 # If your computer is connected to Internet
5 > mydata =
   read.table("https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt")

```

Regression analysis

```

1 > c1 = mydata[,1]
2 > c2 = mydata[,2]
3 > LM = lm(c2 ~ c1)
4
5 > c3 = resid(LM)
6 > c4 = fitted(LM)
7 > c5 = c3^2
8
9 > LM2 = lm(c5 ~ c4)
10 > summary(LM2)
11
12 Call:
13 lm(formula = c5 ~ c4)
14
15 Residuals:
16      Min       1Q   Median       3Q      Max
17 -2760.1  -1765.4  -990.7   609.5   7726.3
18
19 Coefficients:
20             Estimate Std. Error t value Pr(>|t|)
21 (Intercept) 3939.750   1756.371    2.243  0.0348 *
22 c4           -5.593     5.354    -1.045  0.3070
23 ---
24 Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
25
26 Residual standard error: 2690 on 23 degrees of freedom
27 Multiple R-Squared:  0.0453,    Adjusted R-squared:  0.003796
28 F-statistic: 1.091 on 1 and 23 DF,  p-value: 0.307

```

Python

Read the data set

```

1  #!/usr/bin/python3
2
3  # If you have " CH01TA01.txt " in your current computer .
4  # mydata = open("S:/LM/CH01TA01.txt", "r")
5  # file = mydata.read().splitlines()
6
7  # If PC is connected to Internet, then the following works.
8  from urllib.request import urlopen
9  link = "https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt"
10 url = urlopen(link)
11 file = url.read().splitlines()
12 url.close()
13
14 c1 = []
15 c2 = []
16 for line in file:
17     p = line.split()
18     c1.append(float(p[0]))
19     c2.append(float(p[1]))

```

Regression analysis

```

1  import pandas as pd
2  import statsmodels.formula.api as smf
3
4  data1 = pd.DataFrame({"c1":c1, "c2":c2})
5  LM = smf.ols(formula = 'c2 ~ c1', data = data1).fit()
6
7  c3 = LM.resid
8  c4 = LM.fittedvalues
9  c5 = c3 ** 2
10
11 data2 = pd.DataFrame({"c4":c4, "c5":c5})
12 LM2 = smf.ols(formula = 'c5 ~ c4', data = data2).fit()
13 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>

```

1  MTB > READ C1 C2;
2  SUBC> file "S:\LM\CH01TA01.txt" .
3  Entering data from file: S:\LM\CH01TA01.TXT
4  25 rows read.
5
6  MTB > regr c2 1 c1;
7  SUBC> resid c3.
8
9  MTB > sort c1 c3 c4 c5;
10 SUBC> by c1.
11 MTB > print c1 c3 c4 c5
12
13 Data Display
14 Row    C1      C3      C4      C5
15   1    80    51.018    20   -20.770

```

```

16      2      30    -48.472      30    -48.472
17      3      50    -19.876      30     42.528
18      .....
19
20 MTB > copy c5 c11
21 MTB > copy c5 c12
22 MTB > delete 14:25 c11
23 MTB > delete 1:13 c12
24
25 MTB > %S:\LM\levене c11 c12 k1
26 Executing from file: S:\LM\levене.MAC
27
28 Data Display
29
30 K1      1.31648
31
32 MTB > invcdf 0.975;
33 SUBC> t 23.
34
35 Inverse Cumulative Distribution Function
36
37 Student's t distribution with 23 DF
38
39 P( X <= x )      x
40      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. \triangle

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

<https://github.com/AppliedStat/LM>

```

1 > # First, save the levене.R file at your current directory.
2 > # source("S:/LM/levене.R")
3 >
4 > # If your PC is connected to Internet, the following will work:
5 > source("https://raw.githubusercontent.com/AppliedStat/LM/master/levене.R")
6 >
7 > mydata =
8     read.table("https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt")
9 >
10 > c1 = mydata[,1]
11 > c2 = mydata[,2]
12 >
13 > LM = lm(c2 ~ c1)
14 >
15 > c3 = resid(LM)
16 >
17 > c1.order = order(c1)
18 > c4 = c1 [ c1.order ]
19 > c5 = c3 [ c1.order ]
20 > print( cbind(c1,c3,c4,c5) )
21      c1      c3      c4      c5
22 1    80 51.0179798 20 -20.7698990
23 2    30 -48.4719192 30 -48.4719192
24 3    50 -19.8759596 30 42.5280808
25 4    90 -7.6840404 30 103.5280808
26      .....

```

```

27 >
28 > gr1 = c5[1:13]
29 > gr2 = c5[14:25]
30 >
31 > levene.test(gr1, gr2)
32 $t.test.stat
33 [1] 1.316482
34
35 $df
36 [1] 23
37
38 $p.value
39 [1] 0.2009812

```

Python

```

1  #!/usr/bin/python3
2
3  # If you have " CH01TA01.txt " in your current computer .
4  # mydata = open("S:/LM/CH01TA01.txt", "r")
5  # file = mydata.read().splitlines()
6
7  # If PC is connected to Internet, then the following works.
8  from urllib.request import urlopen
9  link = "https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt"
10 url = urlopen(link)
11 file = url.read().splitlines()
12 url.close()
13
14 c1 = []
15 c2 = []
16 for line in file:
17     p = line.split()
18     c1.append(int(p[0]))
19     c2.append(int(p[1]))
20
21 import numpy as np
22 import pandas as pd
23 import scipy.stats as stats
24 import statsmodels.formula.api as smf
25 data1 = pd.DataFrame({"c1":c1, "c2":c2})
26 LM = smf.ols(formula = 'c2 ~ c1', data = data1).fit()
27
28 c3 = LM.resid
29
30 c1 = np.asarray(c1)
31 c3 = np.asarray(c3)
32 c1_order = np.argsort(c1, kind = 'stable')
33 c4 = c1[c1_order]
34 c5 = c3[c1_order]
35
36 # cbind = np.vstack((c1, c3, c4, c5))
37 cbind = pd.DataFrame({"c1":c1, "c2":c2, "c3":c3, "c4":c4})
38 print(cbind)
39
40 gr1 = c5[0:13]
41 gr2 = c5[13:25]
42
43 w, p = stats.levene(gr1, gr2)
44 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:

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

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. \triangle

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>

```

1 MTB ># 1. Read the data
2 MTB > READ C1 C2;
3 SUBC> file "S:\LM\CH01TA01.TXT" .
4 Entering data from file: S:\LM\CH01TA01.TXT
5 25 rows read.
6
7 MTB > # 2. RUN BPtest Macro
8 MTB > %S:\LM\BPtest C2 C1 .
9 Executing from file: S:\LM\BPtest.MAC
10
11 Data Display
12
13 Breusch-Pagan Test Statistic:      0.82092
14 Degrees of Freedom:              1
15 p-value:                          0.36491

```

R

```

1 > mydata =
   read.table("https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt")
2 >
3 > c1 = mydata[,1]
4 > c2 = mydata[,2]
5 >
6 > LM = lm(c2 ~ c1)
7 >
8 > e = resid(LM)
9 >
10 > sigma2 = e^2
11 >
12 > LM2 = lm ( sigma2 ~ c1 )
13 >
14 > SSR.star = sum( (fitted(LM2)-mean(sigma2))^2 )
15 >
16 > SSE = sum( (fitted(LM)-c2)^2 )
17 >
18 > n = length(c2)

```

```

19 >
20 > cbind(SSR.star, SSE, n)
21      SSR.star      SSE      n
22 [1,] 7896142 54825.46 25
23 >
24 > X_BP = SSR.star/2 / ( (SSE/n)^2 )
25 >
26 > X_BP
27 [1] 0.8209192
28 >
29 > qchisq(0.95, df = 1) ## chi-square critical value
30 [1] 3.841459

```

Python

```

1  #!/usr/bin/python3
2
3  # If you have " CH01TA01.txt " in your current computer .
4  # mydata = open("S:/LM/CH01TA01.txt", "r")
5  # file = mydata.read().splitlines()
6
7  # If PC is connected to Internet, then the following works.
8  from urllib.request import urlopen
9  link = "https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt"
10 url = urlopen(link)
11 file = url.read().splitlines()
12 url.close()
13
14 c1 = []
15 c2 = []
16 for line in file:
17     p = line.split()
18     c1.append(int(p[0]))
19     c2.append(int(p[1]))
20
21 import numpy as np
22 import pandas as pd
23 import scipy.stats as stats
24 import statsmodels.formula.api as smf
25
26 data1 = pd.DataFrame({"c1":c1, "c2":c2})
27 LM = smf.ols(formula = 'c2 ~ c1', data = data1).fit()
28
29 e = LM.resid
30 sigma2 = e**2
31 data2 = pd.DataFrame({"c1":c1, "sigma2":sigma2})
32 LM2 = smf.ols(formula = 'sigma2 ~ c1', data = data2).fit()
33 SSR_star = sum((LM2.fittedvalues-np.mean(sigma2))**2)
34 SSE = sum((LM.fittedvalues-c2)**2)
35 n = len(c2)
36
37 cbind = pd.DataFrame({"SSR_star":SSR_star, "SSE":SSE, "n":n}, index = [0])
38 print('\nData Display of SSR_star,SSE and n:')
39 print(cbind)
40
41 X_BP = SSR_star/2/((SSE/n)**2)
42 print('\nX_BP:')
43 print(X_BP)
44
45 qchisq = stats.chi2.ppf(0.95, df=1)
46 print('\n qchisq:')
47 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:

```
> qchisq(1 -  $\alpha$ , df).
```

Since $X_{BP}^2 = 0.8209192$ 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>

```
1 > source("https://raw.githubusercontent.com/AppliedStat/LM/master/Breusch-Pagan.R")
2
3 > BP.test ( c2 ~ c1)
4 $test.stat
5 [1] 0.8209192
6
7 $df
8 [1] 1
9
10 $p.value
11 [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

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

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:

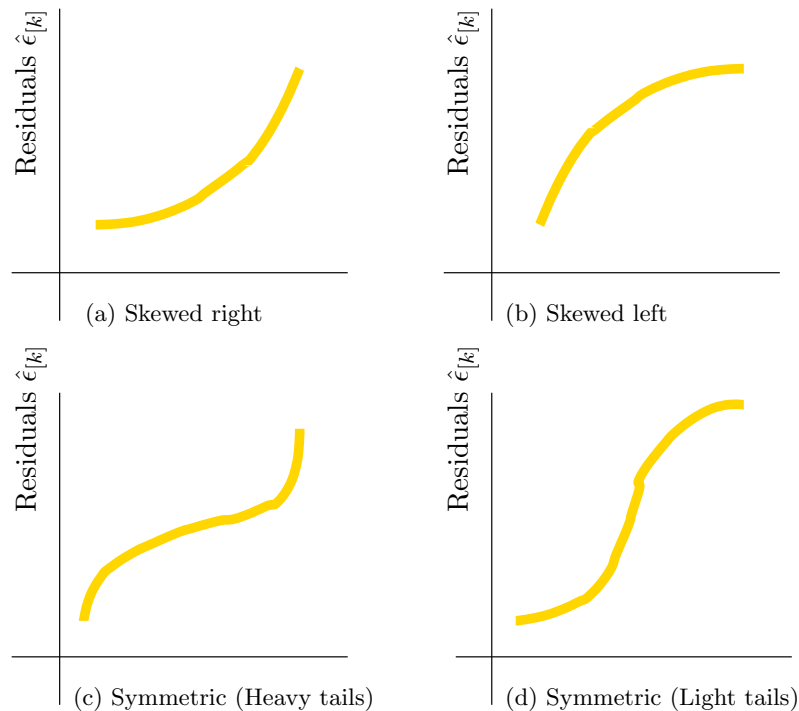
100(1 - α)% of the residuals $\hat{\epsilon}_i$ fall between $\pm \sqrt{\text{MSE}} \cdot t(1 - \frac{\alpha}{2}; \text{df} = n - p)$, where p is the number of parameters.

Test

Calculate the sample correlation coefficient between $\hat{\epsilon}_{[k]}$ and $E(\hat{\epsilon}_{[k]}) \approx \sqrt{\text{MSE}} \Phi^{-1} \left(\frac{k - 0.375}{n + 0.25} \right)$.

Find the critical value for n with α from Table B.6 of Kutner et al. (2005) or Table 2 of Looney and Gullledge, Jr. (1985). If the sample correlation coefficient is at least as large as

Q-Q Plots when error term is not Normal



the critical value from Table B.6 (Kutner et al., 2005) or Table 2 (Looney and Gulledge, Jr., 1985), then one can conclude that the error terms are reasonably normally distributed.

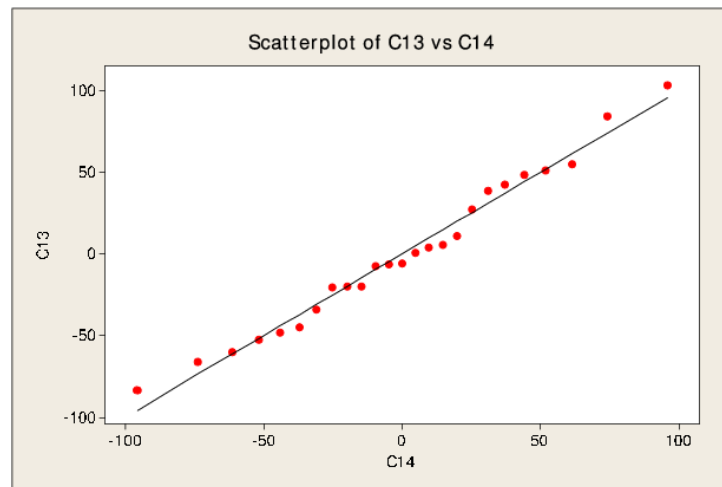
Remark 3.5. Personally, I am more concerned with getting the variance constant. And residuals may appear to be not normal because an inappropriate regression function is used or because the variance of error terms is not constant. \triangle

Example 3.6. Normal Probability Plot, which is based on the Table B.6 of Kutner et al. (2005) (textbook). Note that the normal probability plot using the **Minitab** built-in function, `%resplot`, uses the horizontal axis for the residuals.

Minitab

```

1 MTB > READ C1 C2;
2 SUBC> file "S:\LM\CH01TA01.txt" .
3 Entering data from file: S:\LM\CH01TA01.TXT
4 25 rows read.
5
6 MTB > regr c2 1 c1;
7 SUBC> resid c3;
8 SUBC> mse k2.
9
```



```

10 MTB >let k1 = count(c3)
11
12 MTB > set c10
13 DATA> 1:k1
14 DATA> end .
15
16 MTB >sort c3 c13
17 MTB > let c11 = (c10 - .375) / (k1+.25)
18
19 MTB >invcdf c11 c12;
20 SUBC> normal 0 1.
21 MTB >let c14 = sqrt(k2)*c12
22 MTB >plot c13*c14 ;
23 SUBC> line c14 c14 .

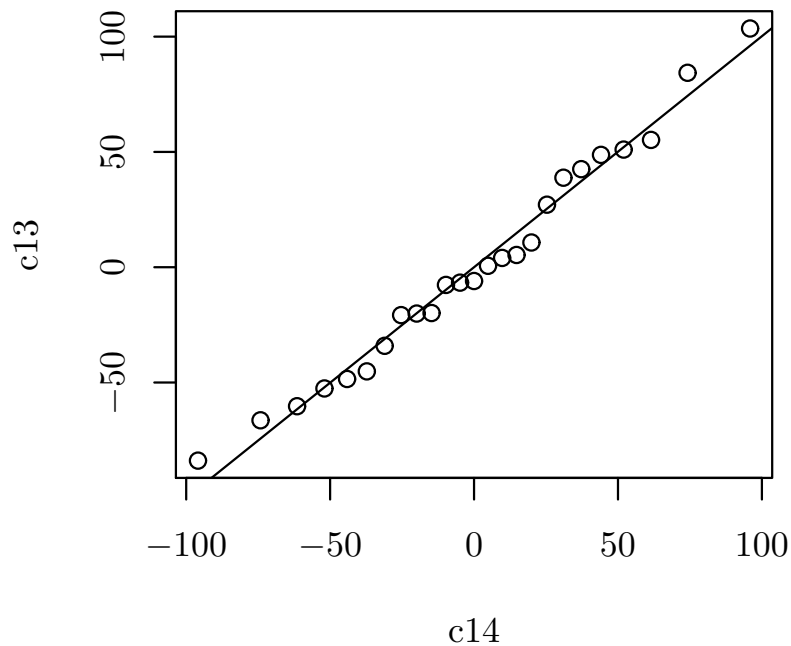
```

R

```

1 \vspace{-0.5\baselineskip}
2 > mydata =
  read.table("https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt")
3
4 > c1 = mydata[,1]
5 > c2 = mydata[,2]
6 >
7 > LM = lm(c2 ~ c1)
8 >
9 > c3 = resid(LM)
10 >
11 > LM.sum = summary(LM)
12 >
13 > attributes(LM.sum)
14 $names
15 [1] "call"          "terms"          "residuals"      "coefficients"
16 [5] "aliased"        "sigma"          "df"             "r.squared"
17 [9] "adj.r.squared" "fstatistic"     "cov.unscaled"
18
19 $class
20 [1] "summary.lm"

```



```

21
22 > s = LM.sum[[ "sigma" ]]
23 >
24 > c13 = sort(c3)
25 >
26 > n = length(c3)
27 >
28 > k = 1:n
29 >
30 > c11 = (k - 0.375) / (n+0.25)
31 >
32 > c12 = qnorm( c11 )
33 >
34 > c14 = s * c12
35 >
36 > postscript( "ex4a.ps", width=4, height=4)
37 >
38 > plot ( c14, c13)
39 > abline(lm(c13~c14))

```

Python

```

1  #!/usr/bin/python3
2
3  # If you have " CH01TA01.txt " in your current computer .
4  # mydata = open("S:/LM/CH01TA01.txt", "r")
5  # file = mydata.read().splitlines()
6
7  # If PC is connected to Internet, then the following works.
8  from urllib.request import urlopen
9  link = "https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt"
10 url = urlopen(link)

```

```

11 file = url.read().splitlines()
12 url.close()
13
14 c1 = []
15 c2 = []
16 for line in file:
17     p = line.split()
18     c1.append(int(p[0]))
19     c2.append(int(p[1]))
20
21 import seaborn as sns
22 # The below may be needed for install seaborn and pandas (numexpr), etc
23 # sudo apt-get update -y
24 # sudo apt-get install -y python3-seaborn
25 # sudo apt-get install -y python3-pandas
26 # sudo apt-get install -y python3-numexpr # latest numexpr is needed for pandas
27 import numpy as np
28 import scipy.stats as stats
29 import pandas as pd
30
31 import statsmodels.formula.api as smf
32 import matplotlib.pyplot as plt
33
34 data1 = pd.DataFrame({"c1":c1, "c2":c2})
35 LM = smf.ols(formula = 'c2 ~ c1', data = data1).fit()
36
37 c3 = LM.resid
38 LM_sum = LM.summary()
39
40
41 # get the residual standard error
42 squared_sigma = LM.mse_resid
43 s = squared_sigma ** 0.5
44
45 c13 = np.sort(c3)
46 n = len(c13)
47
48 c14 = []
49 for k in range(1,n+1):
50     c11 = (k - 0.375)/(n + 0.25)
51     c12 = stats.norm.ppf(c11)
52     c14.append(s * c12)
53
54 c14 = np.asarray(c14)
55
56 fig = plt.figure(figsize = (4,4), dpi = 100)
57 data2 = pd.DataFrame({"c13":c13, "c14":c14})
58 sns.regplot(x="c14",y="c13",data=data2)
59 #sns.lmplot(x="c14",y="c13",data=data2)
60 plt.savefig('ex4a.png')

```

||

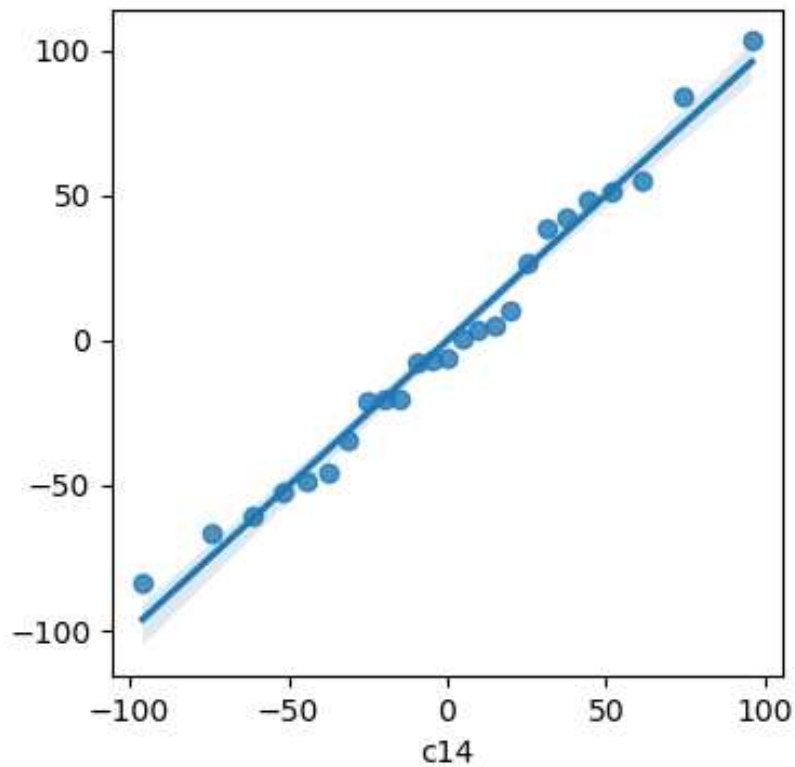
Example 3.7. Correlation test for normality.

Minitab

```

1 MTB > READ C1 C2;
2 SUBC> file "S:\LM\CH01TA01.txt" .
3 Entering data from file: S:\LM\CH01TA01.TXT
4 25 rows read.
5
6 MTB >regr c2 1 c1;
7 SUBC> resid c3;
8 SUBC> mse k2.
9
10 MTB > let k1 = count(c3)

```

```

11
12 MTB > set c10
13 DATA>      1:k1
14 DATA> end .
15
16 MTB > sort c3 c13
17
18 MTB > let c11 = (c10 - .375) / (k1+.25)
19
20 MTB > invcdf c11 c12;
21 SUBC> normal 0 1.
22
23 MTB >let c14 = sqrt(k2)*c12
24
25 MTB > correlation c13 c14 .
26
27 Correlations: C13, C14
28
29 Pearson correlation of C13 and C14 = 0.992
30 P-Value = 0.000

```

Do *not* use the above p -value for the normality test.

R

```

1 > # Using Normal Probability Plot
2 > mydata =
    read.table("https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt")
3 > c1 = mydata[,1]
4 > c2 = mydata[,2]
5 > LM = lm(c2 ~ c1)
6 > c3 = resid(LM)
7 > LM.sum = summary(LM)
8 > s = LM.sum[[ "sigma" ]]
9 > c13 = sort(c3)
10 > n = length(c3)
11 > k = 1:n
12 > c11 = (k - 0.375) / (n+0.25)
13 > c12 = qnorm( c11 )
14 > c14 = s * c12
15 >
16 > cor(c13, c14)
17 [1] 0.9915055
18
19 >
20 > ## Using Q-Q plot
21 > pp = ppoints(c13, a=3/8)    # option "a=3/8=0.375" gives (k-0.375)/(n+0.25)
22 > qq = qnorm(pp)
23 > cor(c13, qq)
24 [1] 0.9915055

```

Python

```

1 #!/usr/bin/python3
2
3 # If you have " CH01TA01.txt " in your current computer .
4 # mydata = open("S:/LM/CH01TA01.txt", "r")
5 # file = mydata.read().splitlines()
6
7
8 # If PC is connected to Internet, then the following works.
9 from urllib.request import urlopen
10 link = "https://raw.githubusercontent.com/AppliedStat/LM/master/CH01TA01.txt"
11 url = urlopen(link)
12 file = url.read().splitlines()
13 url.close()
14
15 c1 = []
16 c2 = []
17 for line in file:
18     p = line.split()
19     c1.append(int(p[0]))
20     c2.append(int(p[1]))
21
22 import numpy as np
23 import scipy.stats as stats
24 import pandas as pd
25 import statsmodels.formula.api as smf
26
27 data1 = pd.DataFrame({"c1":c1, "c2":c2})
28 LM = smf.ols(formula = 'c2 ~ c1', data = data1).fit()
29
30 c3 = LM.resid
31 LM_sum = LM.summary()
32
33 squared_sigma = LM.mse_resid
34 s = squared_sigma ** 0.5
35
36 c13 = np.sort(c3)
37 n = len(c3)
38

```

```

39 c11 = [(k - 0.375)/(n + 0.25) for k in range(1,n+1)]
40
41 c12 = stats.norm.ppf(c11)
42 c14 = s * c12
43
44 cor1 = np.corrcoef(c13, c14)
45 print(cor1)
46
47 a = 3/8
48 m = len(c13)
49
50
51 # pp is written by ourselves, equivalent to ppints in R
52 pp = [(i-a)/(m+1/4) for i in range(1, m+1)]
53
54 qq = stats.norm.ppf(pp)
55 cor2 = np.corrcoef(c13, qq)
56 print(cor2)

```

From Table B.6 of Kutner et al. (2005) (textbook) or Table 2 of Looney and Gulledge, Jr. (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.

||

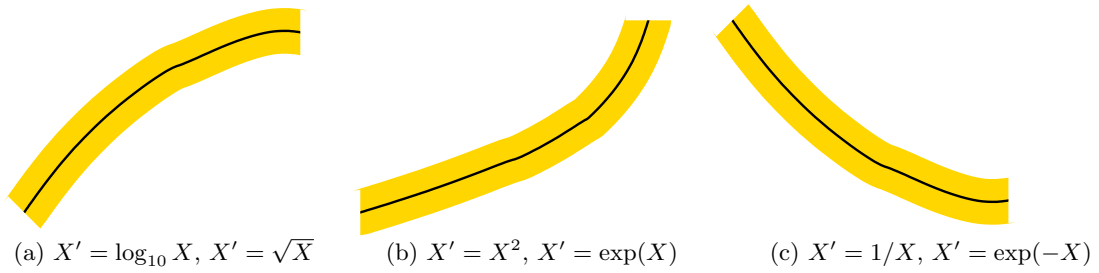
3 Transformations

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.

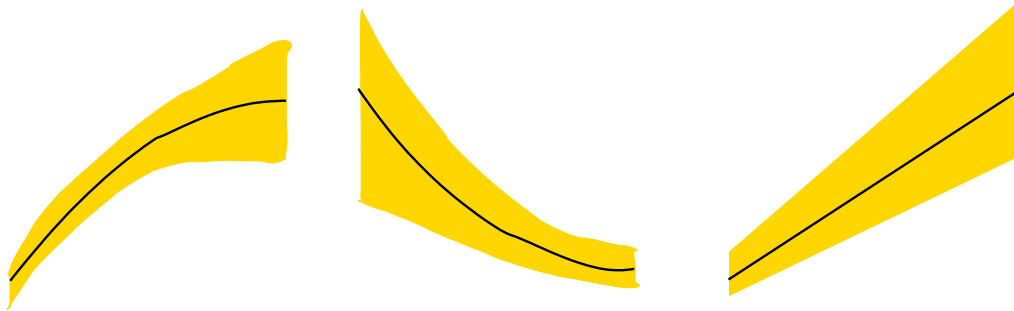
Prototype Nonlinear regression pattern with constant error variance.



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

```

1  ## See Table 3.9 (Section 3.9)
2  MTB > READ c1 c2 c3;
3  SUBC>   file "S:\LM\CH03TA08.txt" .
4  Entering data from file: S:\LM\CH03TA08.TXT
5  25 rows read.
6
7  ## Generate 1.0, 0.9, ..., -1.0
8  MTB >set c3
9  DATA>   10:-10
10 DATA> end
11
12 MTB >let c3 = c3/10
13
14 MTB >%S:\LM\bxcx c2 c1 c3 c4
15 Executing from file: S:\LM\bxcx.MAC
16
17 MTB >print c3 c4
18 Data Display
19 Row      C3      C4
20   1      1.0  77.9831
21   2      0.9  70.3505
22   3      0.8  63.6693
23   4      0.7  57.8369
24   5      0.6  52.7634
25   6      0.5  48.3707
26   7      0.4  44.5905
27   8      0.3  41.3634
28   9      0.2  38.6379
29  10      0.1  36.3694
30  11      0.0  34.5195
31  12     -0.1  33.0552
32  13     -0.2  31.9487
33  14     -0.3  31.1763
34  15     -0.4  30.7186
35  16     -0.5  30.5596
36  17     -0.6  30.6868
37  18     -0.7  31.0907
38  19     -0.8  31.7645
39  20     -0.9  32.7044
40  21     -1.0  33.9089

```

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

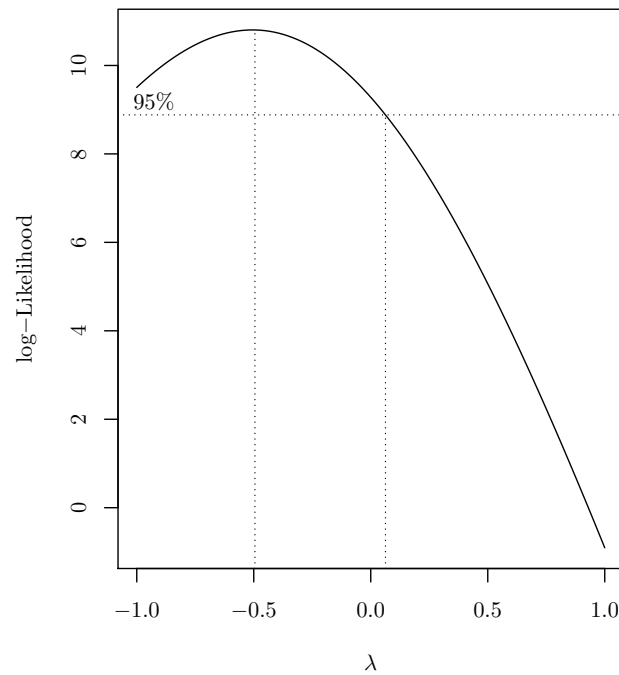
<https://github.com/AppliedStat/LM>

R

```

1 > ## Using web link:
2 > source("https://raw.githubusercontent.com/AppliedStat/LM/master/bxcx.R")
3 >
4 > mydata =
      read.table("https://raw.githubusercontent.com/AppliedStat/LM/master/CH03TA08.txt")
5 >
6 > c1 = mydata[,1]
7 > c2 = mydata[,2]
8 >
9 > lam = seq(1.0, -1.0, by=-0.1)
10 > sse = bxcx(c2, c1, lambda=lam)
11 >
12 > cbind(lam, sse)
13      lam      sse
14 [1,]  1.0 77.98306
15 [2,]  0.9 70.35050
16 [3,]  0.8 63.66932
17 [4,]  0.7 57.83686
18 [5,]  0.6 52.76343
19 [6,]  0.5 48.37072
20 [7,]  0.4 44.59051
21 [8,]  0.3 41.36342
22 [9,]  0.2 38.63791
23 [10,] 0.1 36.36939
24 [11,] 0.0 34.51945
25 [12,] -0.1 33.05520
26 [13,] -0.2 31.94867
27 [14,] -0.3 31.17631
28 [15,] -0.4 30.71859
29 [16,] -0.5 30.55961
30 [17,] -0.6 30.68680
31 [18,] -0.7 31.09066
32 [19,] -0.8 31.76453
33 [20,] -0.9 32.70442
34 [21,] -1.0 33.90887
35
36 > # =====
37 > # Using MASS library
38 > # For help, use
39 > # > library(help="MASS") or help("boxcox")
40 > # -----
41 > library("MASS")
42 >
43 > # Note: 1. find the max. instead of the min.
44 > #       2. log-likelihood is used instead of SSE
45 >
46 > lam = seq(1.0, -1.0, by=-0.1)
47 > boxcox( c2 ~ c1 , lambda = lam )

```



Python

```

1  #!/usr/bin/python3
2
3  # If you have " CH01TA01.txt " in your current computer .
4  # mydata = open("S:/LM/CH01TA01.txt", "r")
5  # file = mydata.read().splitlines()
6
7  # If PC is connected to Internet, then the following works.
8  from urllib.request import urlopen
9  link = "https://raw.githubusercontent.com/AppliedStat/LM/master/CH03TA08.txt"
10 url = urlopen(link)
11 file = url.read().splitlines()
12 url.close()
13
14 c1 = []
15 c2 = []
16 for line in file:
17     p = line.split()
18     c1.append(float(p[0]))
19     c2.append(float(p[1]))
20
21 import numpy as np
22 import pandas as pd
23 import statsmodels.formula.api as smf
24
25 # Implement the boxcox function
26 def bxcx(y, x, lam):
27     n = len(lam)
28     sse = []
29     k2 = np.exp(np.mean(np.log(y)))
30     for i in range(0, n):
31         if abs(lam[i])>0.0001:
32             k1 = 1/(lam[i]* k2**(lam[i]-1))
33             w = k1 * (y**lam[i]-1)

```

```

34         else:
35             w = k2 * np.log(y)
36             data = pd.DataFrame({"x":x, "w":w})
37             LM = smf.ols(formula = 'w ~ x', data = data).fit()
38             res = LM.resid
39             sse.append(sum(res**2))
40         return sse
41
42 lam = np.arange(1.0, -1.1, -0.1)
43 sse = bxcx(c2,c1,lam)
44
45 #cbind = np.vstack((lam, sse))
46 cbind = pd.DataFrame({"lam":lam, "sse":sse})
47 print(cbind)
48
49 #
50
51      lam      sse
52 0  1.000000e+00  77.983064
53 1  9.000000e-01  70.350503
54 2  8.000000e-01  63.669322
55 3  7.000000e-01  57.836861
56 4  6.000000e-01  52.763425
57 5  5.000000e-01  48.370723
58 6  4.000000e-01  44.590514
59 7  3.000000e-01  41.363420
60 8  2.000000e-01  38.637906
61 9  1.000000e-01  36.369386
62 10 2.220446e-16  34.519452
63 11 -1.000000e-01  33.055203
64 12 -2.000000e-01  31.948669
65 13 -3.000000e-01  31.176310
66 14 -4.000000e-01  30.718590
67 15 -5.000000e-01  30.559611
68 16 -6.000000e-01  30.686803
69 17 -7.000000e-01  31.090661
70 18 -8.000000e-01  31.764528
71 19 -9.000000e-01  32.704418
72 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.
- Kutner, M. H., Nachtsheim, C. J., Neter, J., and Li, W. (2005). *Applied Linear Statistical Models*. McGraw-Hill, New York, 5th edition.
- Looney, S. W. and Gulledge, Jr., T. R. (1985). Use of the correlation coefficient with normal probability plots. *The American Statistician*, 39:75–79.