MATH - 4360: Linear Statistical Models

Chapter 5 - Transformation and Weighting to Correct Model Inadequacies

Suthakaran Ratnasingam

Regression model fitting has several implicit assumptions, including the following:

- The model errors have mean zero and constant variance and are uncorrelated.
- The model errors have a normal distribution — this assumption is made in order to conduct hypothesis tests and construct CIs - under this assumption, the errors are independent.
- The form of the model, including the specification of the regressors, is correct.
- The graphical methods help in detecting the violation of basic assumptions in regression analysis. Now we consider the methods and procedures for building the models through data transformation when some of the assumptions are violated.
- We focus on methods and procedures for building regression models when some of the above assumptions are violated.

Example 5.1 The Electric Utility Data

a simple linear regression model is assumed, and the least - squares fit is

$$\hat{y} = -0.8313 + 0.00368x$$

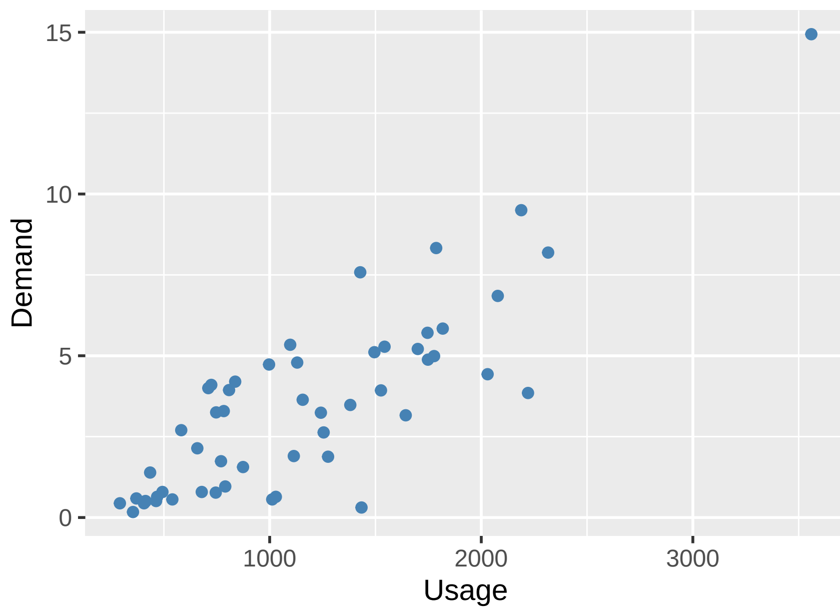
```
rm(list = ls())
# It is assumed that you already have installed the following R packages. If not, please install
# them using the R function: install.packages('package_name')
library(olsrr)
library(ggfortify)
library(ggplot2)
library(car)
library(Rcpp)
library(MASS)
library(faraway)
library(GGally)
library(matlib) # enables function inv()
data1 = read.table("D:\\CSUSB\\Fall 2021\\MATH 4360\\RNotes\\ex51.txt", header = TRUE)
head(data1)
```

```
##   Customer    x    y
## 1        1  679 0.79
## 2        2  292 0.44
## 3        3 1012 0.56
## 4        4  493 0.79
## 5        5  582 2.70
## 6        6 1156 3.64
```

Table 1: Some commonly used variance-stabilizing transformations in the order of their strength are as follows:

Relation of σ^2 to $E(y)$	Transformation
$\sigma^2 \propto$ constant	$y^* = y$ (no transformation)
$\sigma^2 \propto E(y)$	$y^* = \sqrt{y}$ (Poisson data)
$\sigma^2 \propto E(y)[1-E(y)]$	$y^* = \sin^{-1}(\sqrt{y})$ (Binomial proportion $0 \le y_i \le 1$)
$\sigma^2 \propto \Big[E(y)\Big]^2$	$y^* = \ln y$
$\sigma^2 \propto \Big[E(y)\Big]^3$	$y^* = 1/\sqrt{y}$
$\sigma^2 \propto \Big[E(y)\Big]^4$	$y^* = 1/y$

```r
n = nrow(data1)
ggplot(data1, aes(x = x, y = y)) +
  geom_point(color= "steelblue") +
  labs(x = "Usage", y = "Demand", title = "")
```




```r
Fit1 = lm(y ~ x, data = data1)
p = length(coef(Fit1))
summary(Fit1)
```

```
##
## Call:
## lm(formula = y ~ x, data = data1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1399 -0.8275 -0.1934  1.2376  3.1522
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.8313037  0.4416121  -1.882   0.0655 .
## x            0.0036828  0.0003339  11.030 4.11e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.577 on 51 degrees of freedom
## Multiple R-squared:  0.7046, Adjusted R-squared:  0.6988
## F-statistic: 121.7 on 1 and 51 DF,  p-value: 4.106e-15
```

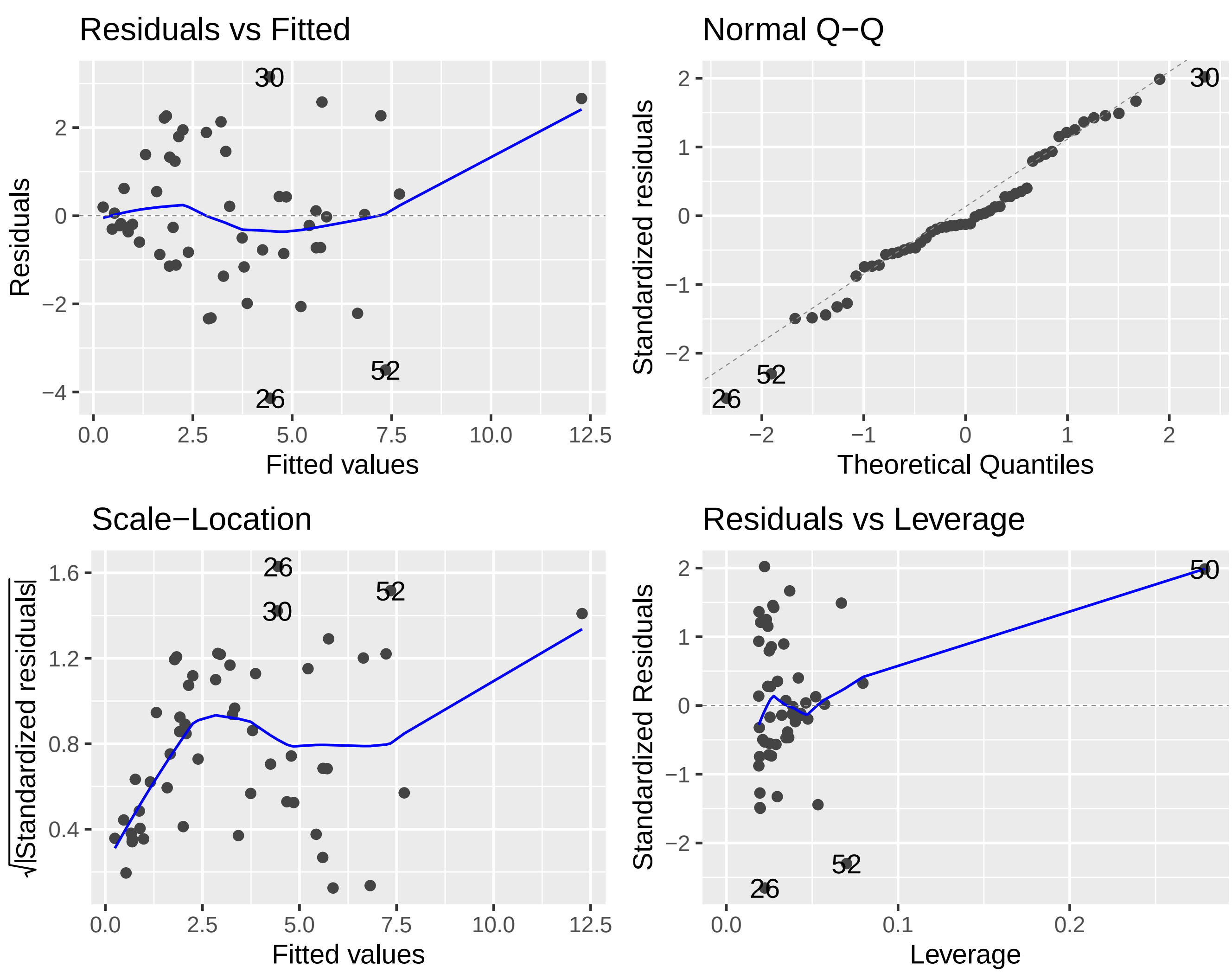
- The analysis of variance is given below. For this model $R^2 = 0.7046$; that is, about 70% of the variability in demand is accounted for by the straight - line fit to energy usage.

```r
anova_fit = anova(Fit1)
tab = as.table(cbind(
  'SS' = c('Regression' = anova_fit[1, 2],
           'Residual' = anova_fit[2, 2],
           'Total' = sum(anova_fit[1:2, 2])),
  'Df' = c( anova_fit[1, 1],
```

```r
            anova_fit[2, 1],
            sum(anova_fit[1:2, 1])),
  'MS' = c( anova_fit[1, 3],
            anova_fit[2, 2] / anova_fit[2, 1],
            NA),
  'F-Test' = c( anova_fit[1, 3]/anova_fit[2, 3],
                NA,
                NA)
))
round(tab, 4)
```

```
##                  SS      Df       MS   F-Test
## Regression 302.6331  1.0000 302.6331 121.6582
## Residual   126.8660 51.0000   2.4876
## Total      429.4992 52.0000
```

```r
autoplot(Fit1)
```

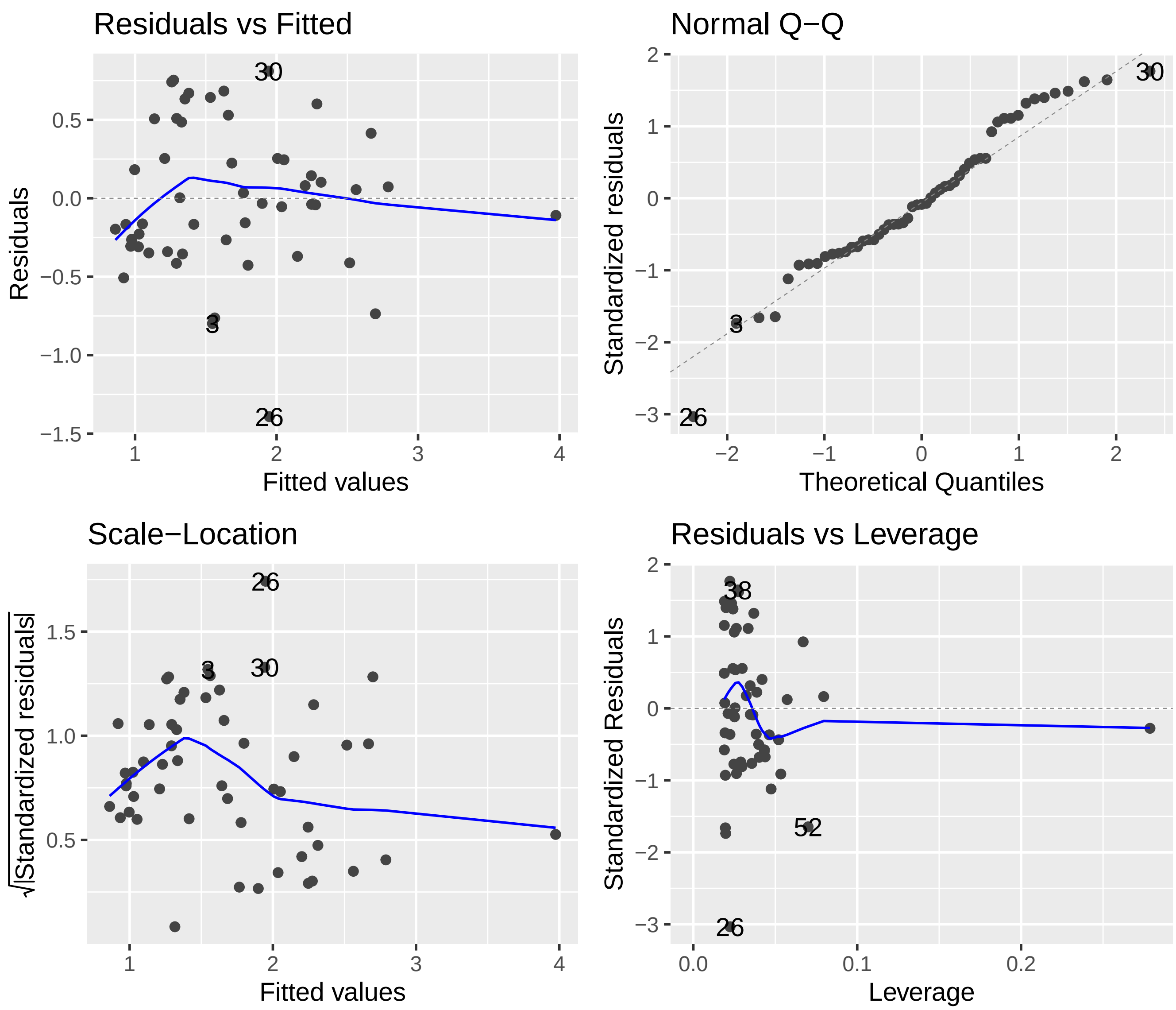


- The $R-$student residuals versus the fitted values $\hat{y}$ shown above. The residuals form an outward - opening funnel, indicating that the error variance is increasing as energy consumption increases.
- To select the form of the transformation, note that the response variable y may be viewed as a "count" of the number of kilowatts used by a customer during a particular hour.
- The simplest probabilistic model for count data is the Poisson distribution.
- This suggests regressing $y^* = \sqrt{y}$ on x as a variance - stabilizing transformation. The resulting least - squares fit is

```r
data1$sqrt_y = sqrt(data1$y)
Fit2 = lm(sqrt_y ~ x, data = data1)
p = length(coef(Fit2))
summary(Fit2)
```

```
##
## Call:
## lm(formula = sqrt_y ~ x, data = data1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.39185 -0.30576 -0.03875  0.25378  0.81027
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.822e-01  1.299e-01   4.481 4.22e-05 ***
## x           9.529e-04  9.824e-05   9.699 3.61e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.464 on 51 degrees of freedom
## Multiple R-squared:  0.6485, Adjusted R-squared:  0.6416
## F-statistic: 94.08 on 1 and 51 DF,  p-value: 3.614e-13
```

```r
autoplot(Fit2)
```

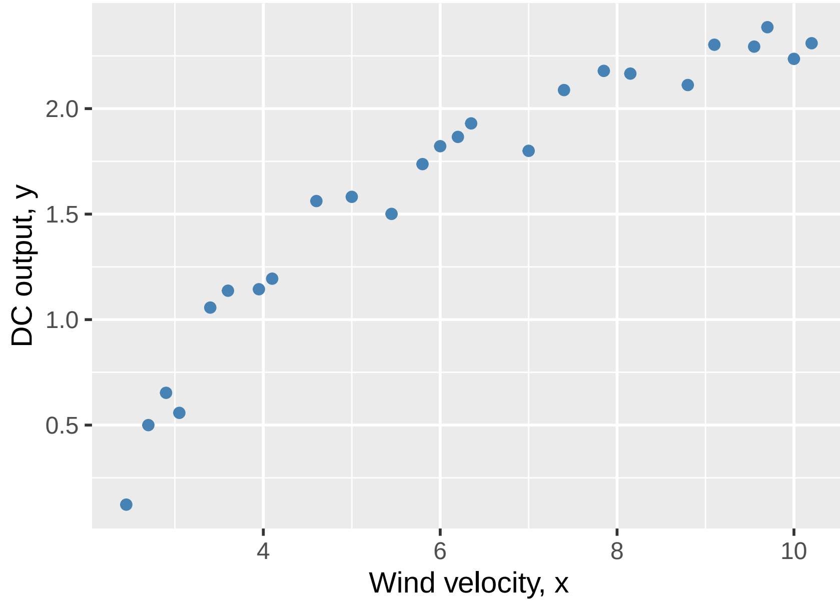


Example 5.2 The Windmill Data

```
data2 = read.table("D:\\CSUSB\\Fall 2021\\MATH 4360\\RNotes\\ex52.txt", header = TRUE)
head(data2)
```

```
##   Observation    x     y
## 1           1  5.0 1.582
## 2           2  6.0 1.822
## 3           3  3.4 1.057
## 4           4  2.7 0.500
## 5           5 10.0 2.236
## 6           6  9.7 2.386
```

```
n = nrow(data2)
ggplot(data2, aes(x = x, y = y)) +
  geom_point(color= "steelblue") +
  labs(x = "Wind velocity, x", y = "DC output, y", title = "")
```




```
Fit3 = lm(y ~ x, data = data2)
p = length(coef(Fit3))
summary(Fit3)
```

```
##
## Call:
## lm(formula = y ~ x, data = data2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.59869 -0.14099  0.06059  0.17262  0.32184
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.13088    0.12599   1.039     0.31
## x            0.24115    0.01905  12.659 7.55e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2361 on 23 degrees of freedom
## Multiple R-squared:  0.8745, Adjusted R-squared:  0.869
## F-statistic: 160.3 on 1 and 23 DF,  p-value: 7.546e-12
```

Inspection of the scatter diagram indicates that the relationship between DC output (y) and wind velocity (x) may be nonlinear. However, we initially fit a straight - line model to the data. The regression model is

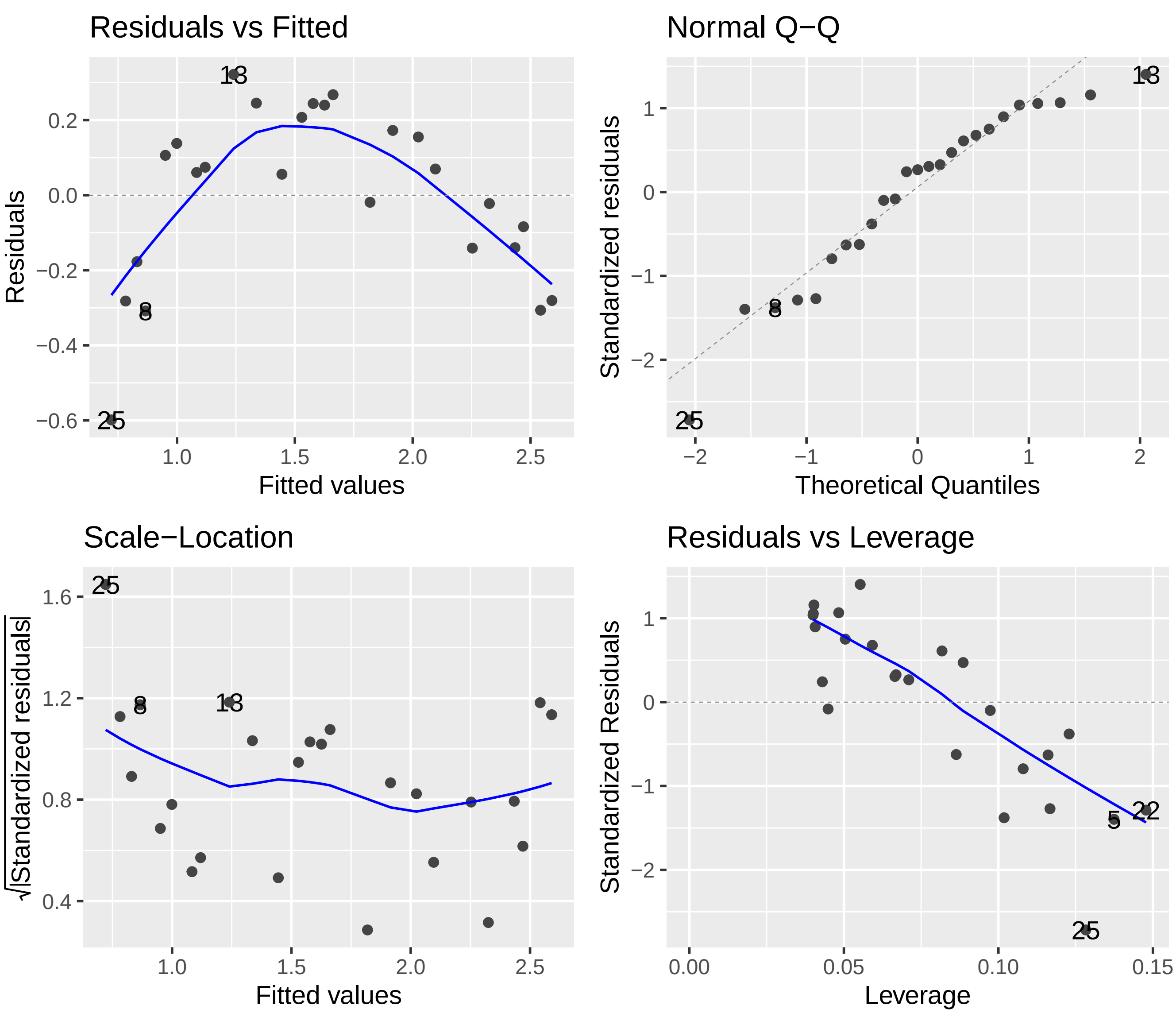
$$\hat{y} = 0.1309 + 0.2411x$$

```
anova_fit = anova(Fit3)
tab = as.table(cbind(
  'SS' = c('Regression' = anova_fit[1, 2],
           'Residual' = anova_fit[2, 2],
           'Total' = sum(anova_fit[1:2, 2])),
  'Df' = c( anova_fit[1, 1],
            anova_fit[2, 1],
            sum(anova_fit[1:2, 1])),
  'MS' = c( anova_fit[1, 3],
```

```
               anova_fit[2, 2] / anova_fit[2, 1],
               NA),
   'F-Test' = c( anova_fit[1, 3]/anova_fit[2, 3],
                    NA,
                    NA)
))
round(tab, 4)
```

```
##                   SS        Df        MS   F-Test
## Regression    8.9296    1.0000    8.9296 160.2571
## Residual      1.2816   23.0000    0.0557
## Total        10.2112   24.0000
```

```
autoplot(Fit3)
```



- The $R-$student residuals versus the fitted values $\hat{y}$ plot suggests that as wind speed increases, DC output approaches an upper limit of approximately 2.5.
- A more reasonable model for the windmill data that incorporates an upper asymptote would be

$$y = \beta_0 + \beta_1 (1/x) + \epsilon$$

```r
data2$inv_x = 1/(data2$x)
Fit4 = lm(y ~ inv_x, data = data2)
p = length(coef(Fit4))
summary(Fit4)
```

```
##
## Call:
## lm(formula = y ~ inv_x, data = data2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.20547 -0.04940  0.01100  0.08352  0.12204
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2.9789     0.0449   66.34   <2e-16 ***
## inv_x        -6.9345     0.2064  -33.59   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09417 on 23 degrees of freedom
## Multiple R-squared:   0.98,  Adjusted R-squared:  0.9792
## F-statistic:  1128 on 1 and 23 DF,  p-value: < 2.2e-16
```

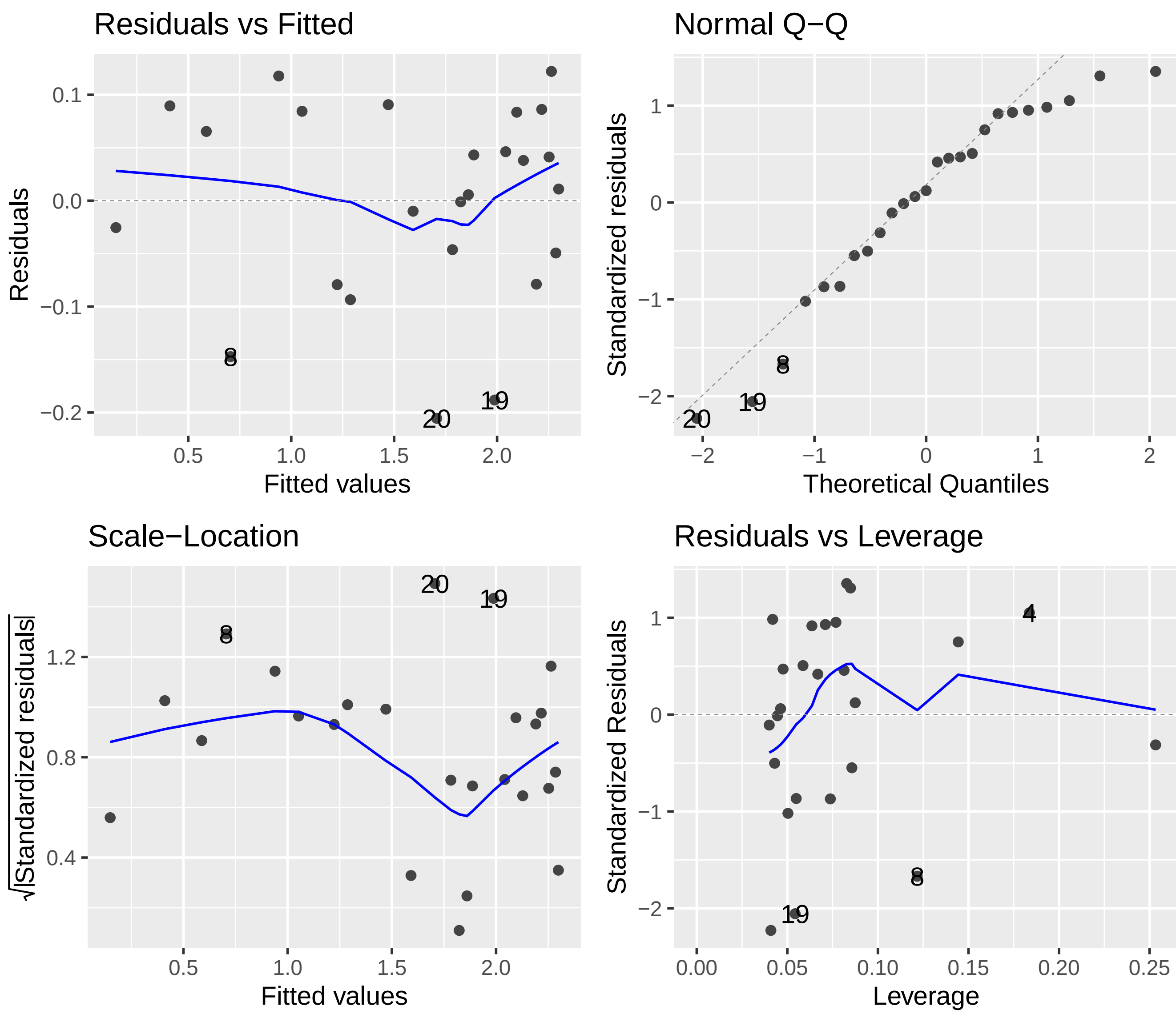
The fitted regression model is

$$\hat{y} = 2.9789 - 6.9345x', \quad \text{where} \quad x' = 1/x$$

```r
anova_fit = anova(Fit4)
tab = as.table(cbind(
  'SS' = c('Regression' = anova_fit[1, 2],
           'Residual' = anova_fit[2, 2],
           'Total' = sum(anova_fit[1:2, 2])),
  'Df' = c( anova_fit[1, 1],
            anova_fit[2, 1],
            sum(anova_fit[1:2, 1])),
  'MS' = c( anova_fit[1, 3],
            anova_fit[2, 2] / anova_fit[2, 1],
            NA),
  'F-Test' = c( anova_fit[1, 3]/anova_fit[2, 3],
                NA,
                NA)
))
round(tab, 4)
```

```
##                   SS        Df        MS    F-Test
## Regression   10.0072    1.0000   10.0072 1128.4327
## Residual      0.2040   23.0000    0.0089
## Total        10.2112   24.0000
```

```r
autoplot(Fit4)
```



Next, we reproduce Table 5.6

```
data3 = data.frame(x = data2$x, y = data2$y, Model1_yhat = fitted(Fit3),
                   Model1_res = resid(Fit3),  Model2_yhat = fitted(Fit4), Model2_res = resid(Fit3))
data3_sort = data3[order(data3$x),]
data3_sort
```

```
##        x     y Model1_yhat  Model1_res Model2_yhat  Model2_res
## 25  2.45 0.123   0.7216899 -0.59868986   0.1484327 -0.59868986
## 4   2.70 0.500   0.7819771 -0.28197708   0.4105093 -0.28197708
## 11  2.90 0.653   0.8302069 -0.17720685   0.5876369 -0.17720685
## 8   3.05 0.558   0.8663792 -0.30837918   0.7052381 -0.30837918
## 3   3.40 1.057   0.9507813  0.10621871   0.9392874  0.10621871
## 16  3.60 1.137   0.9990111  0.13798894   1.0525970  0.13798894
## 24  3.95 1.144   1.0834132  0.06058683   1.2232786  0.06058683
## 23  4.10 1.194   1.1195855  0.07441450   1.2875072  0.07441450
## 13  4.60 1.562   1.2401599  0.32184007   1.4713499  0.32184007
## 1   5.00 1.582   1.3366195  0.24538052   1.5919507  0.24538052
## 20  5.45 1.501   1.4451365  0.05586353   1.7064662  0.05586353
## 14  5.80 1.737   1.5295386  0.20746142   1.7832486  0.20746142
## 2   6.00 1.822   1.5777683  0.24423165   1.8231023  0.24423165
## 10  6.20 1.866   1.6259981  0.24000188   1.8603848  0.24000188
## 12  6.35 1.930   1.6621705  0.26782955   1.8868055  0.26782955
```

```
## 19  7.00 2.800   1.8189172 -0.01891722   1.9882106 -0.01891722
## 15  7.40 2.088   1.9153768  0.17262323   2.0417592  0.17262323
## 17  7.85 2.179   2.0238938  0.15510624   2.0954783  0.15510624
## 9   8.15 2.166   2.0962384  0.06976158   2.1279955  0.06976158
## 18  8.80 2.112   2.2529852 -0.14098518   2.1908434 -0.14098518
## 21  9.10 2.303   2.3253298 -0.02232985   2.2168220 -0.02232985
## 7   9.55 2.294   2.4338468 -0.13984684   2.2527296 -0.13984684
## 6   9.70 2.386   2.4700192 -0.08401917   2.2639584 -0.08401917
## 5  10.00 2.236   2.5423638 -0.30636383   2.2854054 -0.30636383
## 22 10.20 2.310   2.5905936 -0.28059360   2.2990026 -0.28059360
```

Analytical methods for selecting a transformation on study variable: The Box-Cox method

- Suppose the normality and/or constant variance of the study variable y can be corrected through a power transformation on y .
- This means y is to be transformed as y^λ where λ is the parameter to be determined. For example, if 0.5, $\lambda = 0.5$ then the transformation is the square root and $\sqrt{y}$ is used as a study variable in place of y
- Now the linear regression model has parameters β, σ^2 , and λ. Box and Cox method tells how to estimate simultaneously the λ and parameters of the model using the method of maximum likelihood.

$$y^{(\lambda)} = \begin{cases} \dfrac{y^\lambda - 1}{\lambda y_*^{\lambda-1}} & ; \lambda \neq 0 \\ \\ y_* \ln y & ; \lambda = 0 \end{cases}$$

where

$$\ln y_* = \frac{1}{n}\sum_{i=1}^{n} \ln y_i$$

We fit the model

$$y^{(\lambda)} = X\beta + \epsilon$$

by least squares (or maximum likelihood).

Example 5.3 The Electric Utility Data

- We use the Box - Cox procedure to select a variance - stabilizing transformation. Let's find the values of $SS_{Res}(\lambda)$ for various values of λ.

- I wrote a simple `R` function `myboxcox()` which reproduce values given in Table 5.7

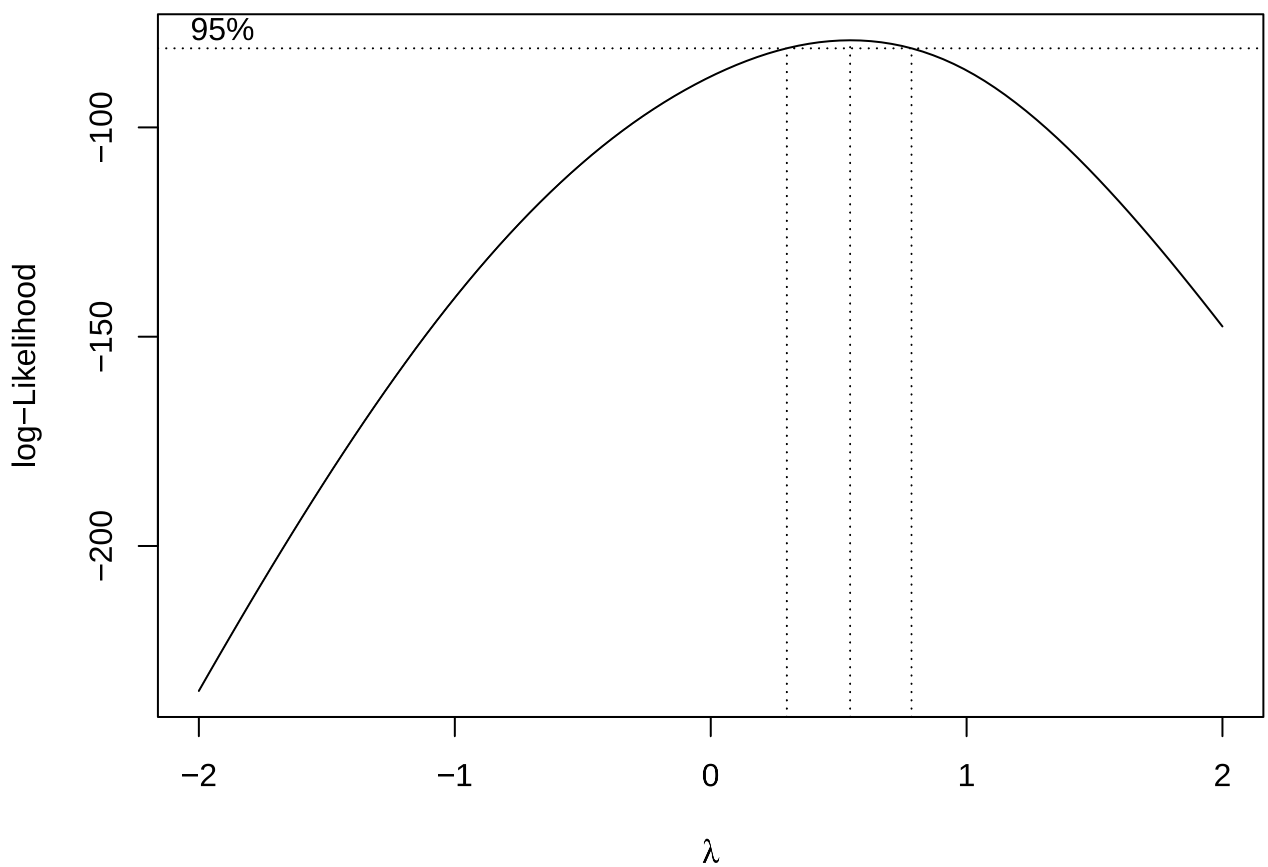
```
SS_res = NULL
myboxcox = function(lambda, y, x){
  for(i in 1: length(lambda)){
    lam = lambda[i]
    y_star = prod(y)^( 1/length(y))
    f1 = (y^lam - 1)/(lam * y_star^(lam - 1))
    f2 = y_star * log(y)
    SS_res[i] = ifelse(lam != 0, deviance(lm(f1 ~ x)), deviance(lm(f2 ~ x)) )
  }
 return(SS_res)
}
```

```
data1 = read.table("D:\\CSUSB\\Fall 2021\\MATH 4360\\RNotes\\ex51.txt", header = TRUE)
n = nrow(data1)
lambda = seq(-2, 2, by = 0.5)
SS_res1 = myboxcox(lambda, data1$y, data1$x)
data4 = data.frame(lam = lambda, SS_res = SS_res1 )
data4
```

```
##    lam      SS_res
## 1 -2.0 34100.60814
## 2 -1.5  5014.69890
## 3 -1.0   986.03414
## 4 -0.5   291.58165
## 5  0.0   134.09350
## 6  0.5    96.94928
## 7  1.0   126.86602
## 8  1.5   325.67911
## 9  2.0  1275.56063
```

- Alternatively, you can use the `R` function `boxcox()` from `MASS` package.

```r
Fit1 = lm(y ~ x, data = data1)
box_fit = boxcox(Fit1, plotit = TRUE, lambda = seq(-2, 2, by = 0.5)) # MASS r package
```




```r
bc_df = data.frame(lambda = box_fit$x, neg_lik = box_fit$y) # create a new data frame for lambda and neg_lik
sorted_bc_df = bc_df[order(-bc_df$neg_lik),]
head(sorted_bc_df, 20)
```

```
##       lambda   neg_lik
## 64 0.5454545 -79.19024
## 65 0.5858586 -79.24020
## 63 0.5050505 -79.24592
## 66 0.6262626 -79.39812
```

```
## 62 0.4646465 -79.40494
## 61 0.4242424 -79.66482
## 67 0.6666667 -79.66632
## 60 0.3838384 -80.02308
## 68 0.7070707 -80.04709
## 59 0.3434343 -80.47721
## 69 0.7474747 -80.54274
## 58 0.3030303 -81.02473
## 70 0.7878788 -81.15558
## 57 0.2626263 -81.66312
## 71 0.8282828 -81.88793
## 56 0.2222222 -82.38990
## 72 0.8686869 -82.74208
## 55 0.1818182 -83.20257
## 73 0.9090909 -83.72035
## 54 0.1414141 -84.09862
```

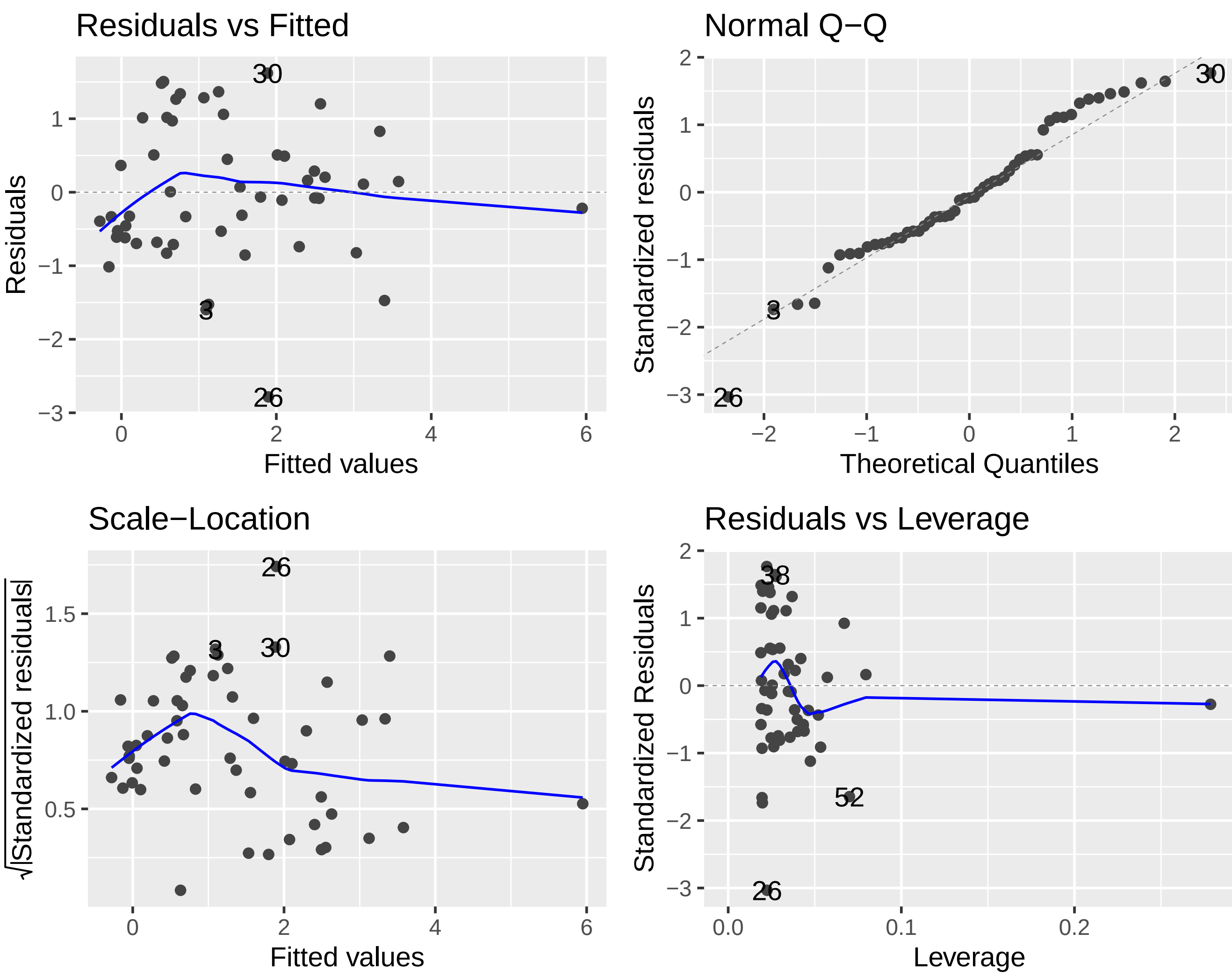
```
fit_cox = lm((((y^ 0.5) - 1) / 0.5) ~ x, data = data1)
summary(fit_cox)
```

```
##
## Call:
## lm(formula = (((y^0.5) - 1)/0.5) ~ x, data = data1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7837 -0.6115 -0.0775  0.5076  1.6205
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.8355482  0.2598620  -3.215  0.00226 **
## x            0.0019057  0.0001965   9.699 3.61e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9281 on 51 degrees of freedom
## Multiple R-squared:  0.6485, Adjusted R-squared:  0.6416
## F-statistic: 94.08 on 1 and 51 DF,  p-value: 3.614e-13
```

```
anova_fit = anova(fit_cox)
tab = as.table(cbind(
  'SS' = c('Regression' = anova_fit[1, 2],
           'Residual' = anova_fit[2, 2],
           'Total' = sum(anova_fit[1:2, 2])),
  'Df' = c( anova_fit[1, 1],
            anova_fit[2, 1],
            sum(anova_fit[1:2, 1])),
  'MS' = c( anova_fit[1, 3],
            anova_fit[2, 2] / anova_fit[2, 1],
            NA),
  'F-Test' = c( anova_fit[1, 3]/anova_fit[2, 3],
                NA,
                NA)
))
round(tab, 4)
```

```
##                  SS      Df      MS  F-Test
## Regression  81.0340  1.0000 81.0340 94.0781
## Residual    43.9288 51.0000  0.8613
## Total      124.9628 52.0000
```

```
autoplot(fit_cox)
```



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.