CS-E-106: Data Modeling - Midterm Exam

Question 2

Instructor: Hakan Gogtas
Submitted by: *Saurabh Kulkarni*

Due Date: *10/21/2019*

Solution 2:

(A)

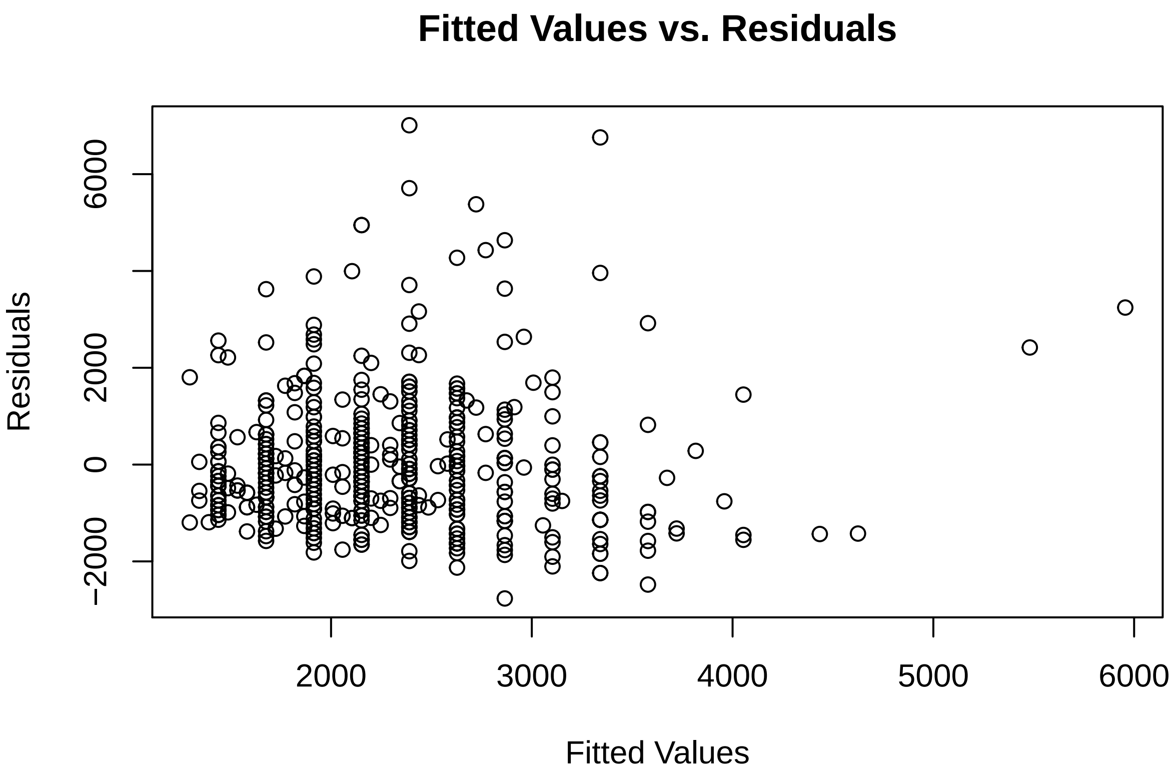
```
q2_data = read.csv("question2.csv")
lm_q2 = lm(y~x, data=q2_data)
summary(lm_q2)
```

```
##
## Call:
## lm(formula = y ~ x, data = q2_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2765.3  -889.8  -239.8   536.8  7010.2
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1201.124    123.325    9.74   <2e-16 ***
## x             47.549      4.652   10.22   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1352 on 494 degrees of freedom
## Multiple R-squared:  0.1745, Adjusted R-squared:  0.1729
## F-statistic: 104.5 on 1 and 494 DF,  p-value: < 2.2e-16
```

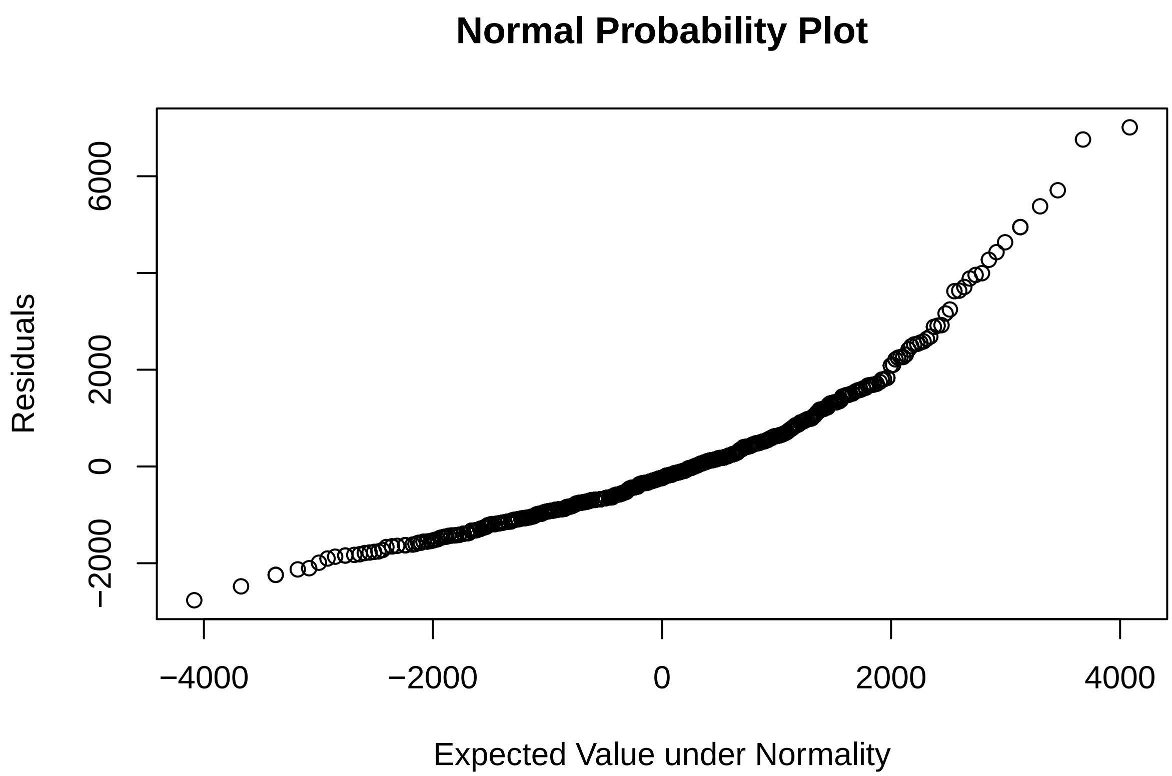
Regression Function: $y = 1201.124 + 47.549 * x$

```
build_residual_qq <- function(lm, df, rse){
  ei = lm$residuals
  fitted_values = lm$fitted.values

  par(mfrow=c(1,1))
  plot(fitted_values, ei, xlab="Fitted Values", ylab="Residuals")
  title(main="Fitted Values vs. Residuals")


  ri = rank(ei)
  n = nrow(df)
  zr = (ri-0.375)/(n+0.25)

  #residual standard error from summary(lm) above
  zr1 = rse*qnorm(zr)

  print(cor.test(zr1, ei))
```

```r
  plot(zr1, ei, xlab="Expected Value under Normality",ylab="Residuals")
  title(main="Normal Probability Plot")

}

build_residual_qq(lm=lm_q2, df=q2_data, rse=1352)
```




```
##
##  Pearson's product-moment correlation
##
## data:  zr1 and ei
## t = 63.43, df = 494, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9332385 0.9526287
## sample estimates:
##       cor
## 0.9437392
```



Interpretation:

Fitted vs. Residual Plot: The residual plot appears to be mostly equally spread and has no distinct patterns. We do see a few outliers. We can say that there is mostly a contant variance in the error term.

Normal Probability Plot: The plot is not linear, which means that the error is not in agreement with the normality.

(B)

Note: The question script only read: "Calculate the simultaneous 90% confidence interval for". Assuming we are supposed to calculate a 90% simultaneous confidence intervals for β_0 and β_1 using Bonferroni method.

```
confint(lm_q2, level=1-0.1/2)
```

```
##                   2.5 %    97.5 %
## (Intercept) 958.81911 1443.4296
## x            38.40798   56.6894
```

(C)

```
Xh = data.frame(x=c(85,90))
g = nrow(Xh)

alpha = 0.1
CI.New = predict(lm_q2, Xh, se.fit= TRUE, level = 1-alpha)
B = qt(1 -alpha / (2*g), lm_q2$df)
S = sqrt( g * qf( 1 -alpha, g, lm_q2$df))
spred = sqrt( CI.New$residual.scale^2 + (CI.New$se.fit)^2 ) # (2.38)

print(B)
```

```
## [1] 1.964778
```

```r
print(S)
```

```
## [1] 2.150977
```

Interpretation: We see that Bonferroni is more efficient, since it has tigher limits.

```r
pred_new_CI = t(
rbind(
"Xh" = array(t(Xh)),
"s.pred" = array(spred),
"fit" = array(CI.New$fit),
"lower.B" = array(CI.New$fit-B * spred),
"upper.B" = array(CI.New$fit+ B * spred))
)

pred_new_CI
```

```
##      Xh   s.pred      fit  lower.B  upper.B
## [1,] 85 1383.269 5242.763 2524.947 7960.580
## [2,] 90 1388.300 5480.507 2752.805 8208.208
```

Double-check:

```r
predict(lm_q2, Xh, se.fit= TRUE, interval = "prediction", level = 1-alpha/g)
```

```
## $fit
##        fit      lwr      upr
## 1 5242.763 2524.947 7960.580
## 2 5480.507 2752.805 8208.208
##
## $se.fit
##        1        2
## 294.4081 317.2062
##
## $df
## [1] 494
##
## $residual.scale
## [1] 1351.576
```

(D)

Brown-Forsythe Test

Note: Assuming $\alpha = 0.05$, since not specified in part (D).

Null Hypothesis: H_0: Error variance is constant Alternate Hypothesis: H_1: Error variance is not constant

```r
summary(q2_data$x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00   15.00   21.00   23.08   30.00  100.00
```

```r
ei = lm_q2$residuals
df = data.frame(cbind(q2_data$y,q2_data$x,ei))
df1 = df[df[,2]<=21,]
df2 = df[df[,2]>21,]

med1 = median(df1[,3])
```

```r
med2 = median(df2[,3])

#n1
n1 = nrow(df1)
print(n1)
```

```
## [1] 252
```

```r
#n2
n2 = nrow(df2)
print(n2)
```

```
## [1] 244
```

```r
d1 = abs(df1[,3]-med1)
d2 = abs(df2[,3]-med2)

#calculate means for our answer
mean_d1 = mean(d1)
print(mean_d1)
```

```
## [1] 818.3534
```

```r
mean_d2 = mean(d2)
print(mean_d2)
```

```
## [1] 1104.361
```

```r
s2 = (var(d1)*(n1-1)+var(d2)*(n2-1))/(n1+n2-2)
print(s2)
```

```
## [1] 938356.2
```

```r
#calculate s
s = sqrt(s2)
print(s)
```

```
## [1] 968.6879
```

```r
#testStastic = (mean.d1 - mean.d2) / (s * sqrt((1/n1)+1/n2)
testStastic = (mean_d1-mean_d2)/(s*sqrt((1/n1)+(1/n2)))
print(testStastic)
```

```
## [1] -3.287369
```

```r
t = qt(1-0.05/2, lm_q2$df.residual)
print(t)
```

```
## [1] 1.964778
```

Decision Rule:

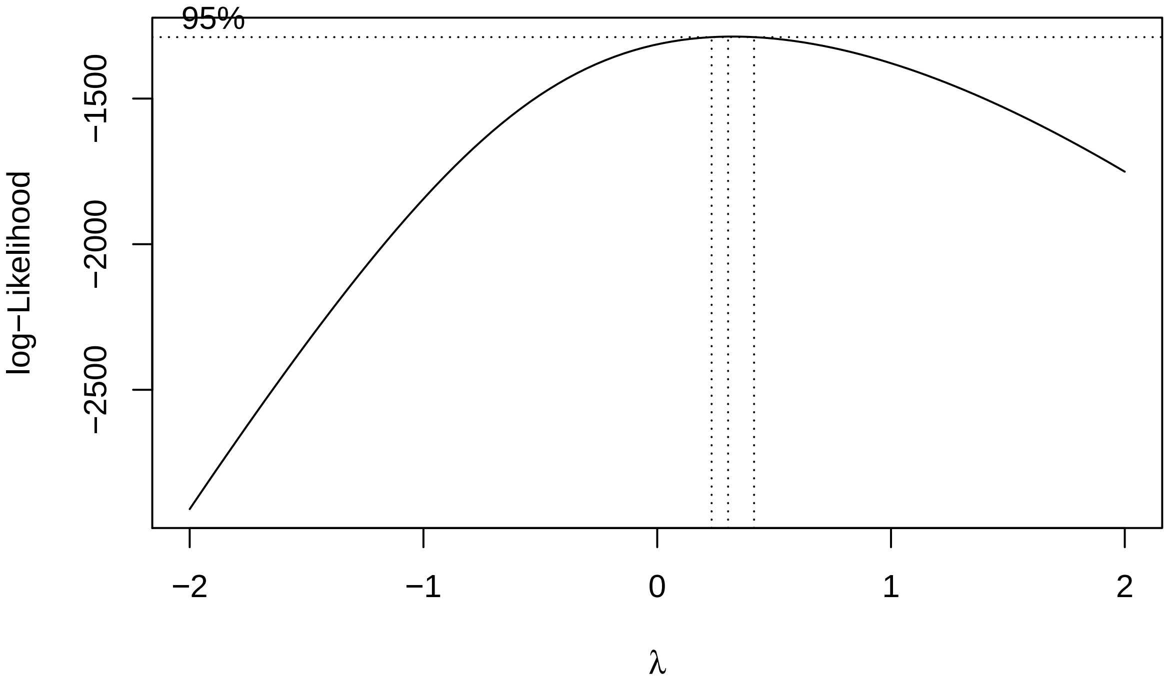
- If $|testStatistic| \leq t(1-\alpha/2, n-2)$, conclude H_0: constant error variance
- If $|testStatistic| > t(1-\alpha/2, n-2)$, conclude H_1: non-constant error variance

Result:

Since $|-3.287369| > 1.647944$ i.e. $|testStatistic| > t(1-\alpha/2, n-2)$, we conclude H_1. The error variance is not constant and thus varies with X.

(E)

```r
library(MASS)
par(mfrow=c(1,1))
boxcox(lm_q2)
```



Interpretation:

The suggested Y transformation with Box-Cox method is: $\lambda \approx 0$. Thus, we'll assume the suggested $\lambda = 0$ (as suggested in notes Ch.3, slide 77 - "a nearby lambda is easy to understand"), which implies the suggested transformation is: $Y' = log(Y)$.

```r
y1 = log(q2_data$y)
q2_data = cbind(q2_data, y1)
```

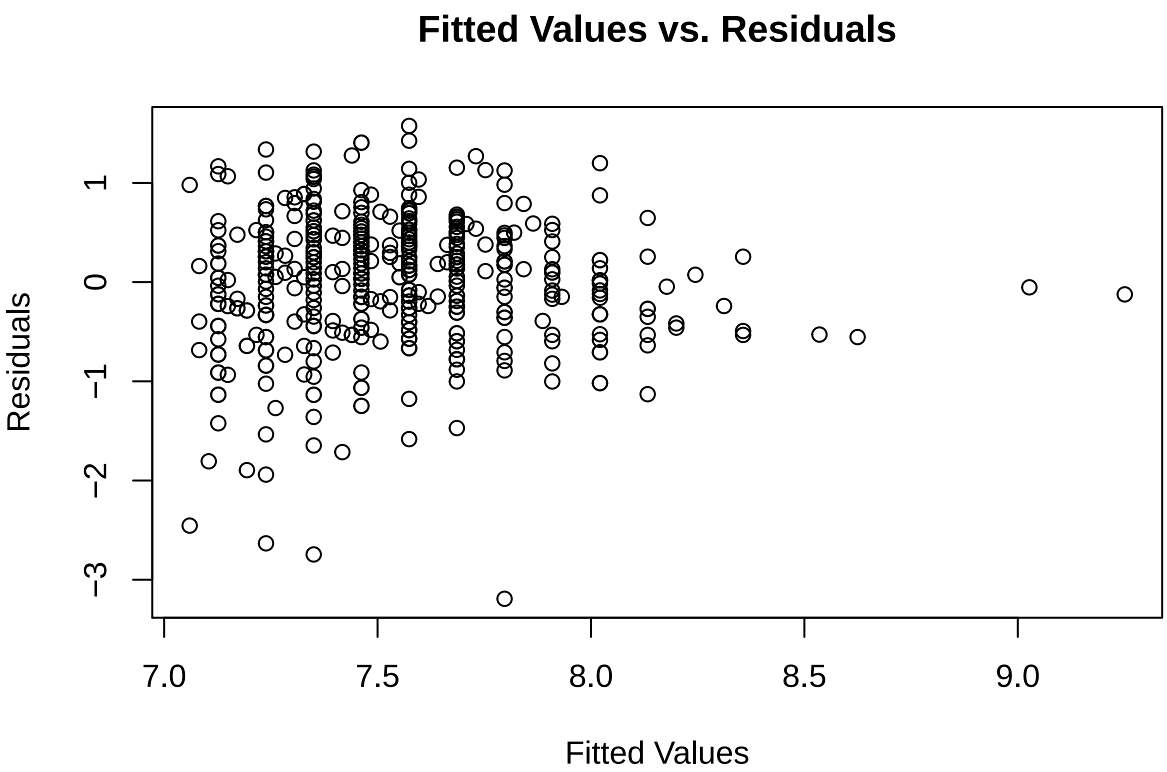
```r
lm_q2_t = lm(y1~x, data=q2_data)
summary(lm_q2_t)
```

```
##
## Call:
## lm(formula = y1 ~ x, data = q2_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1924 -0.3309  0.0536  0.4098  1.5745
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.015047   0.058037  120.87   <2e-16 ***
## x           0.022357   0.002189   10.21   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6361 on 494 degrees of freedom
## Multiple R-squared:  0.1743, Adjusted R-squared:  0.1726
```

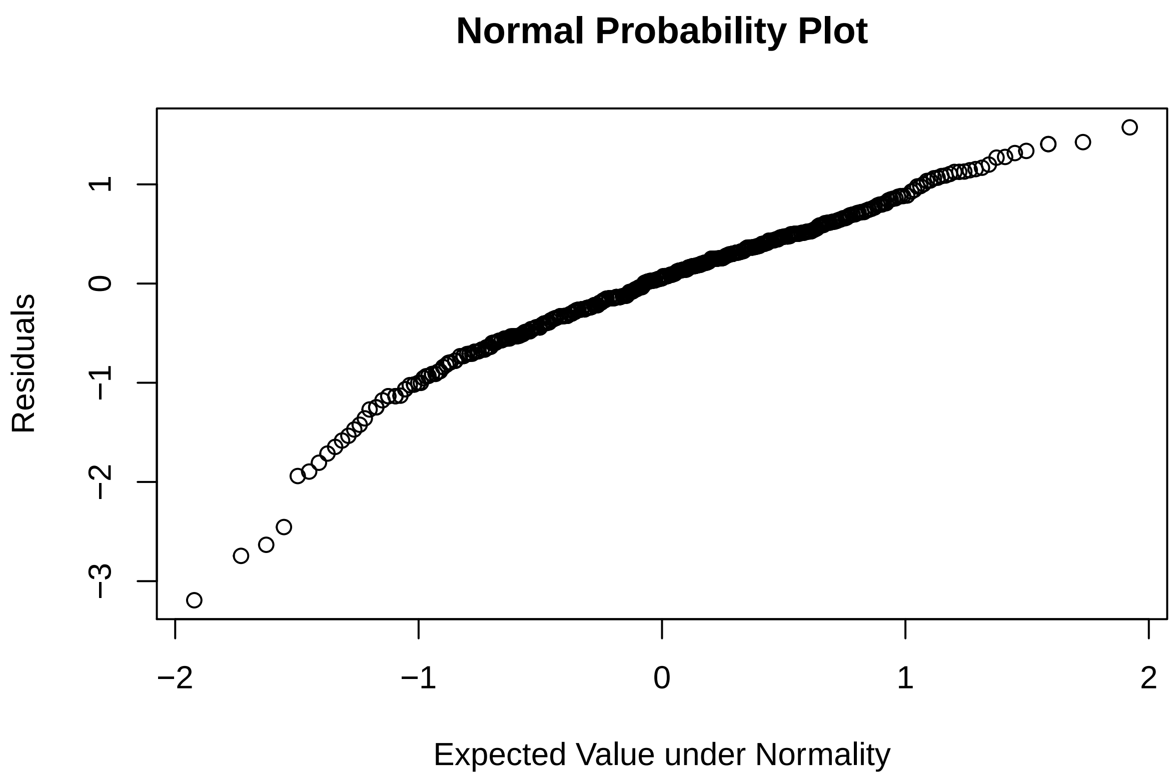
```
## F-statistic: 104.3 on 1 and 494 DF,  p-value: < 2.2e-16
```

The regression function using the transformed data $= log(y) = 7.015047 + 0.022357 * x$ or $y = \exp(7.015047 + 0.022357 * x)$

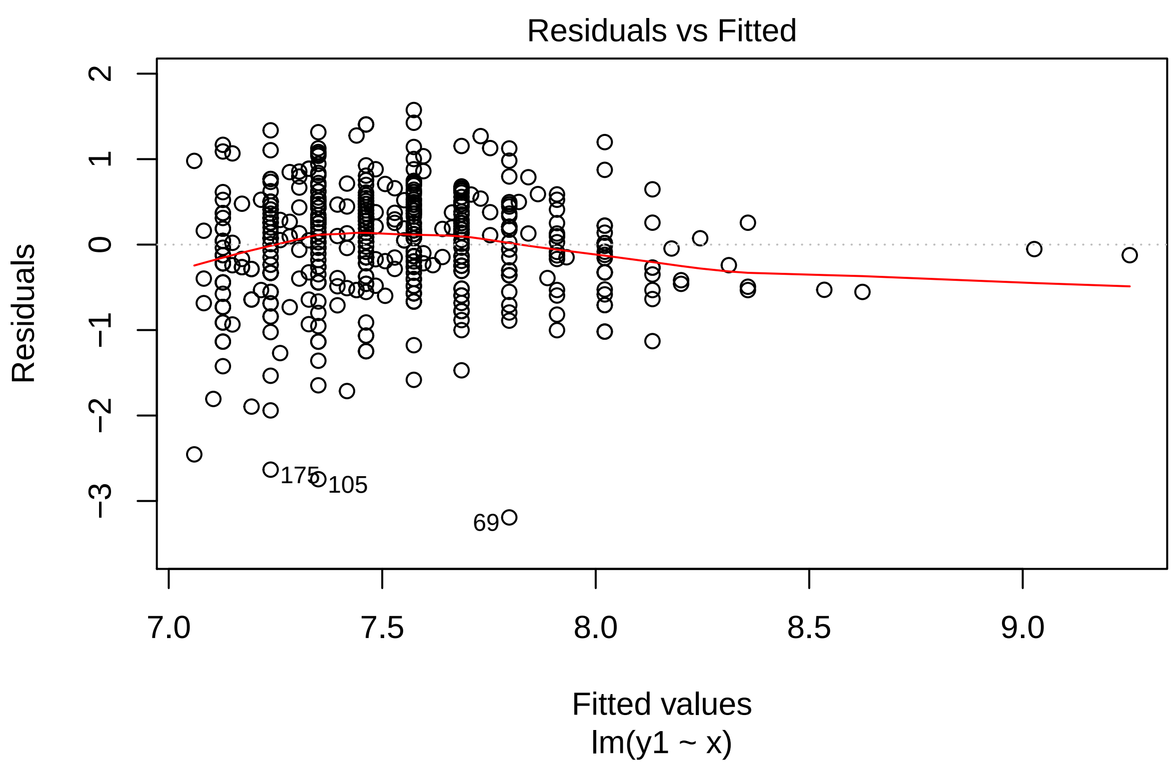
```
build_residual_qq(lm=lm_q2_t, df=q2_data, rse=0.6361)
```

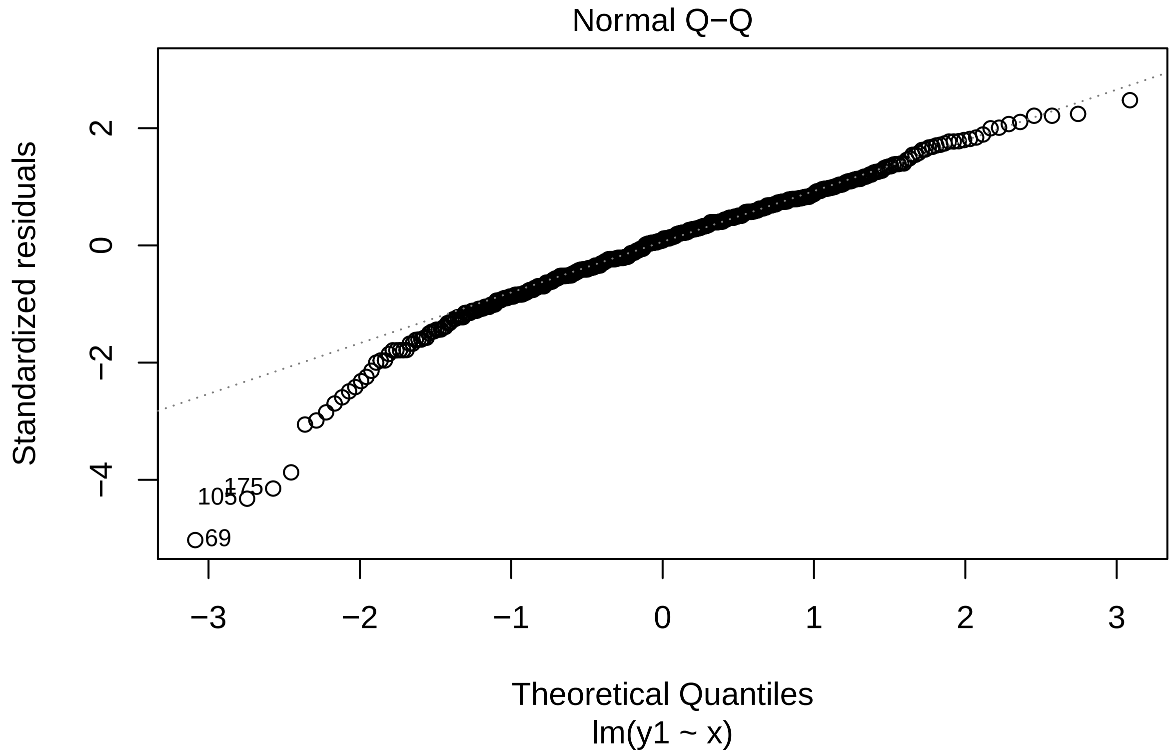



```
##
##  Pearson's product-moment correlation
##
## data:  zr1 and ei
## t = 111.39, df = 494, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9769787 0.9837716
## sample estimates:
##       cor
## 0.9806684
```

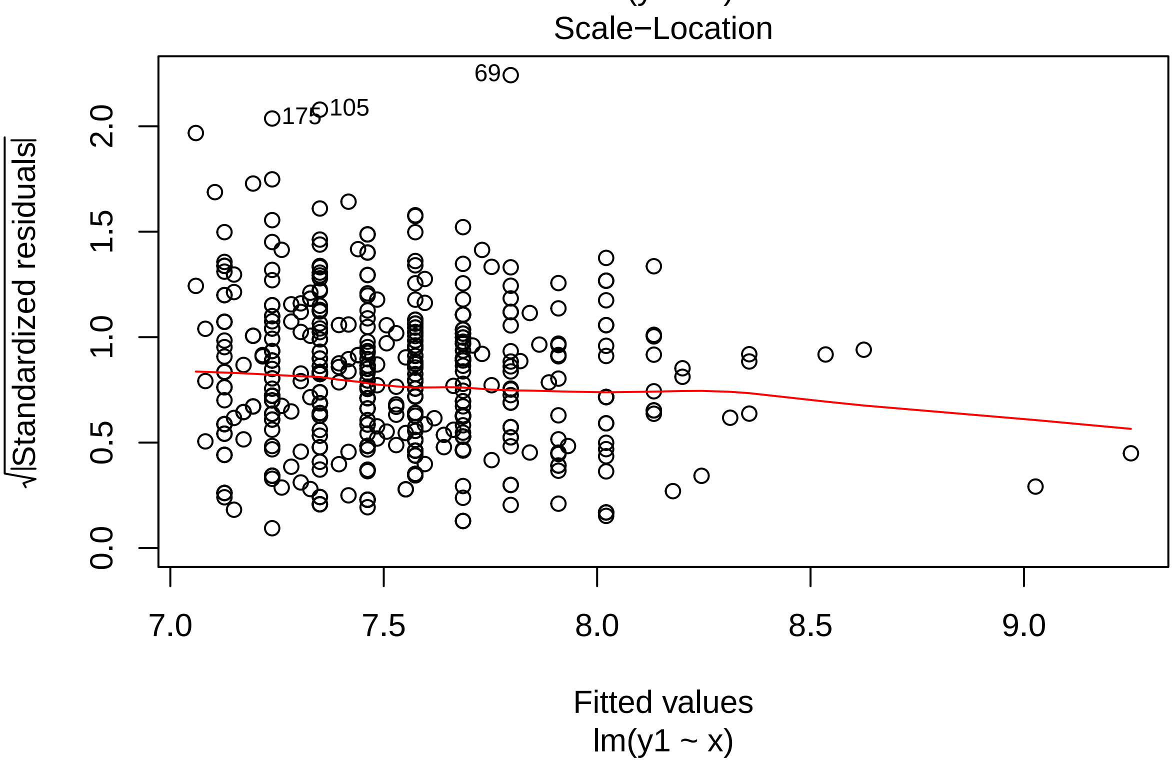

Normal Probability Plot
Residuals
1
0
−1
−2
−3
−2
−1
0
1
2
Expected Value under Normality

```r
plot(lm_q2_t)
```

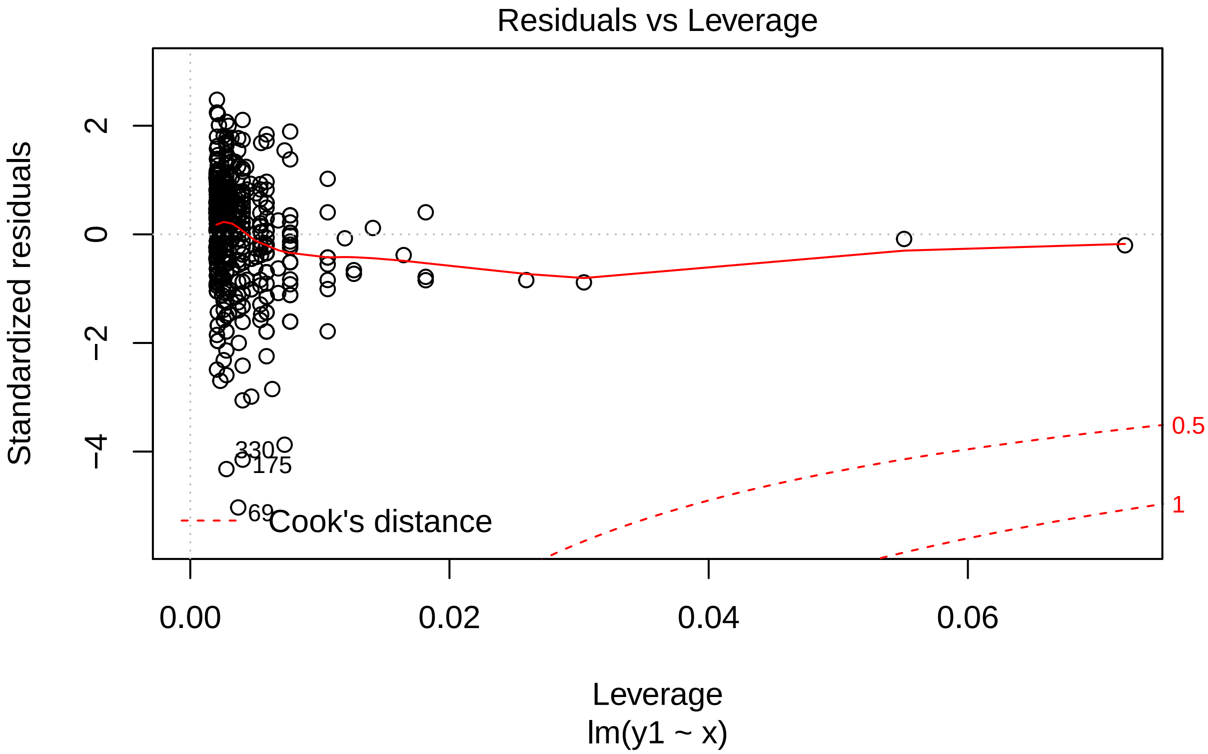

Residuals vs Fitted
Residuals
2
1
0
−1
−2
−3
175
105
69
7.0
7.5
8.0
8.5
9.0
Fitted values
lm(y1 ~ x)



Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(y1 ~ x)
105
175
69



Scale−Location
√|Standardized residuals|
Fitted values
lm(y1 ~ x)
69
175
105



Interpretation:

Fitted vs. Residual Plot: The residual plot appears to be mostly equally spread and has no distinct patterns. We still do see a few outliers. We can say that there is mostly a contant variance in the error term.

Normal Probability Plot: The plot is mostly linear, which means that the error is mostly in agreement with the normality. This could be due to the approximation we did of the λ value we got using Box-Cox method.