MATH 3080 Lab Project 12

Aini Liang

April 05, 2018

Remember: I expect to see commentary either in the text, in the code with comments created using `#` *, or (preferably) both!* ***Failing to do so may result in lost points!***

Problem 1

Polyester fiber ropes are increasingly being used as components of mooring lines for offshore structures in deep water.The authors of the paper “Quantifying the Residual CreepLife of Polyester Mooring Ropes” (IntL J. of Offshore and Polar Exploration 2005: 223—228) used the accompanying data as a basis for studying how time to failure (hr) depended on load (% of breaking load)

L: 77.7 77.8 77.9 77.8 85.5 85.5 89.2 89.3 73.1 85.5 89.2 85.5 89.2 85.5 89.2 82.3 82.0 82.3

T: 5.067 552.056 127.809 7.611 .124 .077 .008 .013 49.439 .503 362 9.930 .677 5.322 .289 53.079 7.625 155.299

A linear regression of log(time) versus load was fit. The investigators were particularly interested in estimating the slope of the true regression line relating these variables. Investigate the quality of the fit, estimate the slope, and predict time to failure when load is 80, in a way that conveys information about reliability and precision.

In other words: `log(T) ~ L`

```
# Your code here

T = c(5.067, 552.056, 127.809, 7.611, 0.124, 0.077, 0.008, 0.013, 49.439, 0.503,
    362, 9.93, 0.677, 5.322, 0.289, 53.079, 7.625, 155.299)
L = c(77.7, 77.8, 77.9, 77.8, 85.5, 85.5, 89.2, 89.3, 73.1, 85.5, 89.2, 85.5,
    89.2, 85.5, 89.2, 82.3, 82, 82.3)
a = log(T)
f = lm(a ~ L)
summary(f)
```

```
##
## Call:
## lm(formula = a ~ L)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8732 -1.9421 -0.0982  1.6364  6.8467
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  34.9328    11.5102   3.035  0.00788 **
## L            -0.4023     0.1375  -2.927  0.00988 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.84 on 16 degrees of freedom
## Multiple R-squared:  0.3487, Adjusted R-squared:  0.3079
## F-statistic: 8.565 on 1 and 16 DF,  p-value: 0.009881
```

Problem 2

The following data resulted from an experiment to assess the potential of unburnt colliery spoil as a medium for plant growth. The variables are `x = acid extractable cations` and `y = exchangeable acidity/total cation exchange capacity` (“Exchangeable Acidity in Unbumt Colliery Spoil,” Nature, 1969: 161):

x: -23 -5 16 26 30 38 52 58 67 81 96 100 113

y: 1.50 1.46 1.32 1.17 .96 .78 .77 .91 .78 .69 .52 .48 .55

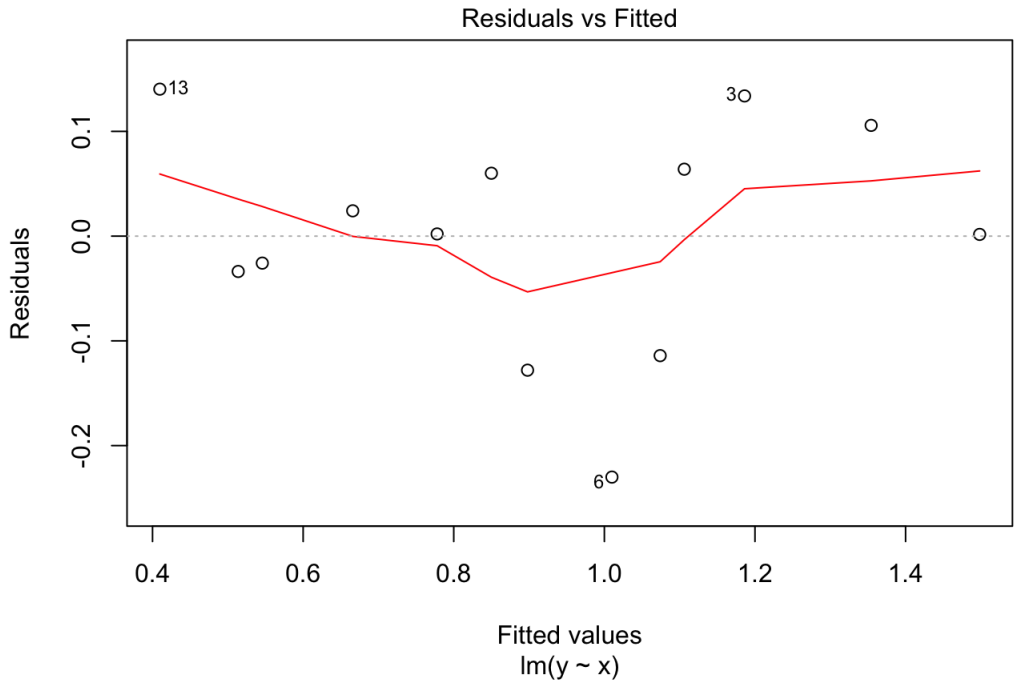
Find the best model that you can to fit the data.

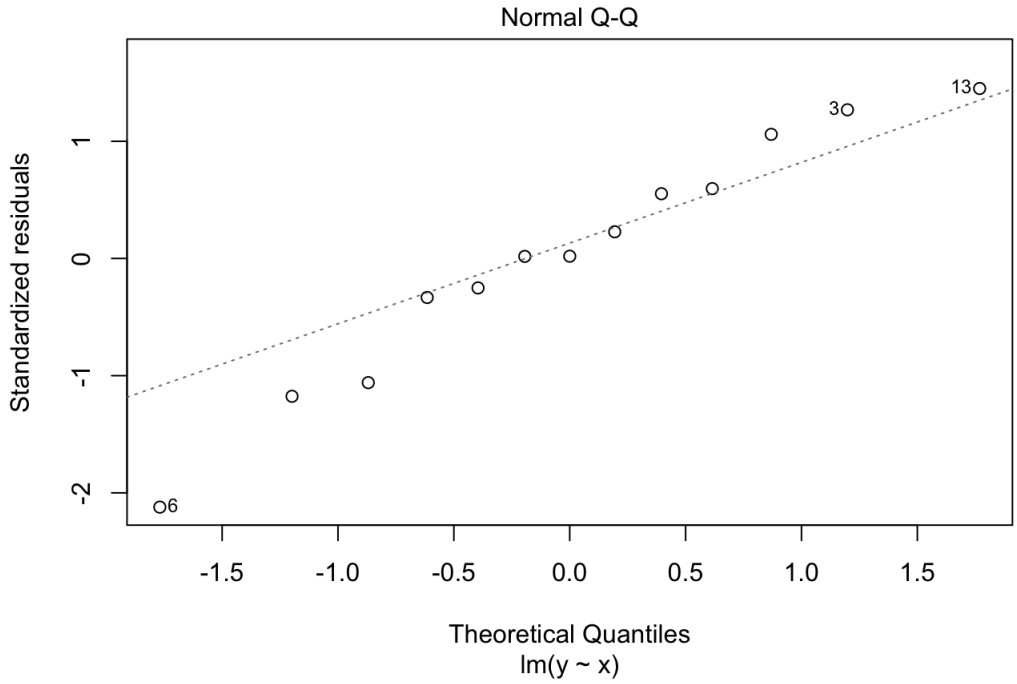
Plot the data and the three different models all on the same plot. Which one would you use? Defend your choice.

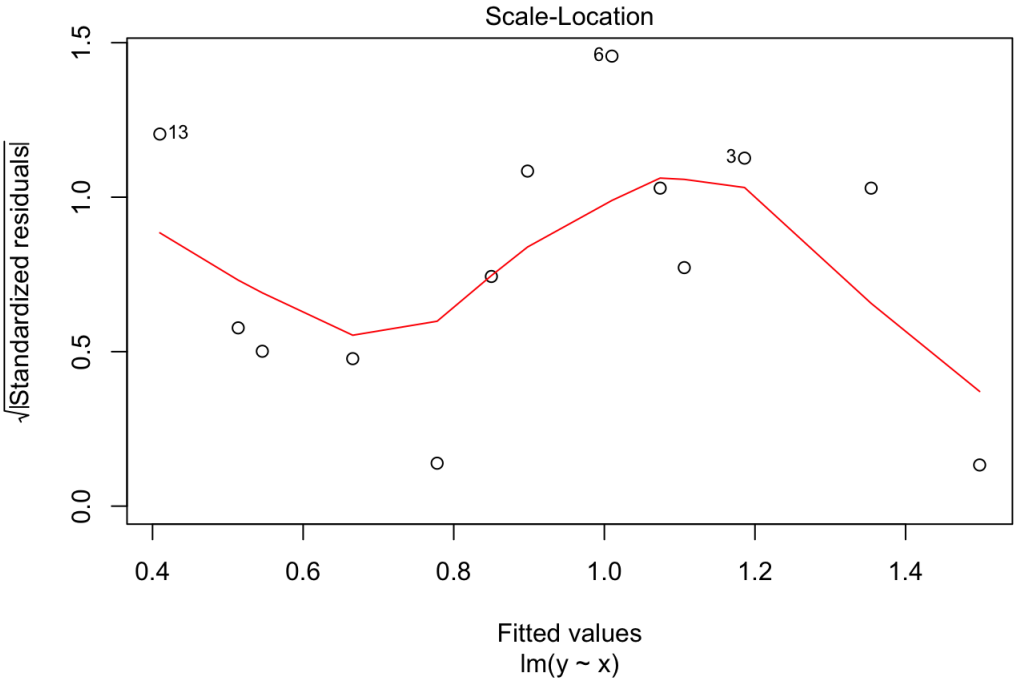
```r
# Your code here
x = c(-23, -5, 16, 26, 30, 38, 52, 58, 67, 81, 96, 100, 113)
y = c(1.5, 1.46, 1.32, 1.17, 0.96, 0.78, 0.77, 0.91, 0.78, 0.69, 0.52, 0.48,
    0.55)
fit = lm(y ~ x)
summary(fit)
```

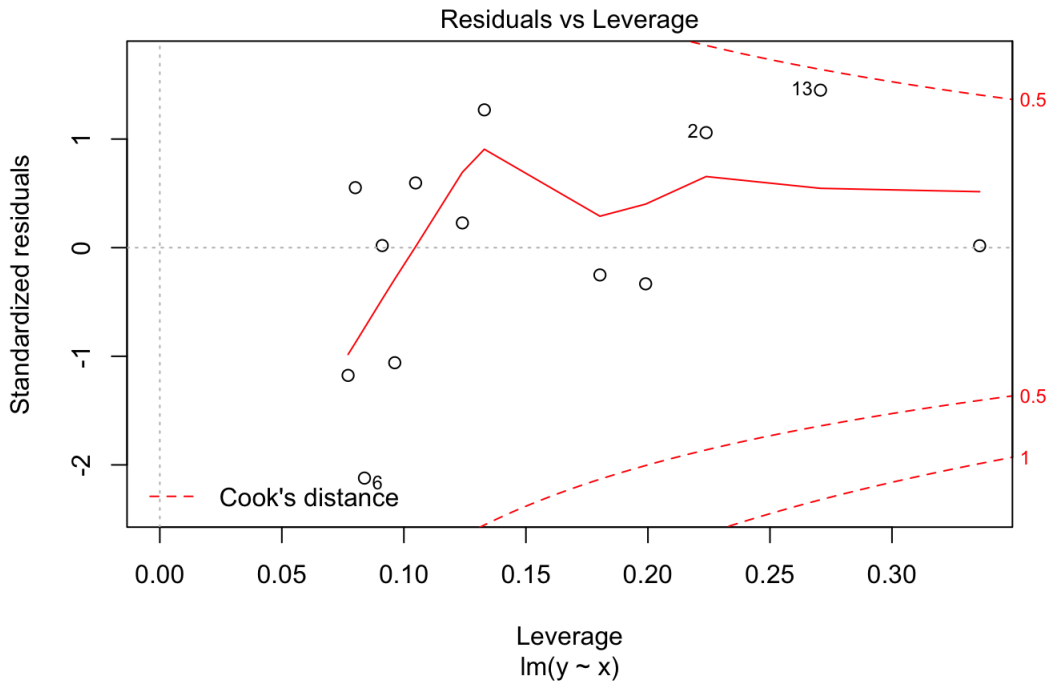
```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.230059 -0.033751  0.002085  0.063881  0.140313
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.3142480  0.0504501   26.05 3.09e-11 ***
## x           -0.0080050  0.0007906  -10.12 6.53e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1133 on 11 degrees of freedom
## Multiple R-squared:  0.9031, Adjusted R-squared:  0.8943
## F-statistic: 102.5 on 1 and 11 DF,  p-value: 6.53e-07
```

```r
plot(fit)  # which transformation would be appropriate?
```

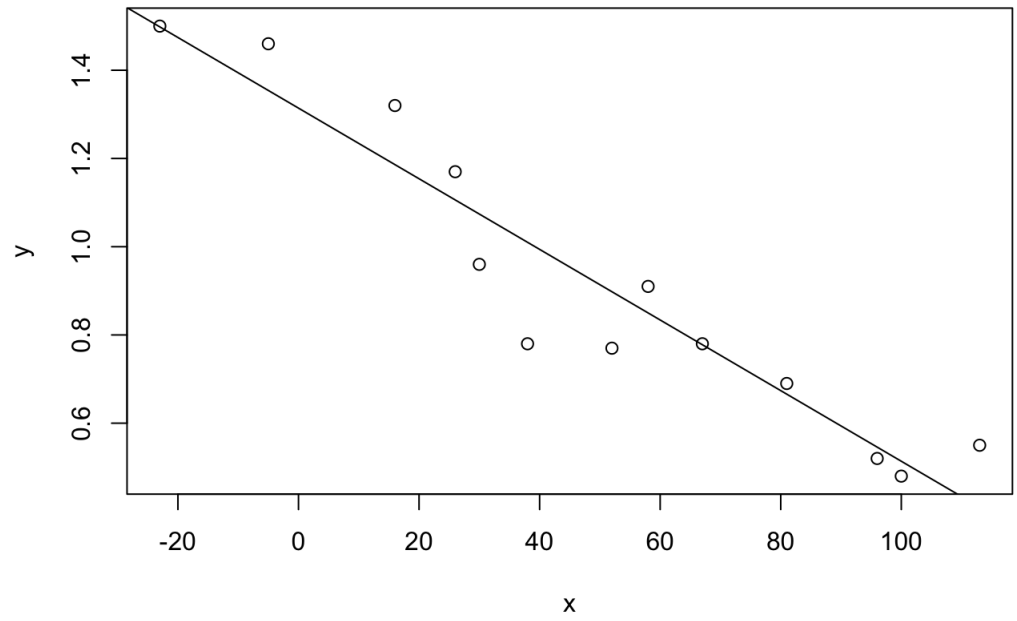



Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(y ~ x)
13
3
6
-2
-1
0
1
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5


Scale-Location
√|Standardized residuals|
Fitted values
lm(y ~ x)
13
6
3
0.0
0.5
1.0
1.5
0.4
0.6
0.8
1.0
1.2
1.4



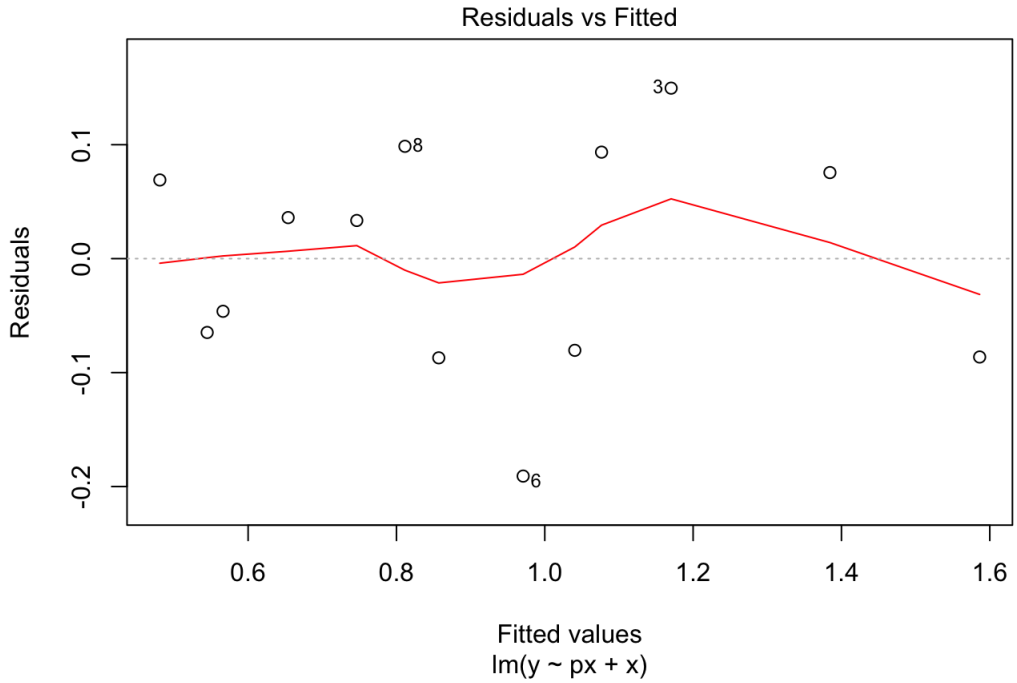

```
plot(x, y)
abline(fit)
```

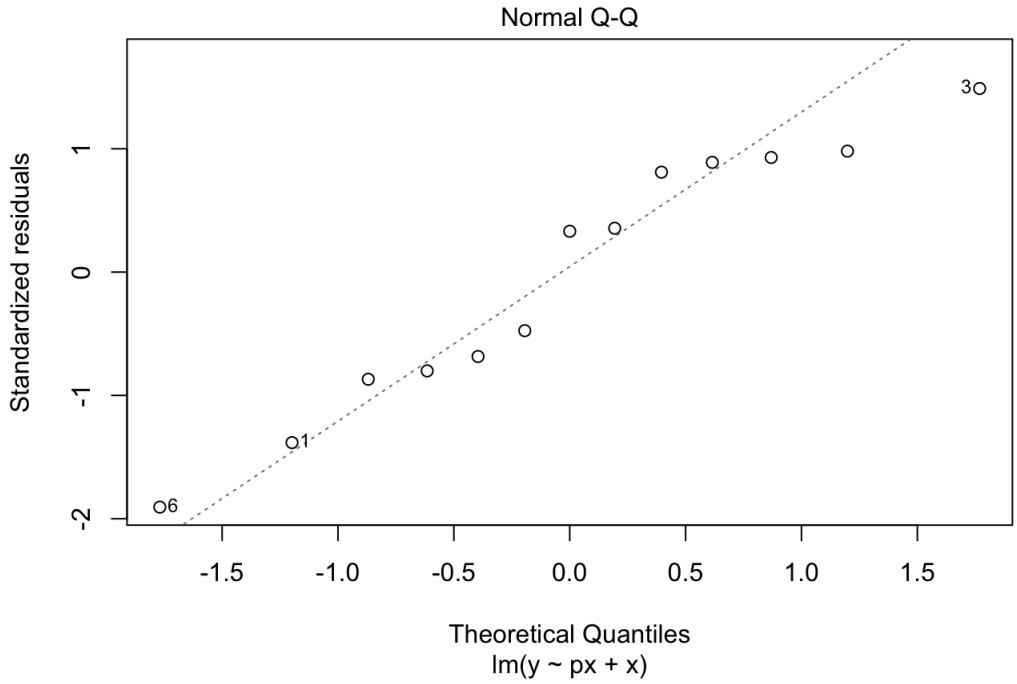


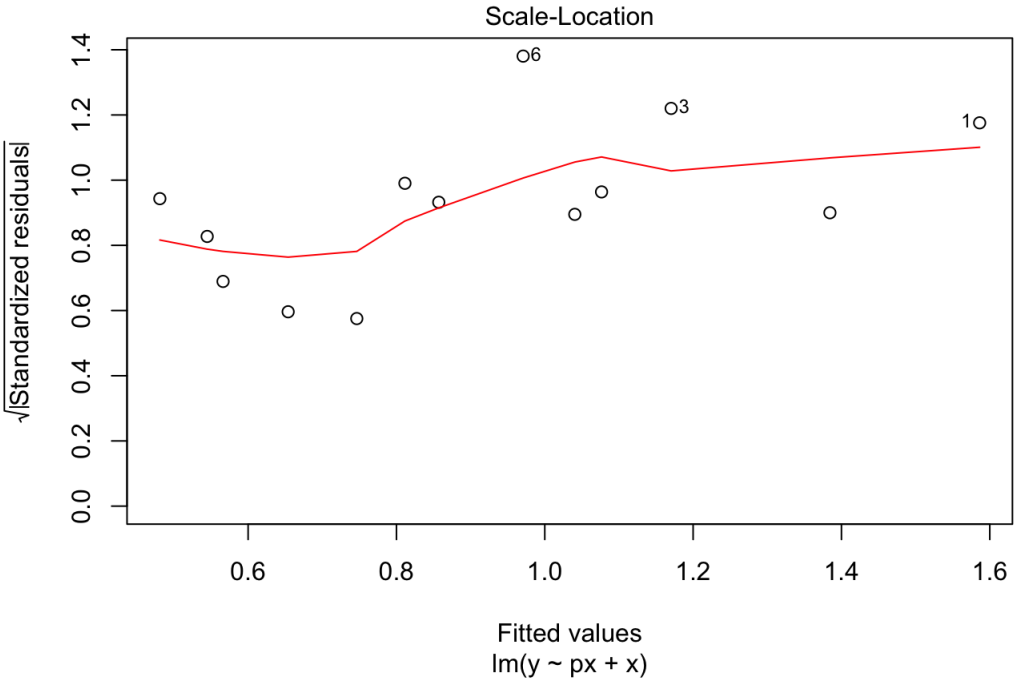

```
px = x^2
fit2.0 = lm(y ~ px + x)
summary(fit2.0)
```

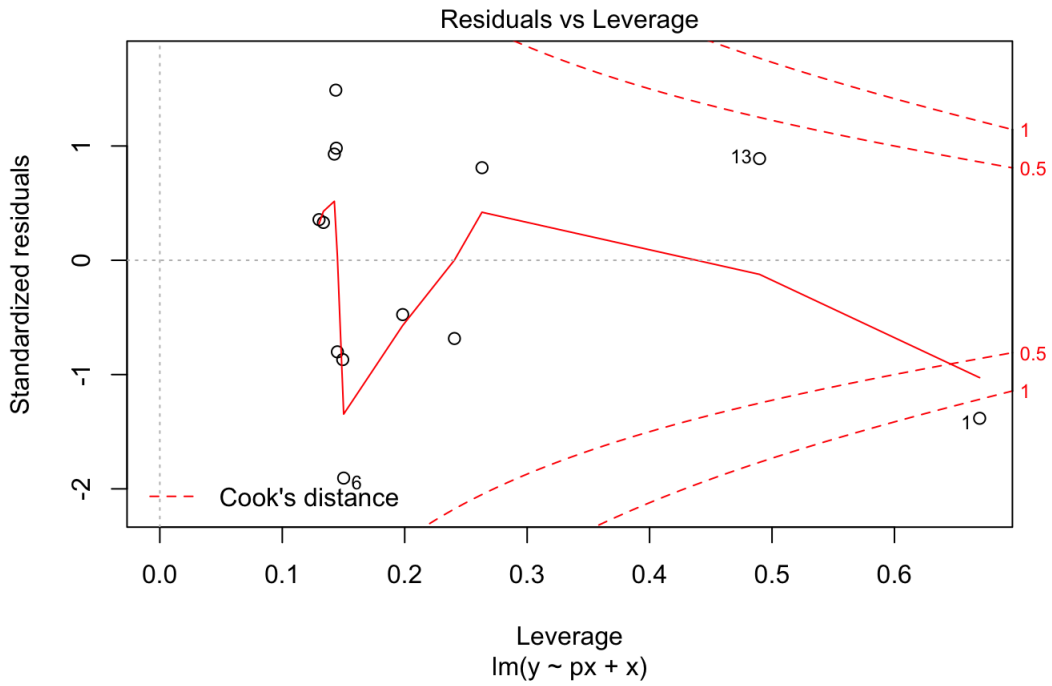
```
##
## Call:
## lm(formula = y ~ px + x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.19084 -0.08047  0.03348  0.07554  0.14963
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.331e+00  4.989e-02  26.685 1.26e-10 ***
## px           2.616e-05  1.866e-05   1.402  0.19118
## x           -1.048e-02  1.923e-03  -5.452  0.00028 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1086 on 10 degrees of freedom
## Multiple R-squared:  0.919,  Adjusted R-squared:  0.9028
## F-statistic: 56.74 on 2 and 10 DF,  p-value: 3.484e-06
```

```
plot(fit2.0)
```

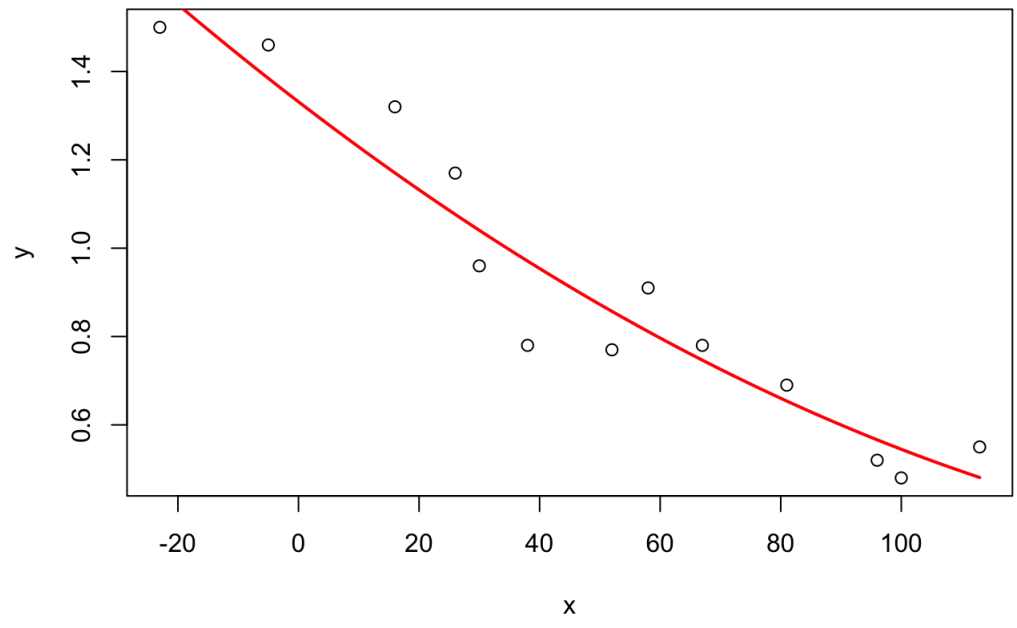



Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(y ~ px + x)
3
1
6
-2
-1
0
1
-1.5
-1.0
-0.5
0.0
0.5
1.0
1.5


Scale-Location
√|Standardized residuals|
Fitted values
lm(y ~ px + x)
6
3
1
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0.6
0.8
1.0
1.2
1.4
1.6



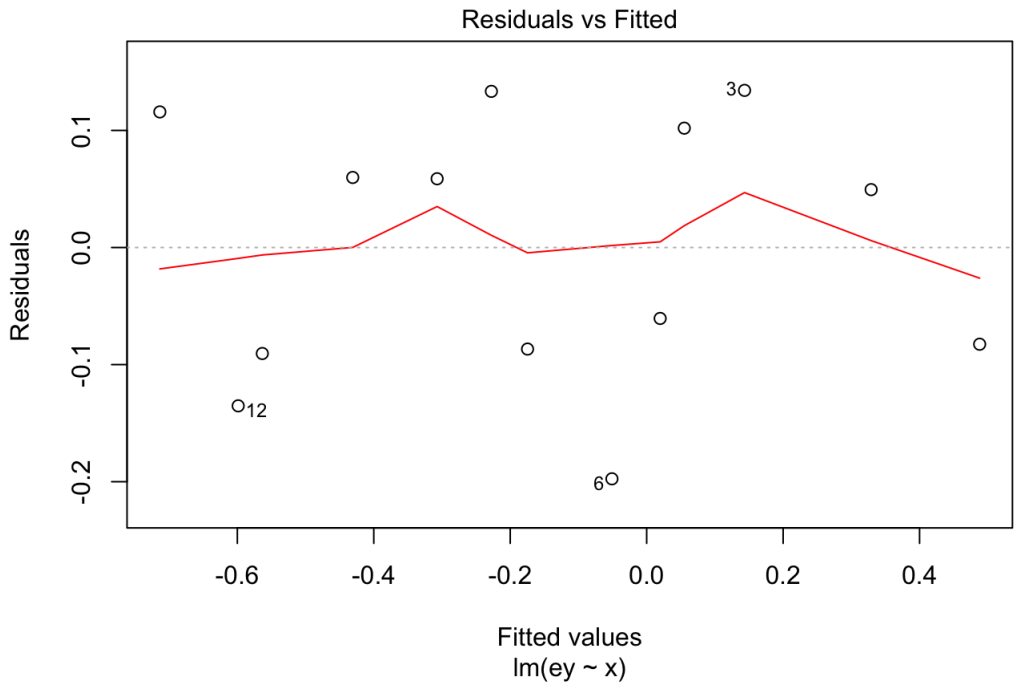

```
x.2 = seq(min(x), max(x), length = 1000)
plot(x, y)
points(x.2, predict(fit2.0, data.frame(px = x.2^2, x = x.2)), type = "l", col = "red",
    lwd = "2")
```

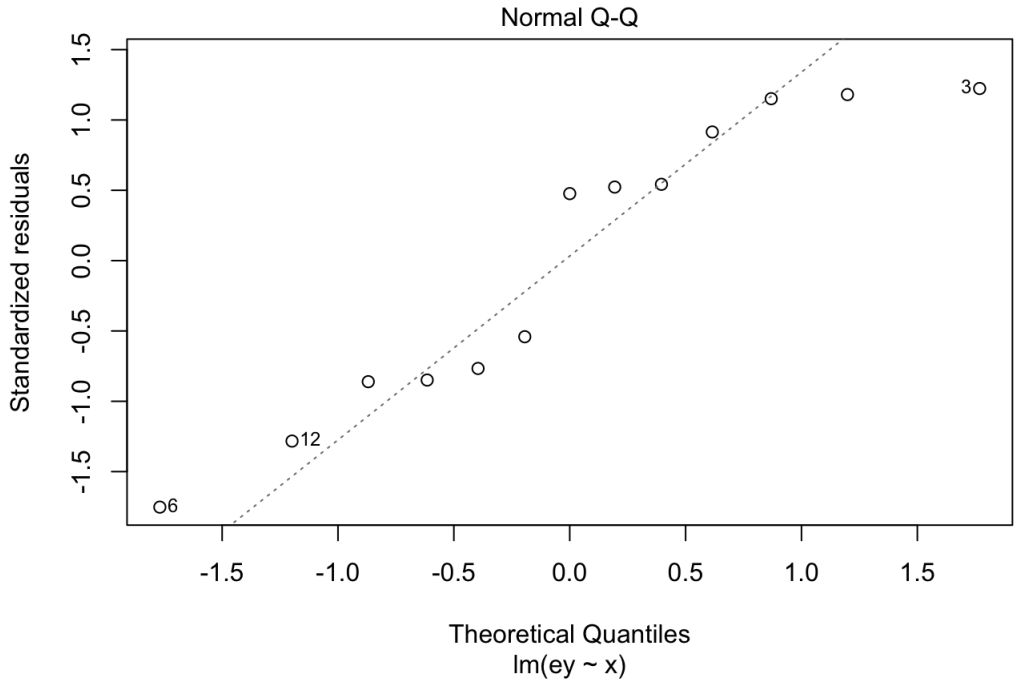


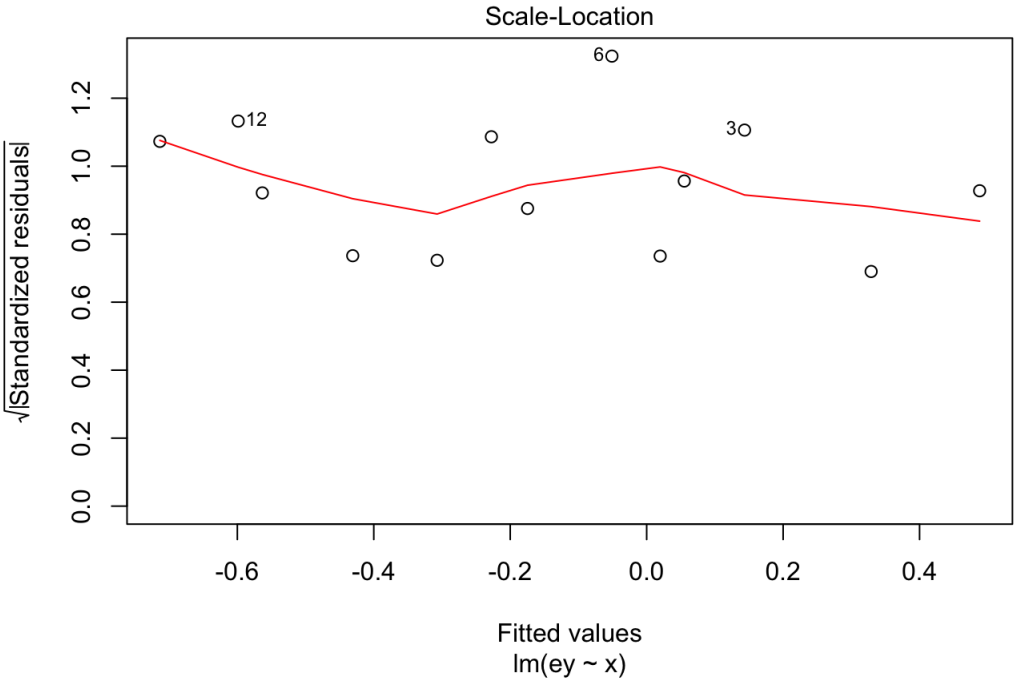

```
ey = log(y)
f3 = lm(ey ~ x)
summary(f3)
```

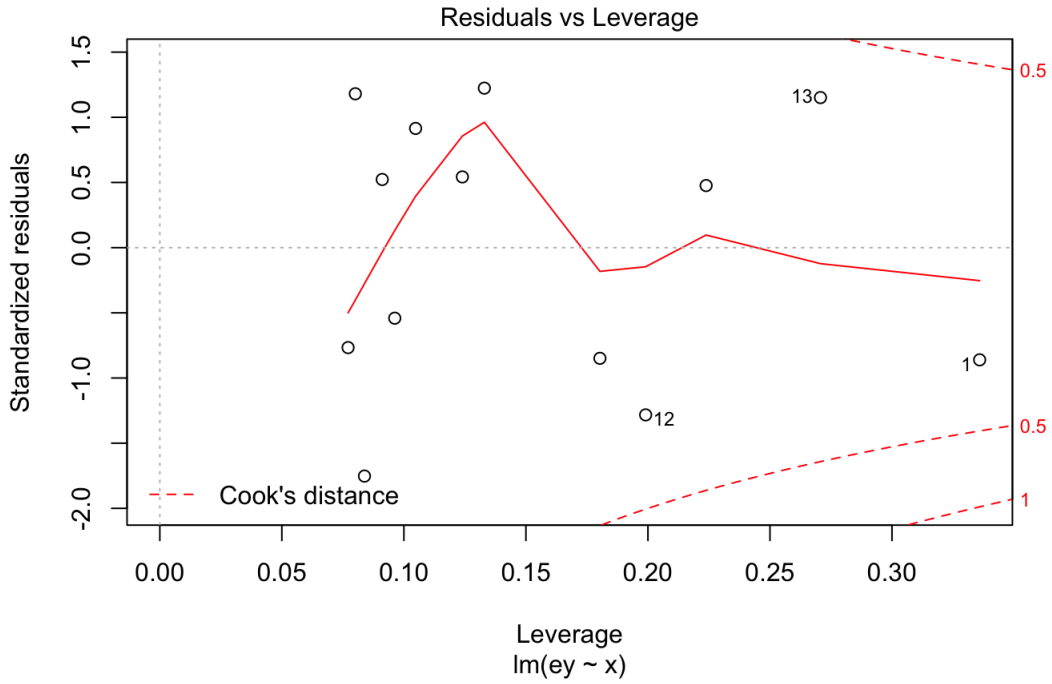
```
##
## Call:
## lm(formula = ey ~ x)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.19753 -0.08673  0.04943  0.10191  0.13418
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.2848294  0.0524410   5.431 0.000207 ***
## x           -0.0088358  0.0008218 -10.751 3.57e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1178 on 11 degrees of freedom
## Multiple R-squared:  0.9131, Adjusted R-squared:  0.9052
## F-statistic: 115.6 on 1 and 11 DF,  p-value: 3.568e-07
```

```
plot(f3)
```

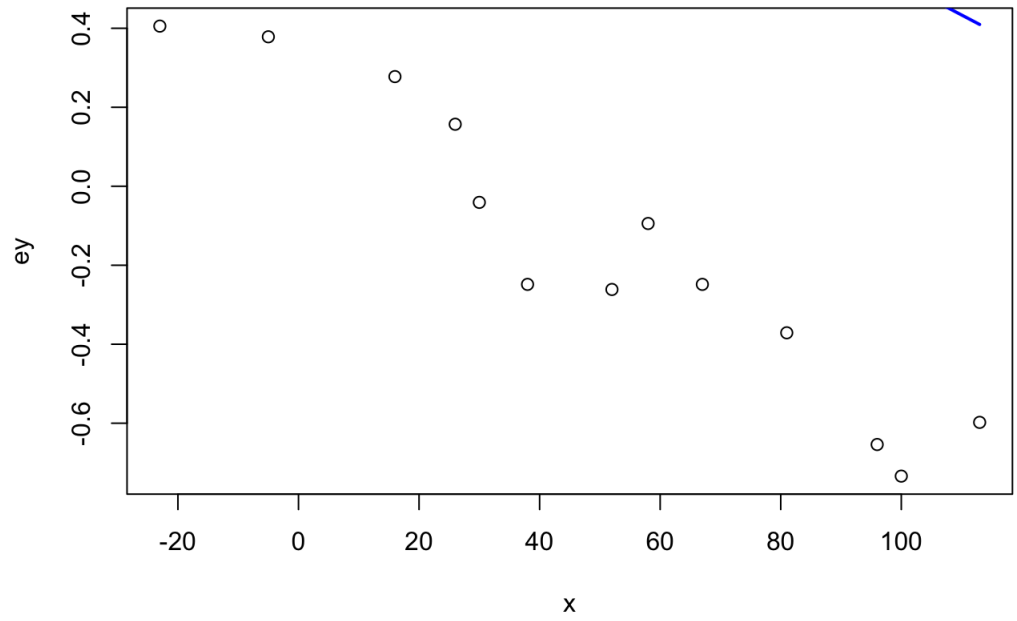



Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(ey ~ x)
3
12
6


Scale-Location
√|Standardized residuals|
Fitted values
lm(ey ~ x)
6
12
3




```
plot(x, ey)
points(x, predict(fit, data.frame(x = x)), type = "l", col = "blue", lwd = "2")
```



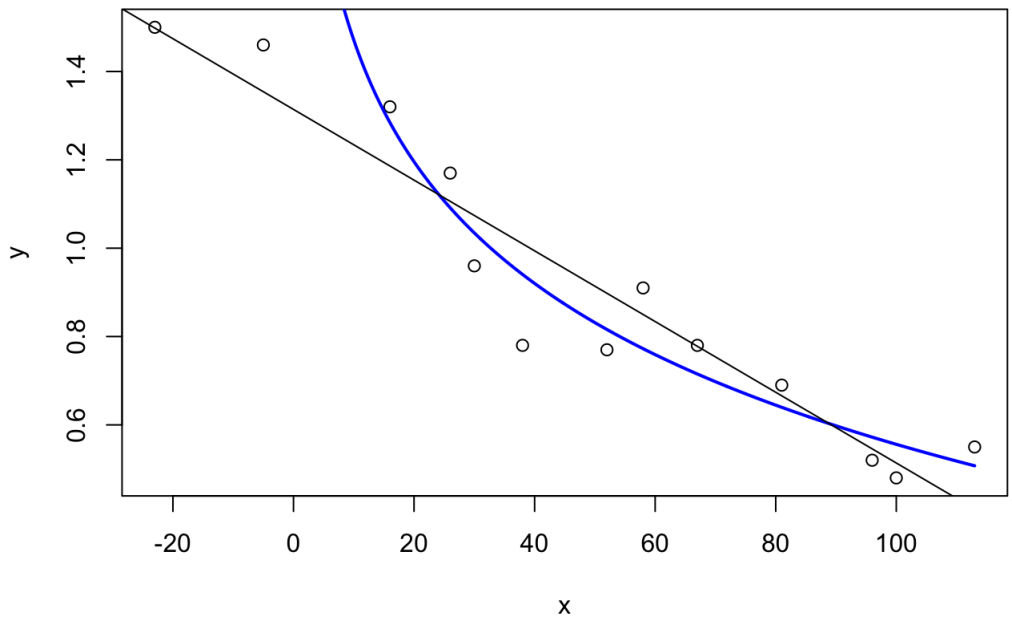

```
lx = log(x)
```

```
## Warning in log(x): 产生了NaNs
```

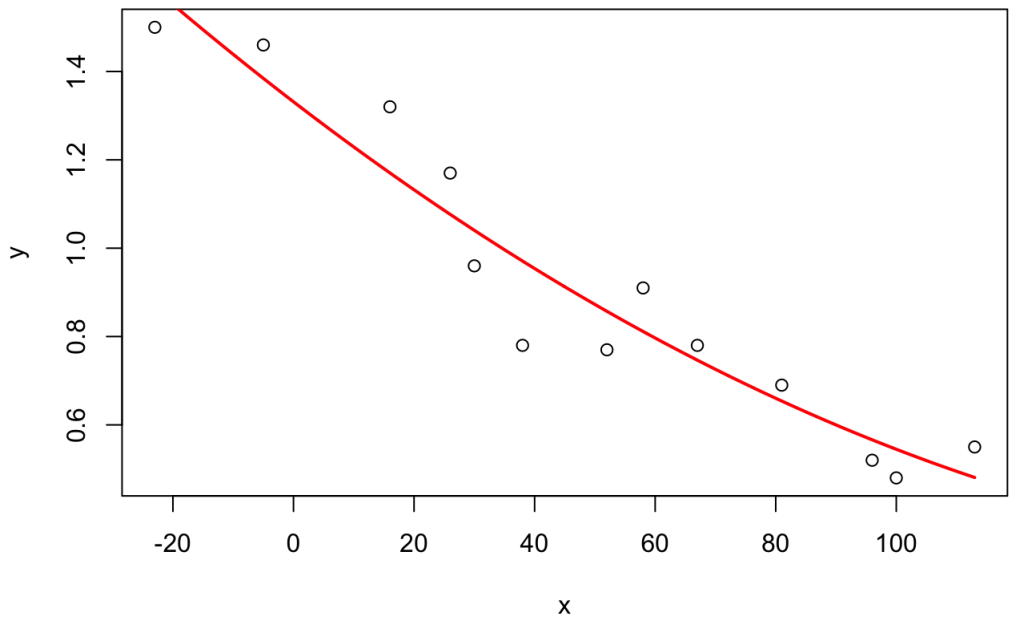
```
f3 = lm(y ~ lx)
plot(x, y)
points(x.2, predict(f3, data.frame(lx = log(x.2))), type = "l", col = "blue",
    lwd = 2)
```

```
## Warning in log(x.2): 产生了NaNs
```

```
abline(fit)
```

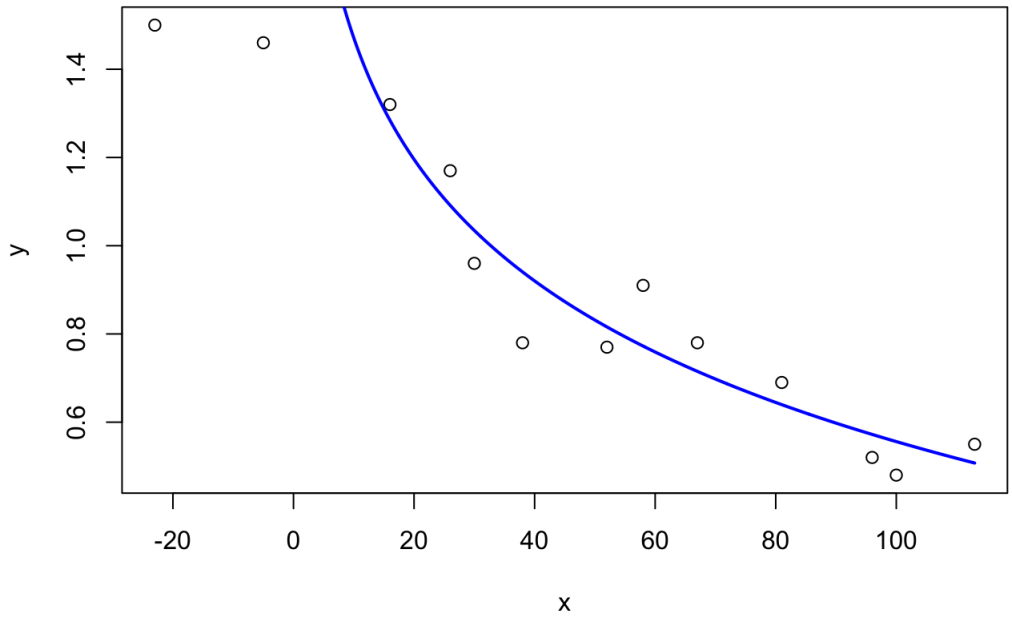



```
plot(x, y)
points(x.2, predict(fit2.0, data.frame(px = x.2^2, x = x.2)), type = "l", col = "red",
    lwd = "2")
```




```
plot(x, y)
points(x.2, predict(f3, data.frame(lx = log(x.2))), type = "l", col = "blue",
    lwd = 2)
```

```
## Warning in log(x.2): 产生了NaNs
```


y
0.6
0.8
1.0
1.2
1.4
-20
0
20
40
60
80
100
x