Getting Started with Regression in R

How much variable x interfer with variable y is possible to predict using regression. Using linear and no-linear regression using polynomias and spines.

load library

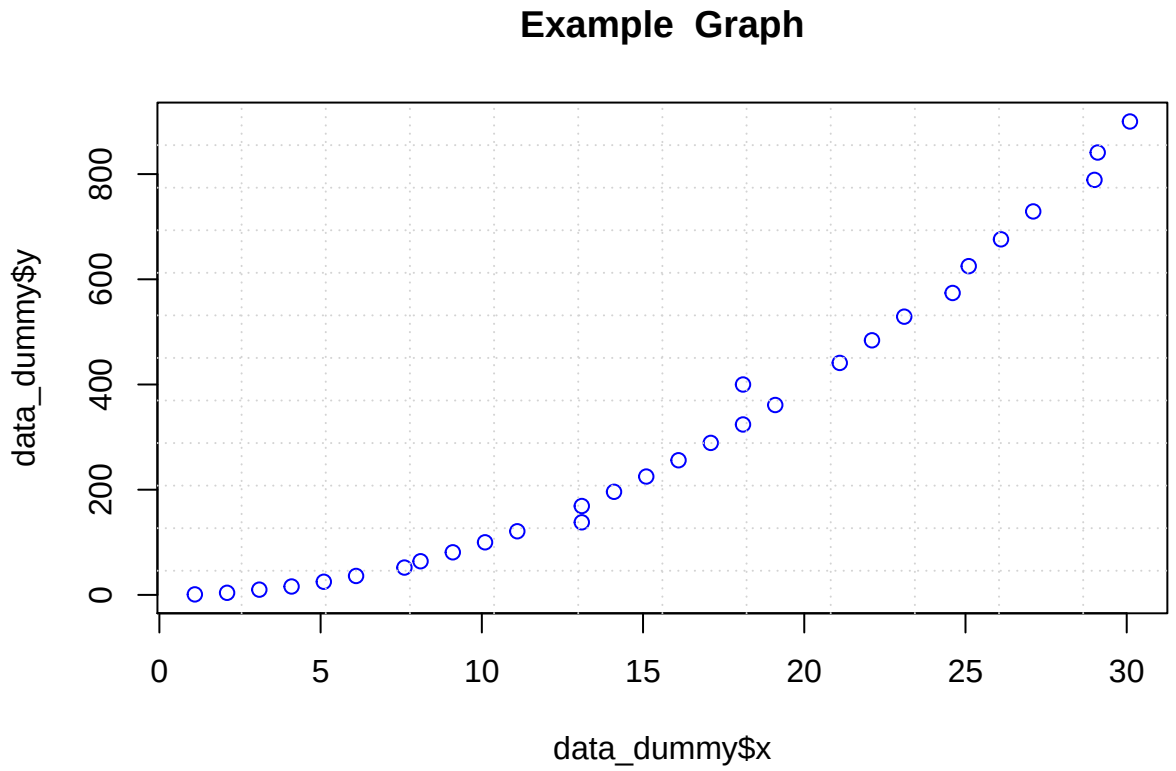
```
library("readxl")

library(splines) # for complex datasets
```

load dataset

```
# dummy dataset to analyse
data_dummy <- read_xlsx("C:/Regression with R/01_example.xlsx")
```

graph analysis

```
plot(data_dummy$x,data_dummy$y, col="blue", main="Example  Graph")
grid(nx= 12, ny=12, col = "lightgray", lty = "dotted", lwd = par("lwd"), equilogs = TRUE)
```



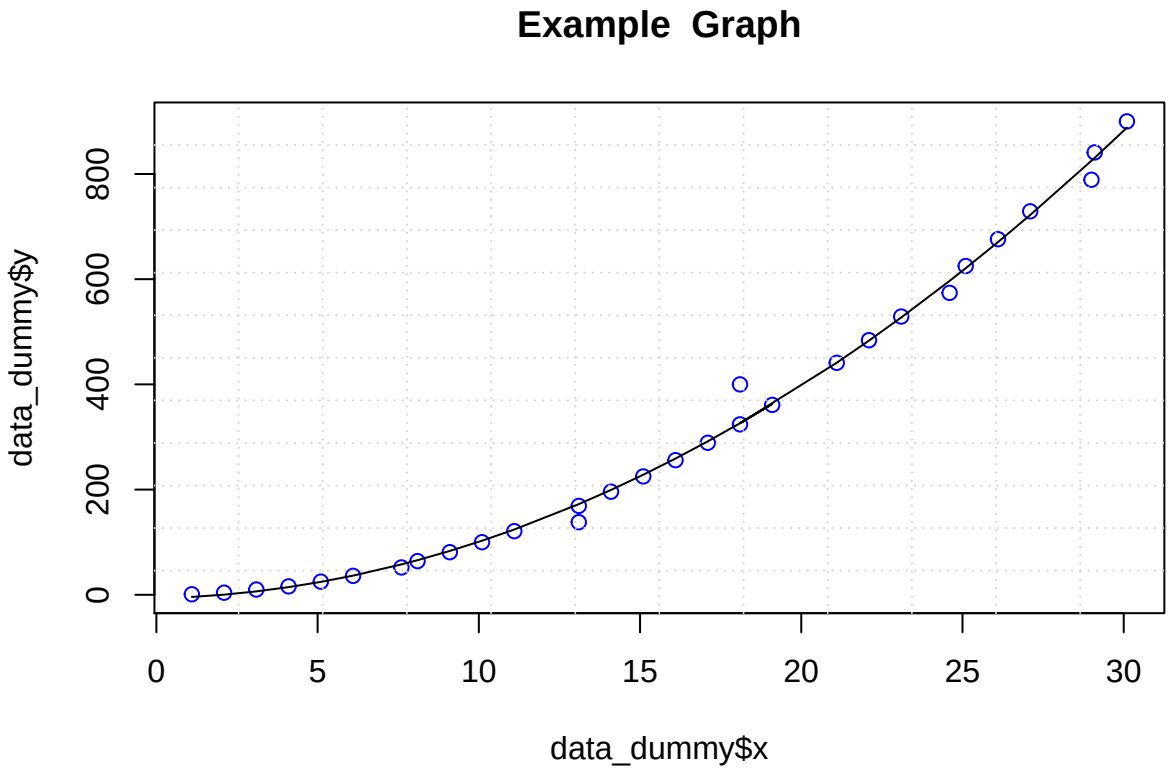
The result seems to follow a non-linear pattern

model guessing that is second degree polynomial

```
my_model <- lm(data_dummy$y ~ poly(data_dummy$x, 2))

# but you can test another alternatives
my_model_linear <- lm(data_dummy$y ~ poly(data_dummy$x, 1))

my_model_degree_20 <- lm(data_dummy$y ~ poly(data_dummy$x, 20))

my_model_spline <- lm(data_dummy$y ~ bs(data_dummy$x)) # Here, bs is the base function. Use the paramet
```

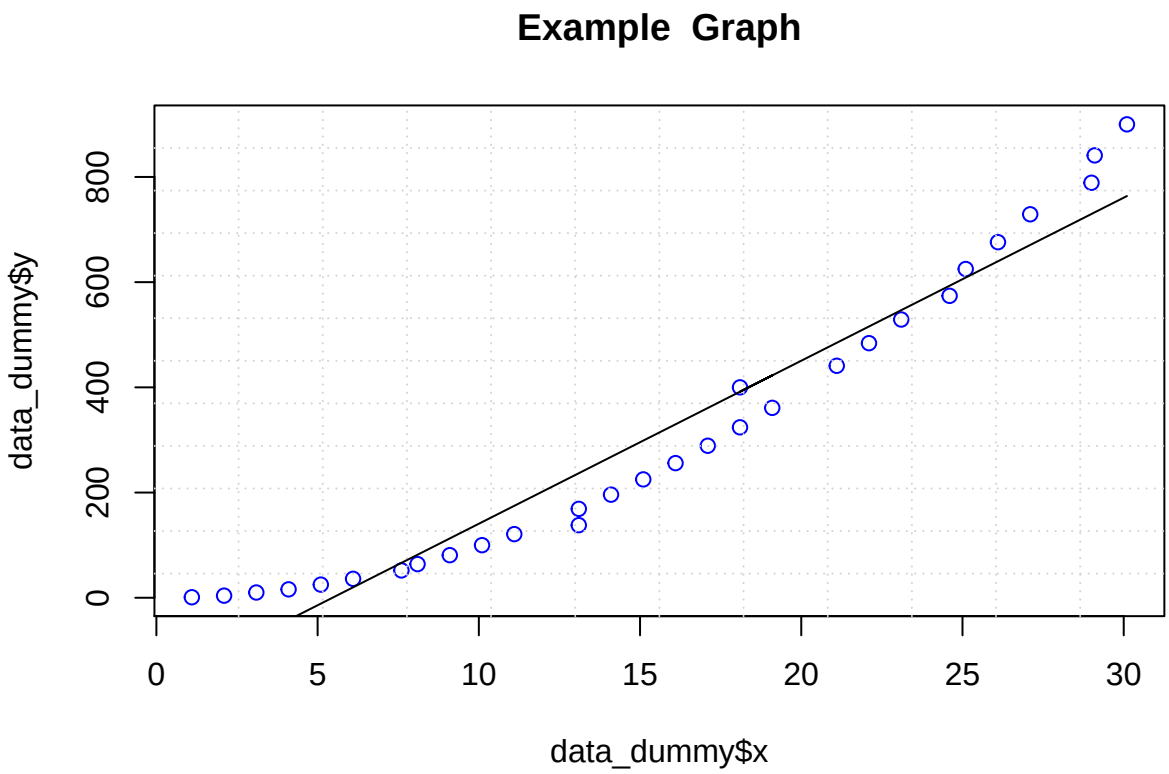
check polynommial result

```
plot(data_dummy$x,data_dummy$y, col="blue", main="Example  Graph")
grid(nx= 12, ny=12, col = "lightgray", lty = "dotted", lwd = par("lwd"), equilogs = TRUE)
lines(data_dummy$x, predict(lm(data_dummy$y ~ poly(data_dummy$x, 2))))
```



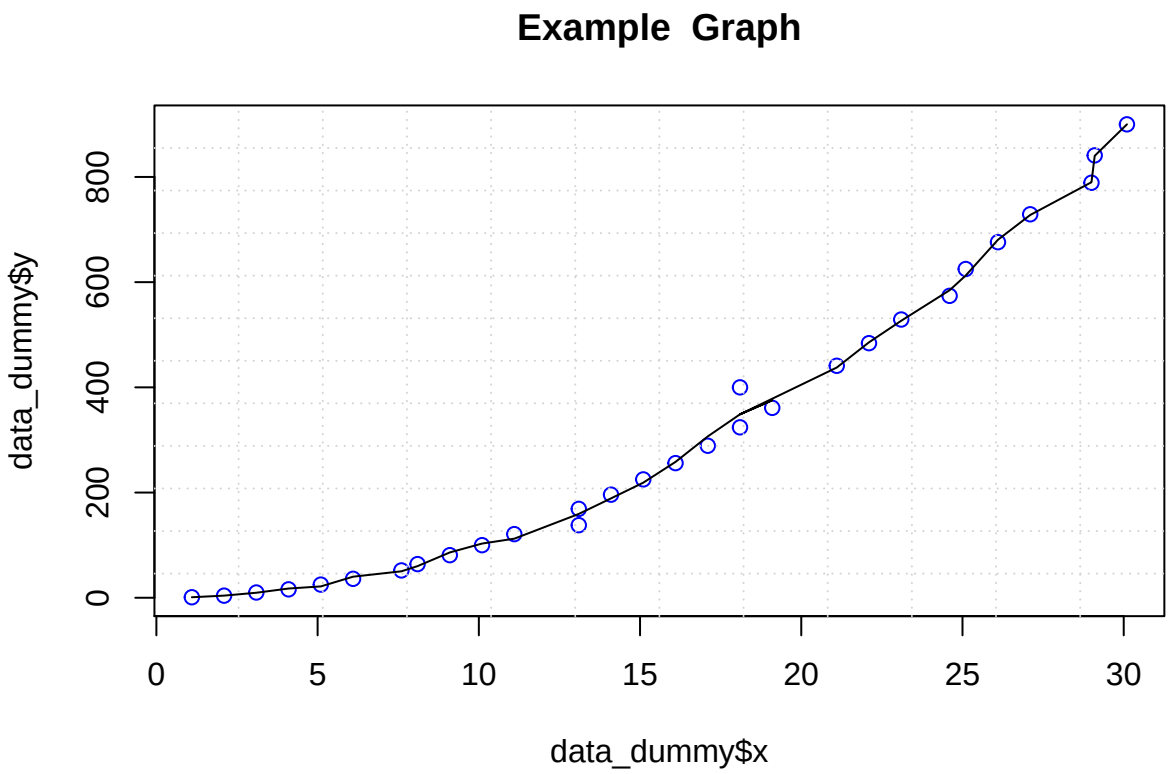
check alternative linear

```
plot(data_dummy$x,data_dummy$y, col="blue", main="Example  Graph")
grid(nx= 12, ny=12, col = "lightgray", lty = "dotted", lwd = par("lwd"), equilogs = TRUE)
lines(data_dummy$x, predict(my_model_linear <- lm(data_dummy$y ~ poly(data_dummy$x, 1))))
```



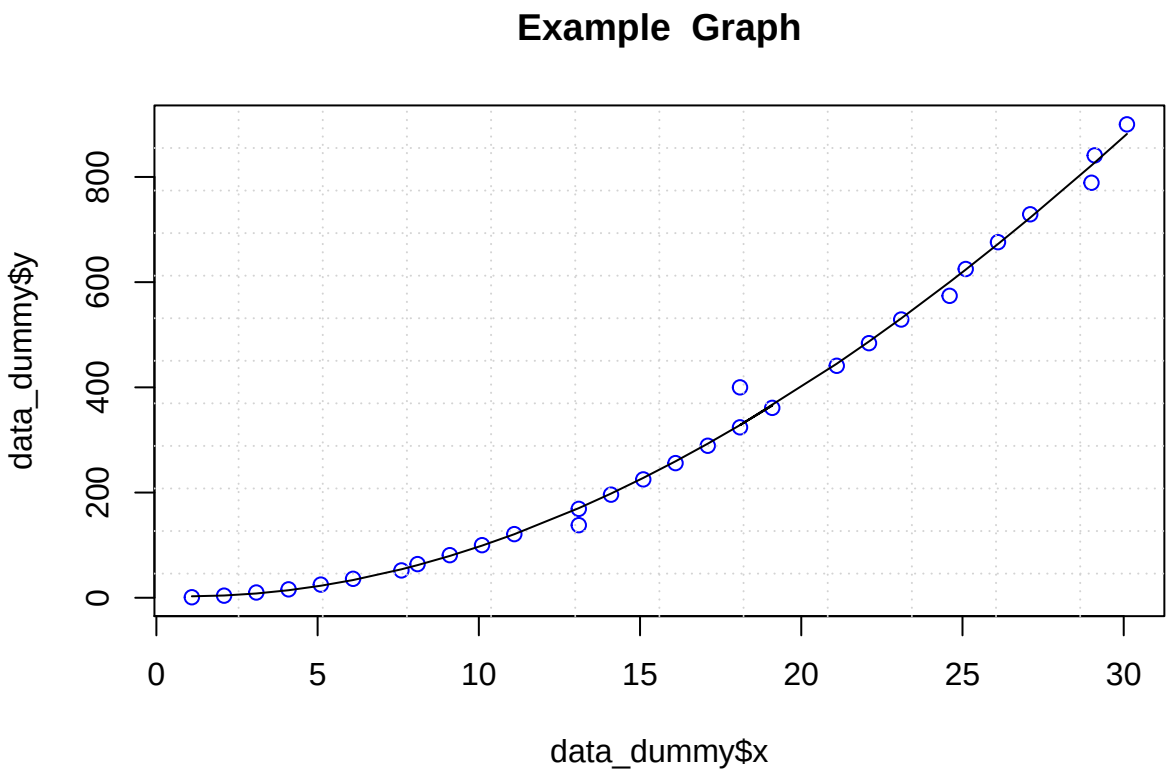
check alternative degree 20

```
plot(data_dummy$x,data_dummy$y, col="blue", main="Example  Graph")
grid(nx= 12, ny=12, col = "lightgray", lty = "dotted", lwd = par("lwd"), equilogs = TRUE)
lines(data_dummy$x, predict(my_model_degree_20 <- lm(data_dummy$y ~ poly(data_dummy$x, 20))))
```



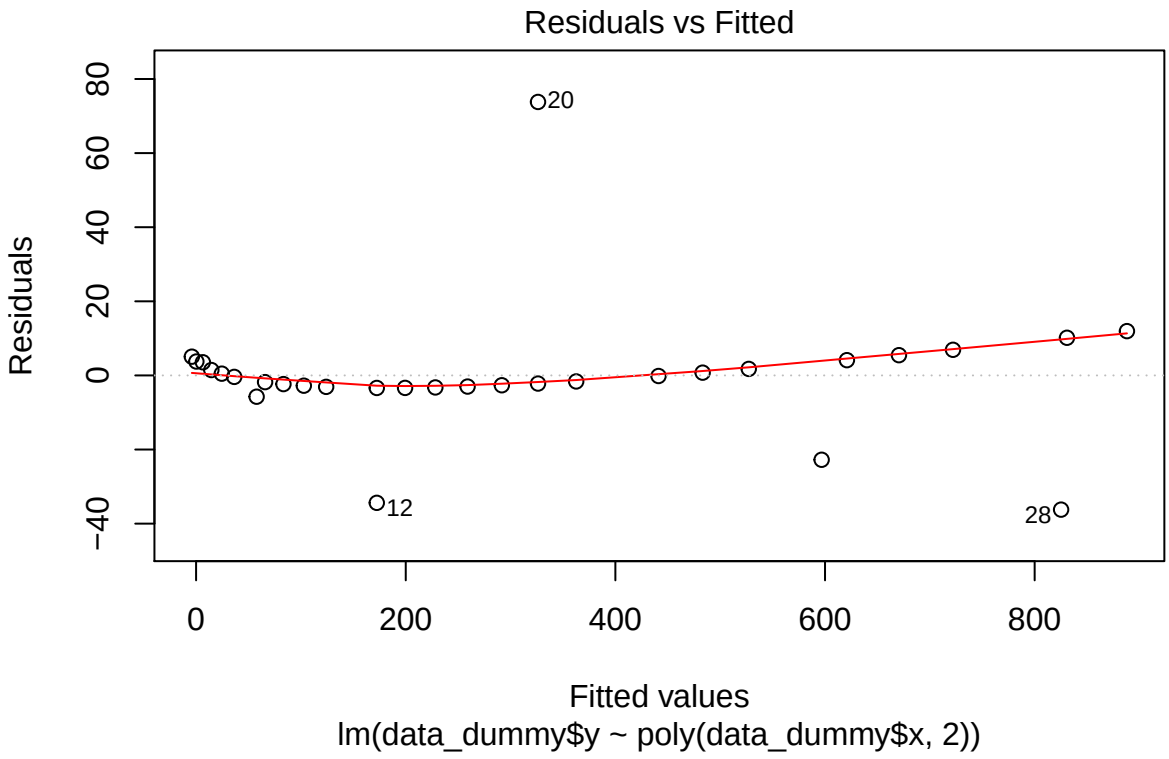
check model spline

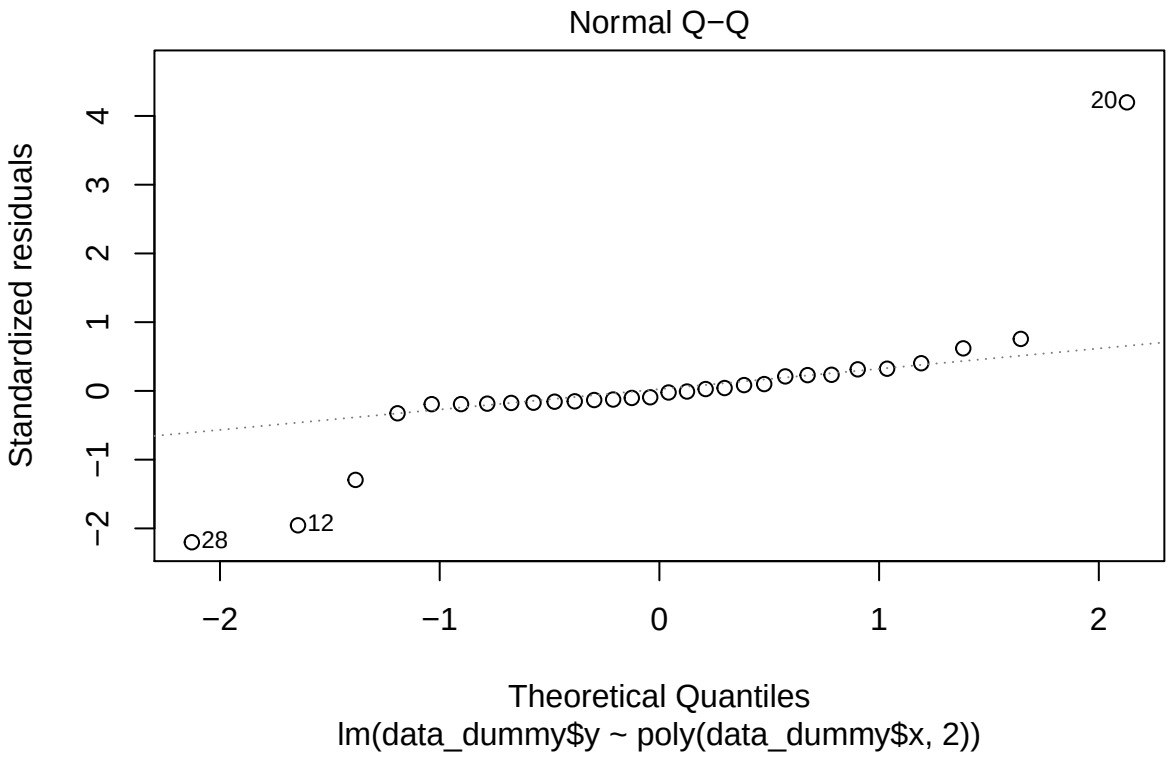
```
plot(data_dummy$x,data_dummy$y, col="blue", main="Example  Graph")
grid(nx= 12, ny=12, col = "lightgray", lty = "dotted", lwd = par("lwd"), equilogs = TRUE)
lines(data_dummy$x, predict(my_model_spline <- lm(data_dummy$y ~ bs(data_dummy$x))))
```

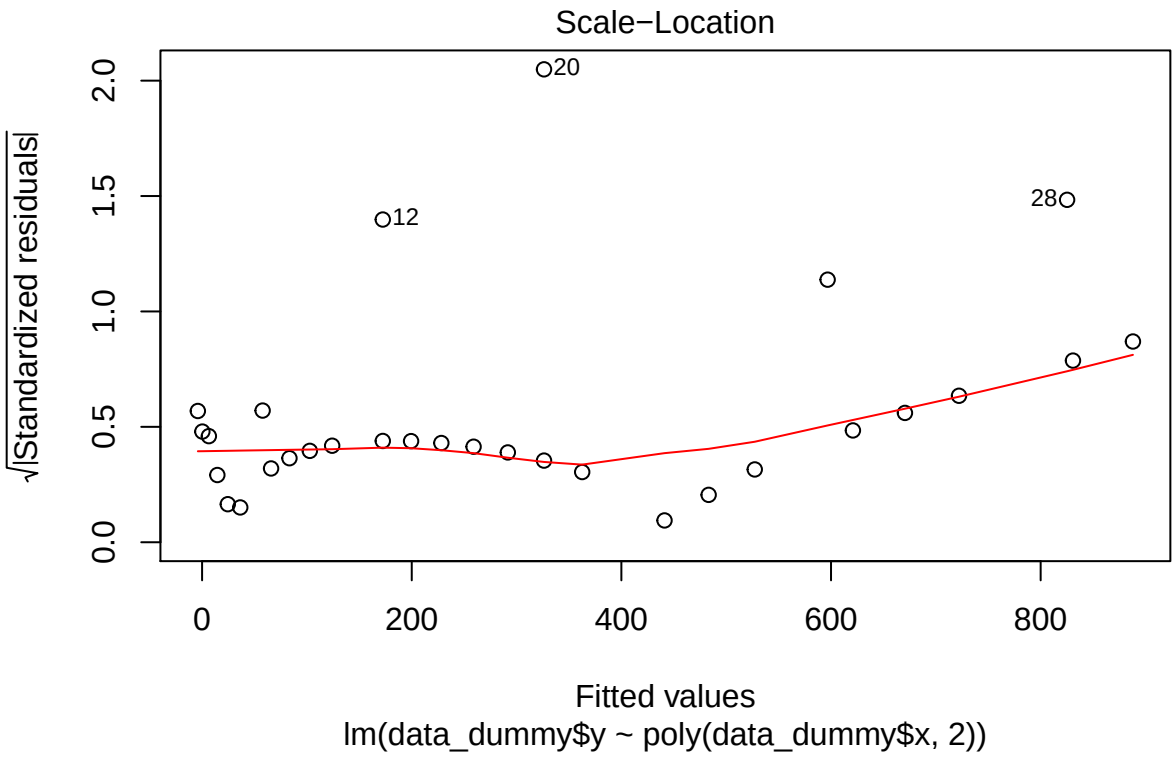


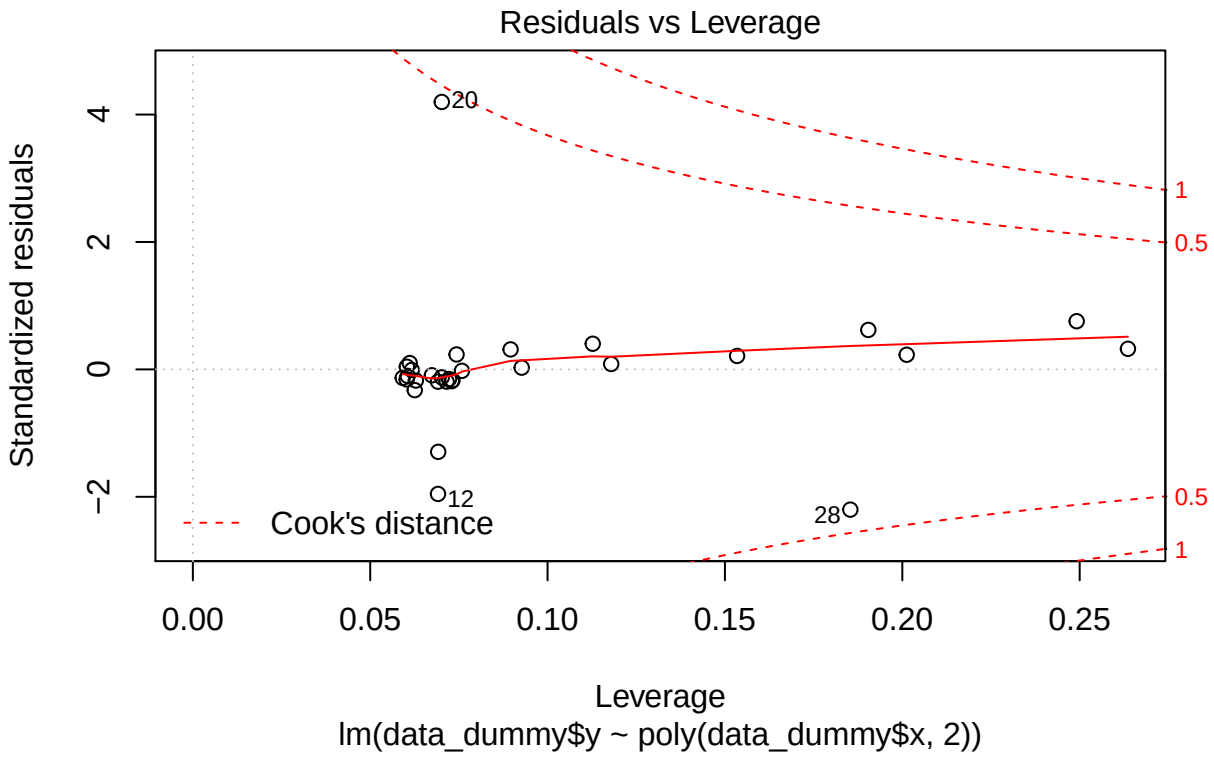
plot models

```
plot(my_model)
```

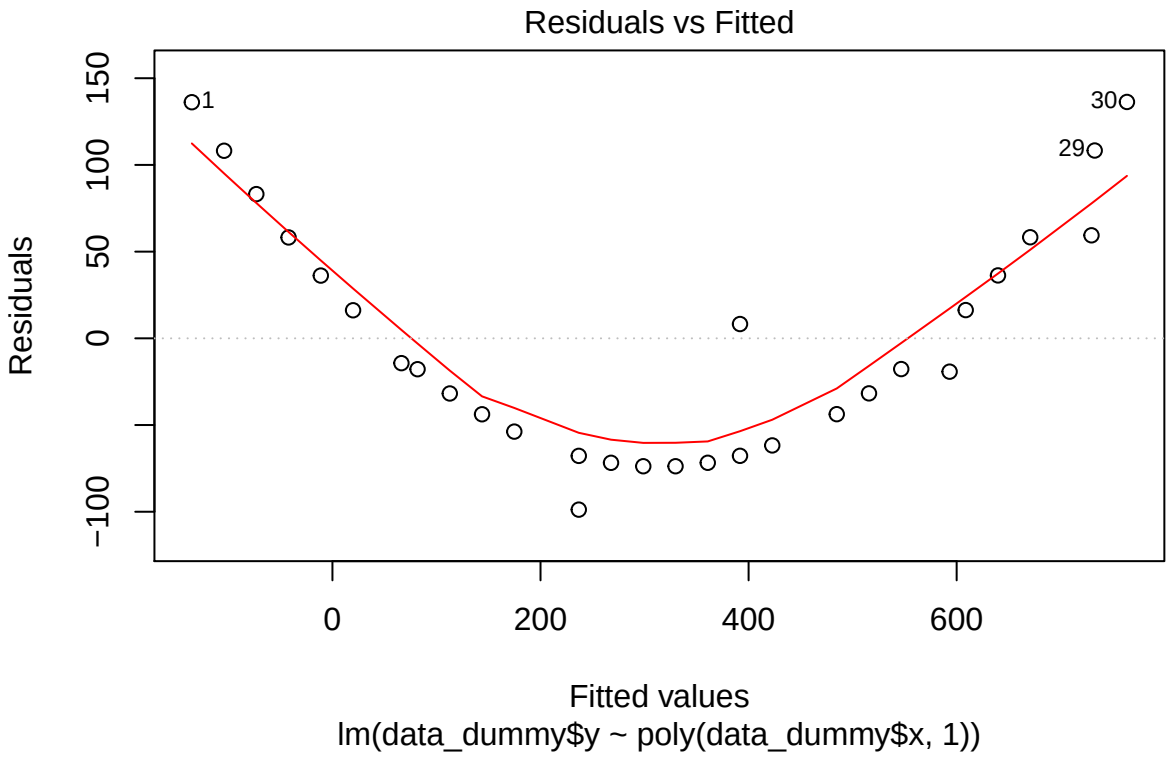

Residuals vs Fitted
Residuals
80
60
40
20
0
−40
20
12
28
0
200
400
600
800
Fitted values
lm(data_dummy$y ~ poly(data_dummy$x, 2))

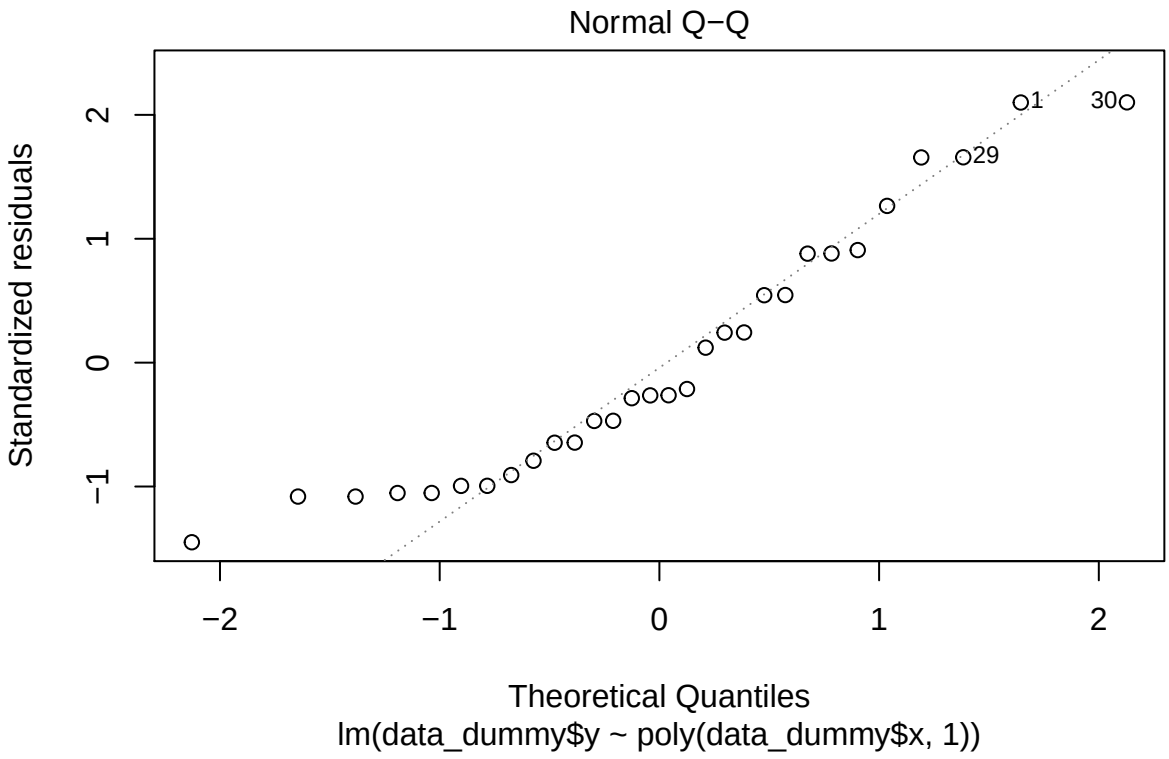

Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(data_dummy$y ~ poly(data_dummy$x, 2))
20
12
28
−2
−1
0
1
2
3
4

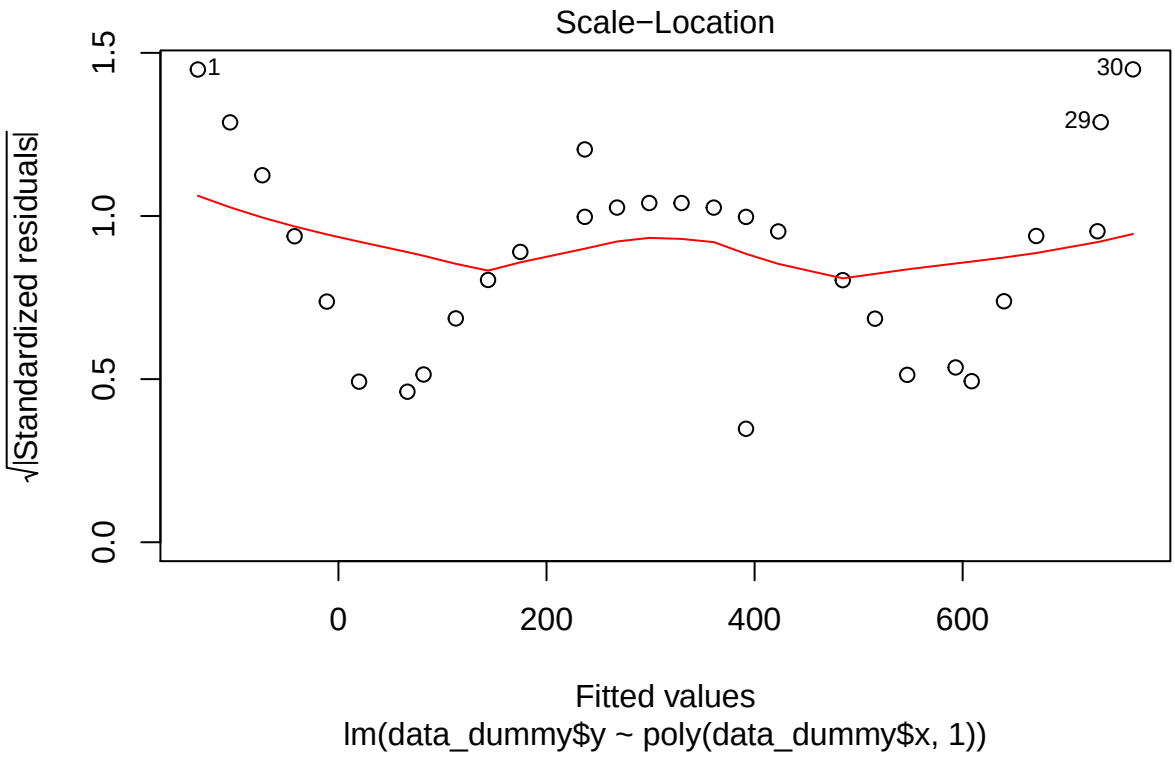

Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
0
200
400
600
800
20
12
28
Fitted values
lm(data_dummy$y ~ poly(data_dummy$x, 2))

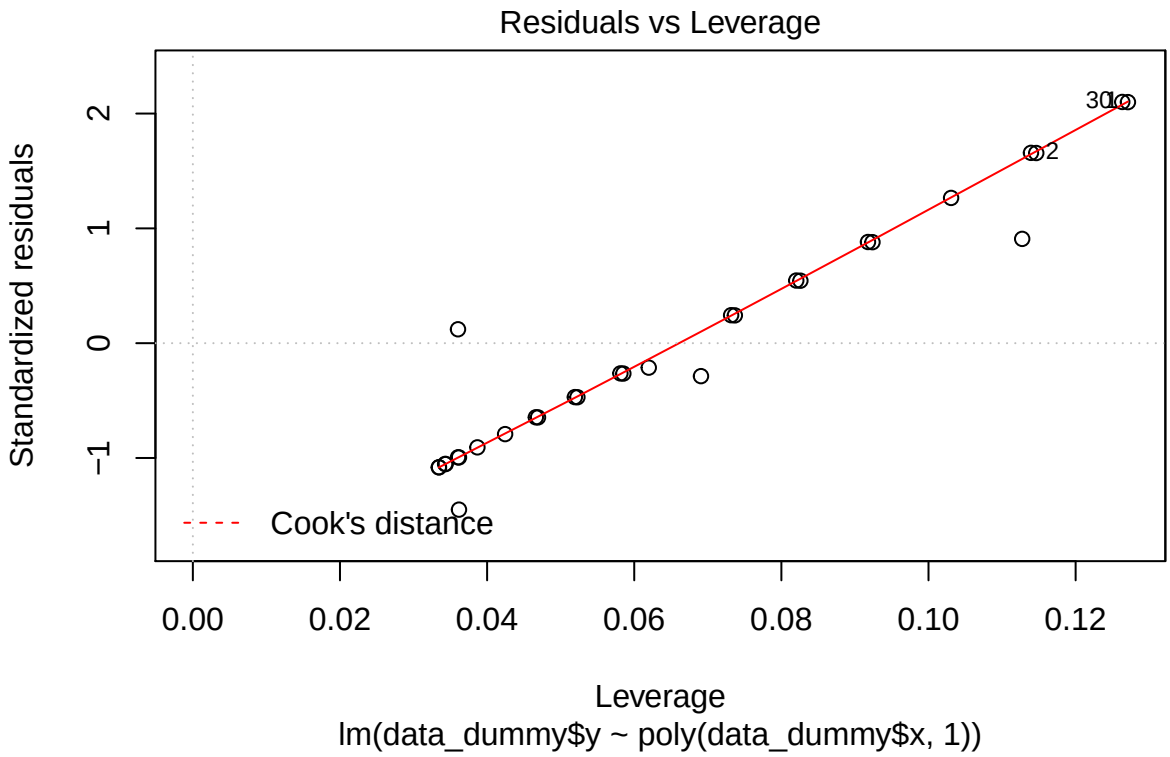



```
plot(my_model_linear)
```

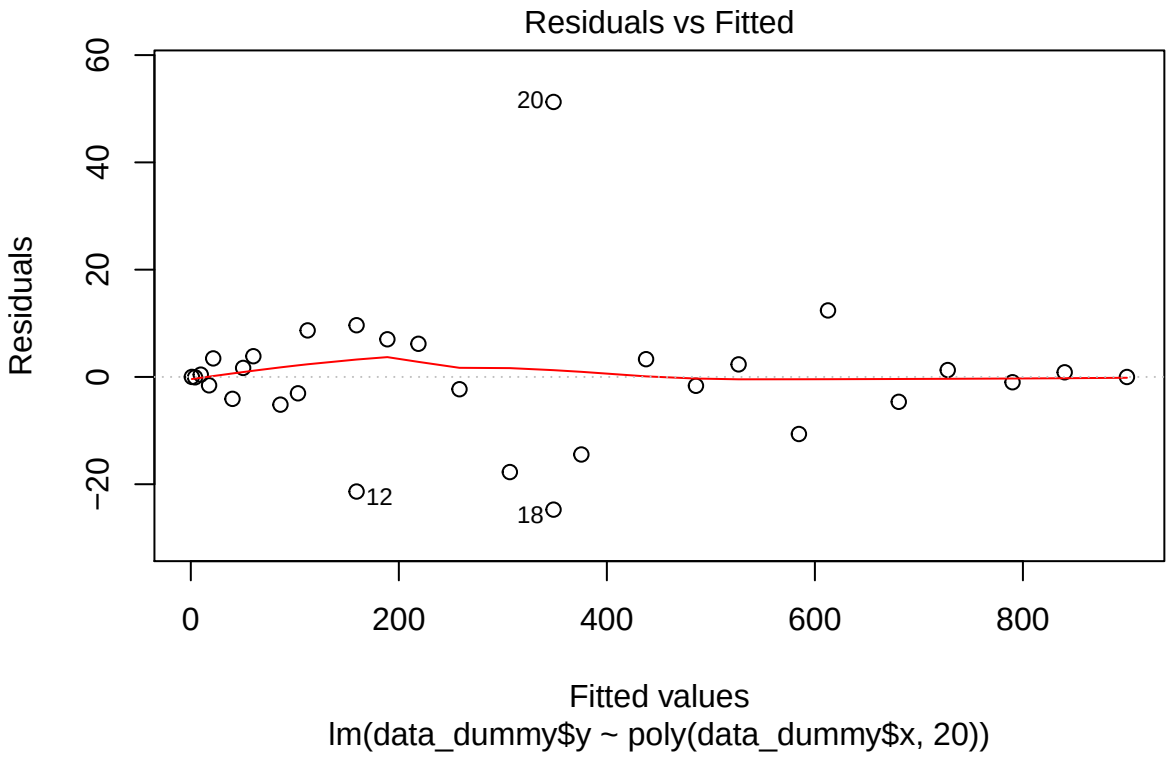

Residuals vs Fitted
Residuals
150
100
50
0
−100
1
30
29
0
200
400
600
Fitted values
lm(data_dummy$y ~ poly(data_dummy$x, 1))

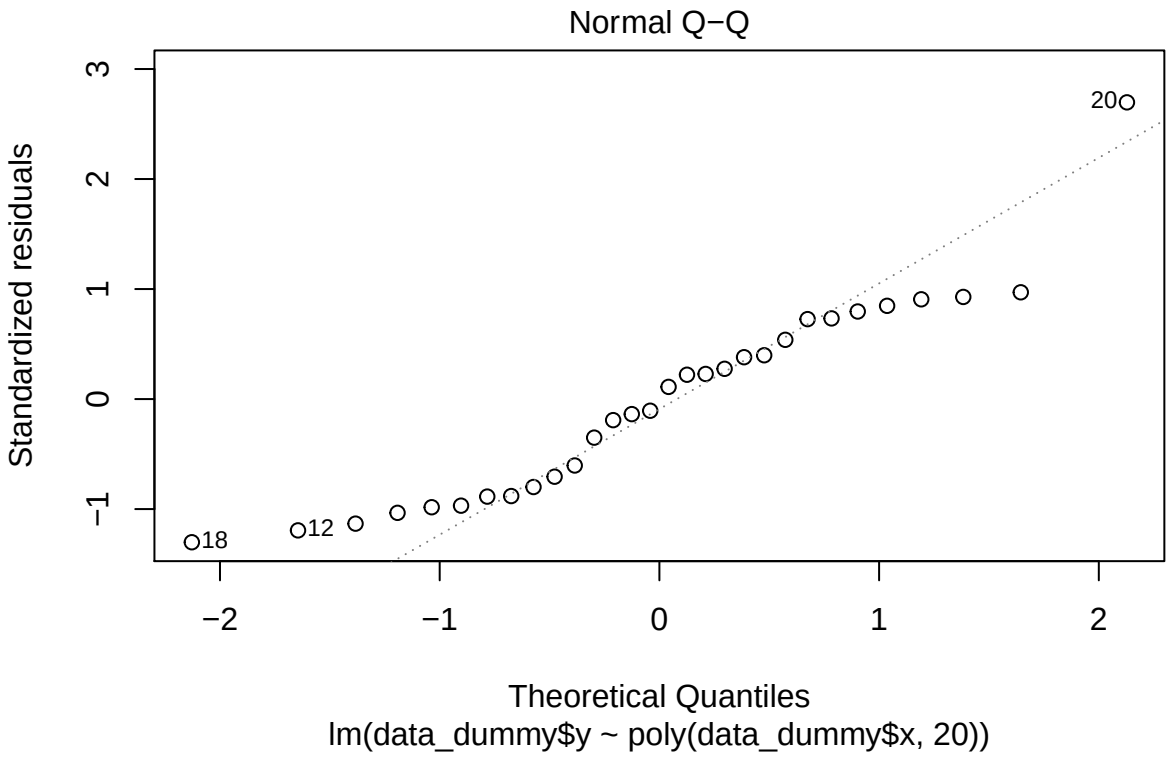

Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(data_dummy$y ~ poly(data_dummy$x, 1))
1
30
29
−2
−1
0
1
2

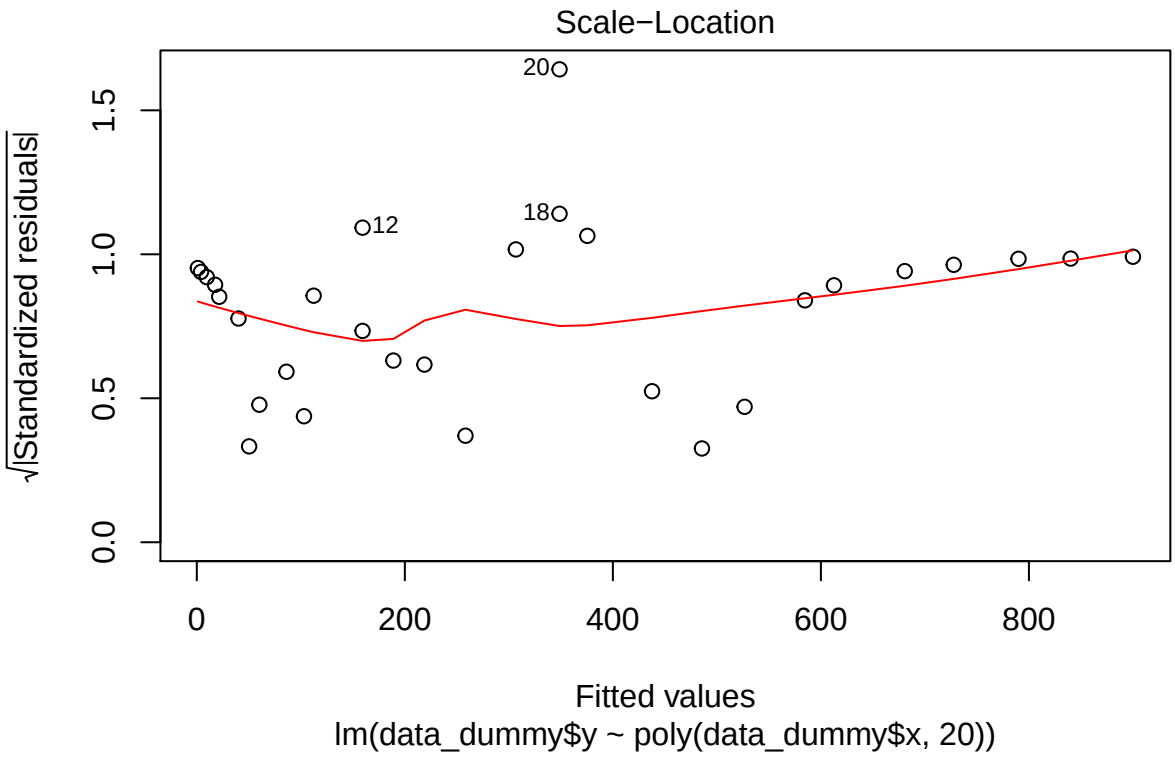

Scale−Location
√|Standardized residuals|
1.5
1.0
0.5
0.0
1
30
29
0
200
400
600
Fitted values
lm(data_dummy$y ~ poly(data_dummy$x, 1))



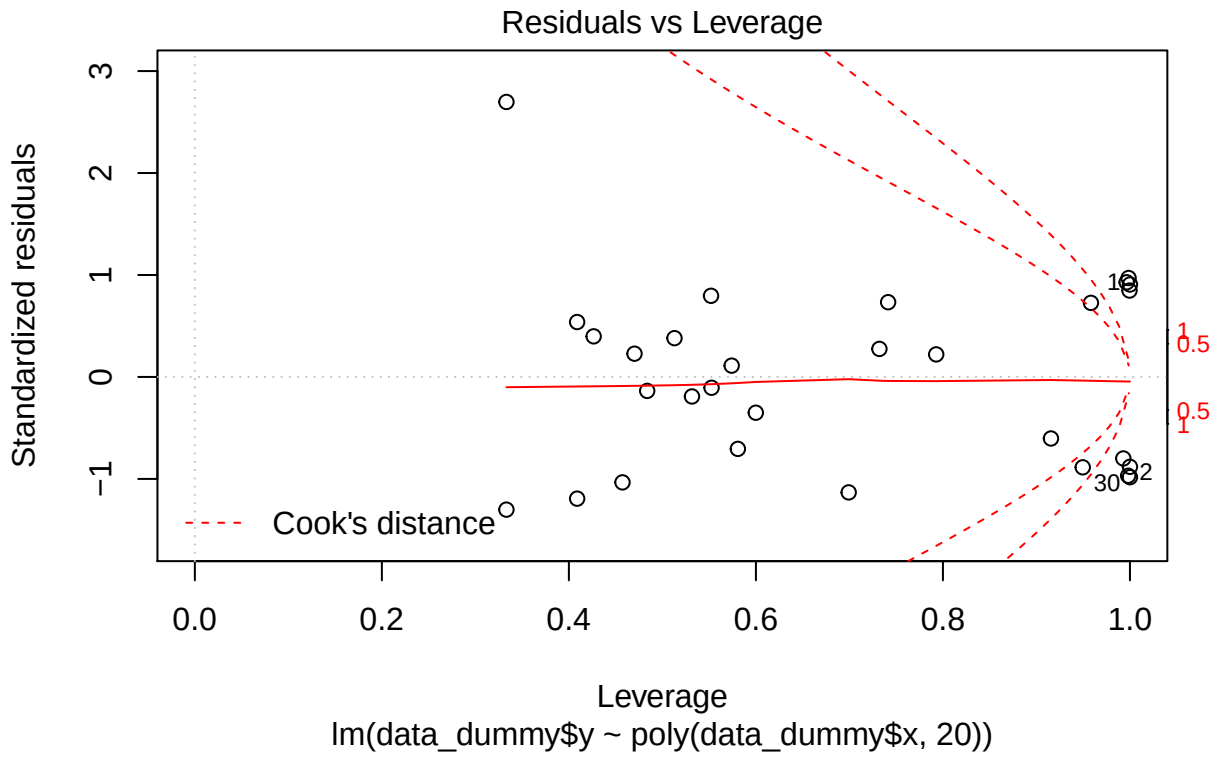

```
plot(my_model_degree_20)
```


Residuals vs Fitted
Residuals
60
40
20
0
−20
20
12
18
0
200
400
600
800
Fitted values
lm(data_dummy$y ~ poly(data_dummy$x, 20))

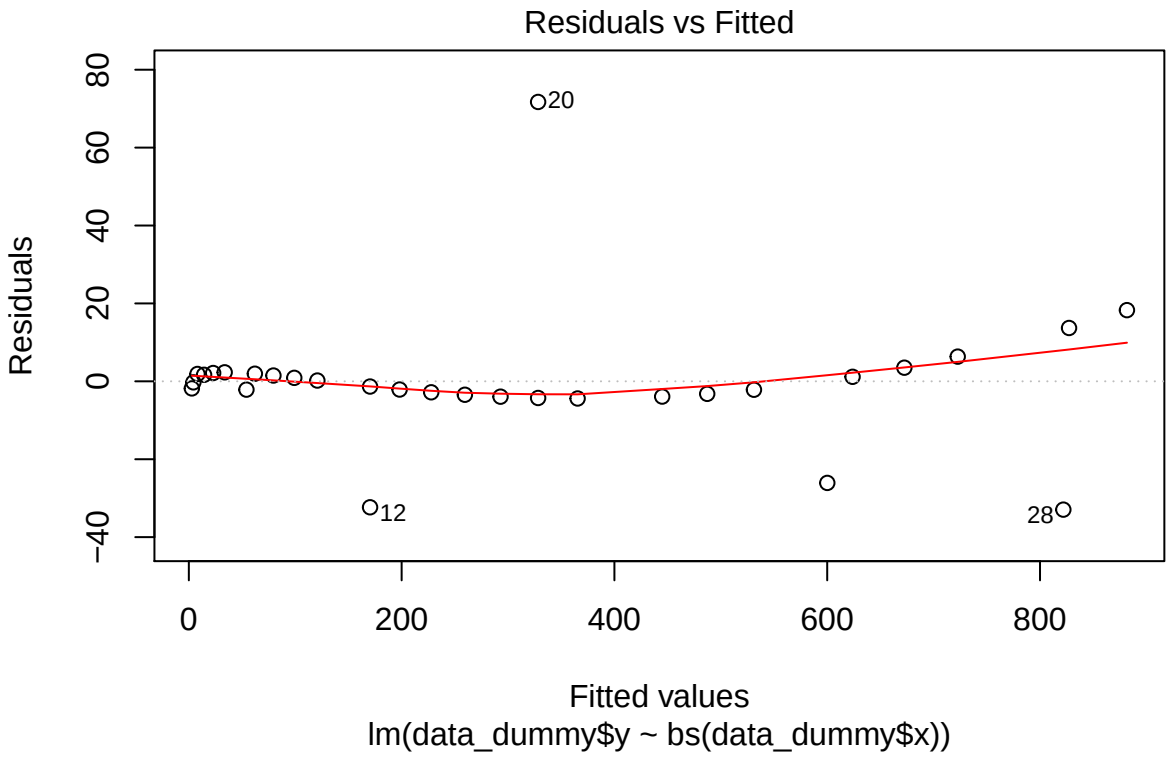

Normal Q−Q
Standardized residuals
3
2
1
0
−1
20
18
12
−2
−1
0
1
2
Theoretical Quantiles
lm(data_dummy$y ~ poly(data_dummy$x, 20))

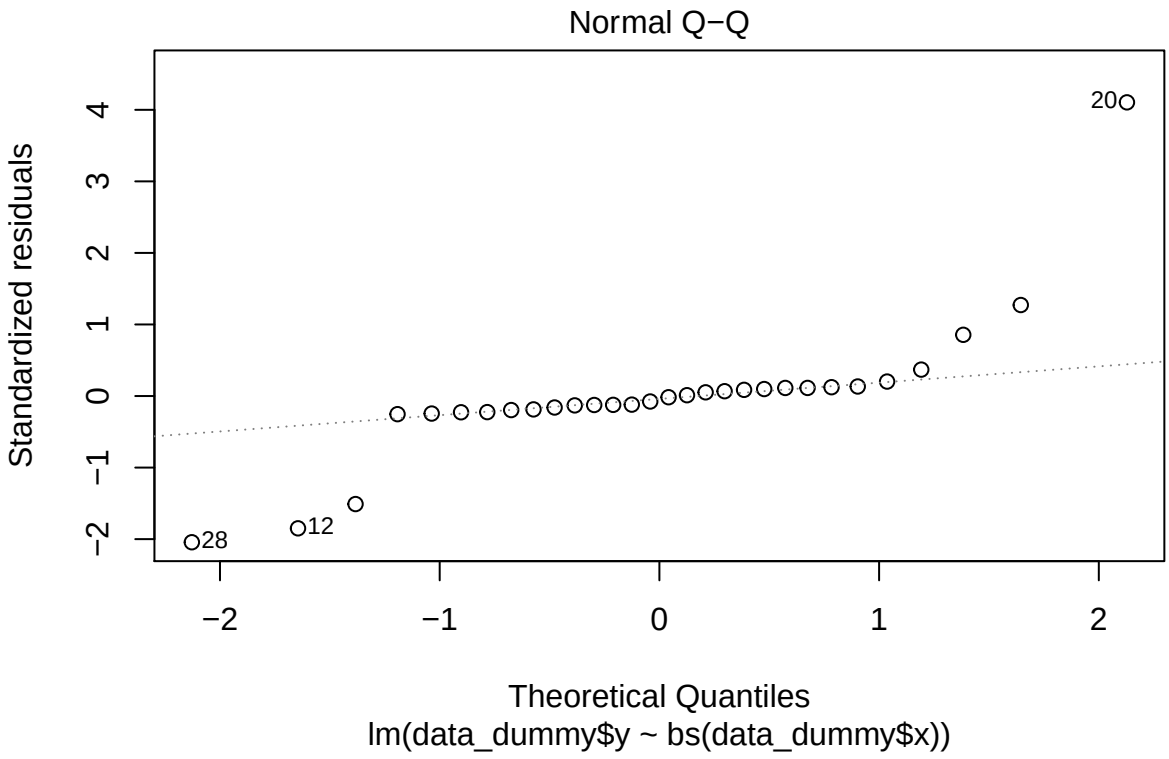


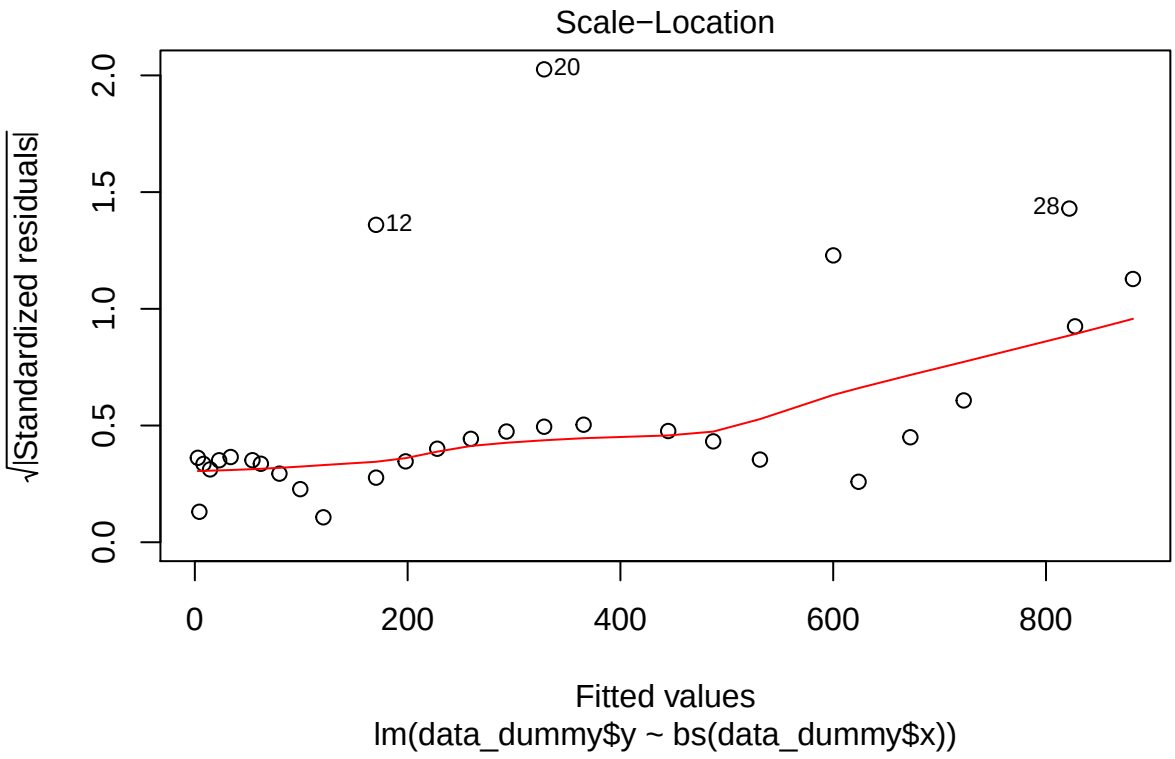

```
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produzidos

## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produzidos
```

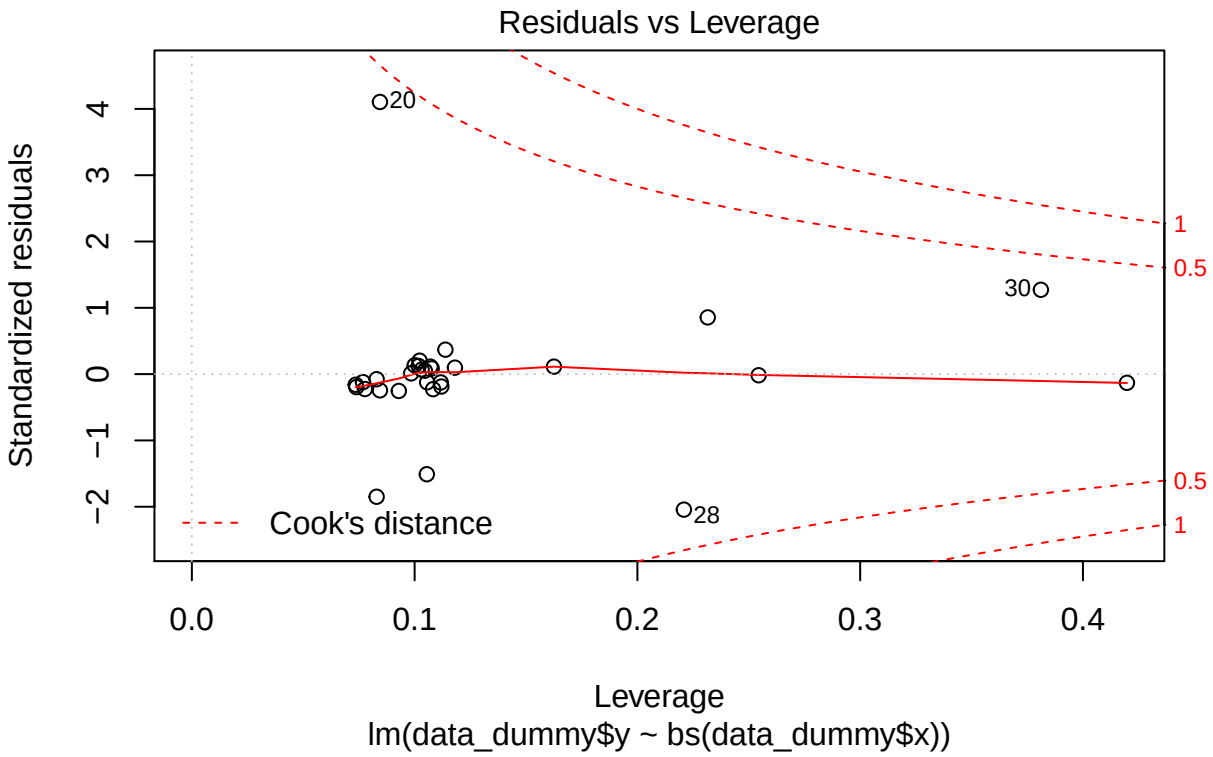



```
plot(my_model_spline)
```


Residuals vs Fitted
Residuals
80
60
40
20
0
−40
20
12
28
0
200
400
600
800
Fitted values
lm(data_dummy$y ~ bs(data_dummy$x))


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(data_dummy$y ~ bs(data_dummy$x))
20
12
28


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
20
12
28
0
200
400
600
800
Fitted values
lm(data_dummy$y ~ bs(data_dummy$x))



t test –This test is going to compare their means, assuming they both are under a normal distribution.

```
t.test(data_dummy$y, predict(my_model))
```

```
##
##  Welch Two Sample t-test
##
## data:  data_dummy$y and predict(my_model)
## t = 7.829e-16, df = 58, p-value = 1
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -145.3372  145.3372
## sample estimates:
## mean of x mean of y
##     315.2     315.2
```

```
t.test(data_dummy$y, predict(my_model_linear))
```

```
##
##  Welch Two Sample t-test
##
## data:  data_dummy$y and predict(my_model_linear)
## t = 7.9387e-16, df = 57.947, p-value = 1
## alternative hypothesis: true difference in means is not equal to 0
```

```
## 95 percent confidence interval:
##  -143.3311  143.3311
## sample estimates:
## mean of x mean of y
##     315.2     315.2
```

```
t.test(data_dummy$y, predict(my_model_degree_20))
```

```
##
##  Welch Two Sample t-test
##
## data:  data_dummy$y and predict(my_model_degree_20)
## t = 7.8255e-16, df = 58, p-value = 1
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -145.4021  145.4021
## sample estimates:
## mean of x mean of y
##     315.2     315.2
```

```
t.test(data_dummy$y, predict(my_model_spline))
```

```
##
##  Welch Two Sample t-test
##
## data:  data_dummy$y and predict(my_model_spline)
## t = 0, df = 58, p-value = 1
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -145.342  145.342
## sample estimates:
## mean of x mean of y
##     315.2     315.2
```

Fonte: https://www.datasciencecentral.com/profiles/blogs/getting-started-with-regression-in-r