DATA 609 Assignment 3: Model Fitting and Experiment Modeling

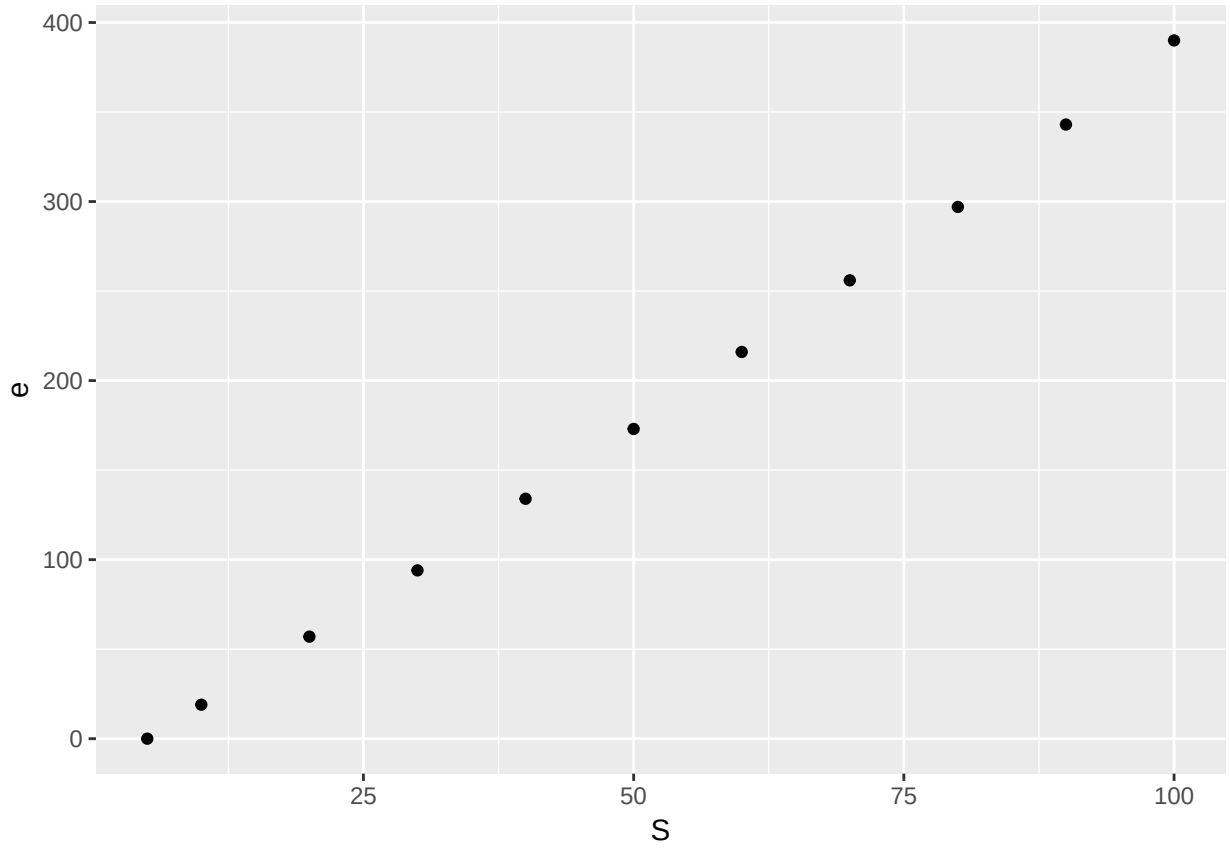
Dan Smilowitz

February 15, 2017

Chapter 3: Model Fitting

Section 3.1, Problem 2

```
S <- c(5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
e <- c(0, 19, 57, 94, 134, 173, 216, 256, 297, 343, 390)
```




```
c1 <- (e[length(e)] - e[1]) / (S[length(S)] - S[1])
```

For the model $e = c_1 S$, the estimated value of the constant is $c_1 = 4.11$.

Section 3.2, Problem 2a

The residuals r_i for a line $y = ax + b$ are given by $r_i = y_i - \hat{y}_i = y_i - ax_i - b$. The goal is to minimize the largest residual $r = max(r_i)$ subject to $r - r_i \geq 0$ and $r + r_i \geq 0$. For the six data points given, this equates to the conditions

$$\begin{aligned} r - y_i + ax_i + b &\geq 0 \\ r + y_i - ax_i - b &\geq 0 \end{aligned} \qquad \text{for } i = 1 \ldots 6$$

This can be rearranged to give

$$\begin{aligned} r + ax_i + b &\geq y_i \\ r - ax_i - b &\geq -y_i \end{aligned} \qquad \text{for } i = 1 \ldots 6$$

This system can be represented through matrix multiplication:

$$\begin{bmatrix} 1 & x_1 & 1 \\ 1 & -x_1 & -1 \\ \vdots & \vdots & \vdots \\ 1 & x_6 & 1 \\ 1 & -x_6 & 1 \end{bmatrix} \begin{bmatrix} r \\ a \\ b \end{bmatrix} \geq \begin{bmatrix} y_1 \\ -y_1 \\ \vdots \\ y_6 \\ y_6 \end{bmatrix}$$

This system of inequalities can be evaluated in R using linear programming

```
x <- c(1.0, 2.3, 3.7, 4.2, 6.1, 7.0)
y <- c(3.6, 3.0, 3.2, 5.1, 5.3, 6.8)
# set up matrix of coefficients for r, a, and b
A <- matrix(c(
  rep(1, 12), # each inequality has r coefficient of 1
  c(rbind(-x, x)), # alternating a coefficients of x & -x
  rep(c(-1, 1), 6)), # alternating b coefficients of 1 & -1
  nrow = 12)
# vector of constants -- alternating y & -y
b <- c(rbind(-y, y))
# solve system
library(lpSolve)
min_r <- lp("min", c(1, 0, 0), A, rep(">=", nrow(A)), b)$solution
```

The mathematical model that minimizes the largest deviation is given by

$$y = 0.5333x + 2.1467$$

The largest deviation under this model is $r = 0.92$, as shown below:

x	y	$\hat{y}$	$\lvert r \rvert$
1	3.6	2.68	0.92
2.3	3	3.373	0.3733
3.7	3.2	4.12	0.92
4.2	5.1	4.387	0.7133
6.1	5.3	5.4	0.1
7	6.8	5.88	0.92

Section 3.3, Problem 10

For a power curve of the form $y = ax^n$, the equation for a is given by

$$a = \frac{\sum x_i^n y_i}{\sum x_i^{2n}}$$

Using the provided planet data and $n = 3/2$, the model $y = ax^{3/2}$ can be solved — this is Kepler's third law of planetary motion, where y is the period of orbit and x is the distance from the sun.

```
body <- c('Mercury', 'Venus', 'Earth', 'Mars', 'Jupiter', 'Saturn', 'Uranus', 'Neptune')
period <- c(7.60e6, 1.94e7, 3.16e7, 5.94e7, 3.74e8, 9.35e8, 2.64e9, 5.22e9)
distance <- c(5.79e10, 1.08e11, 1.5e11, 2.28e11, 7.79e11, 1.43e12, 2.87e12, 4.5e12)

n <- 3/2

a = sum(distance^n * period) / sum(distance^(2 * n))
```

This gives a value of $a \approx 5.4605 \times 10^{-10}$, so the model is given by

$$period = 5.4605 \times 10^{-10} \times distance^{3/2}$$

Section 3.4, Problem 7

Part a

For the model $W = k_1 l^3$, the least squares estimate of k_1 is given by

$$k_1 = \frac{\sum l_i^3 W_i}{\sum l_i^6}$$

```
l <- c(14.5, 12.5, 17.25, 14.5, 12.625, 17.75, 14.125, 12.625)
W <- c(27, 17, 41, 26, 17, 49, 23, 16)

k1 <- sum(l^3 * W) / sum(l^6)
```

The least squares estimate of the model is

$$W = 0.008437 l^3$$

Part b

If the product lg^2 is treated as the independent variable x, then the least squares estimate can be represented as

$$k_2 = \frac{\sum x_i W_i}{\sum x_i^2} = \frac{\sum l_i g_i^2 W_i}{\sum (l_i g_i^2)^2} = \frac{\sum l_i g_i^2 W_i}{\sum l_i^2 g_i^4}$$

```
g <- c(9.75, 8.375, 11.0, 9.75, 8.5, 12.5, 9.0, 8.5)

k2 <- sum(l * g^2 * W) / sum(l^2 * g^4)
```

The least squares estimate of the model is

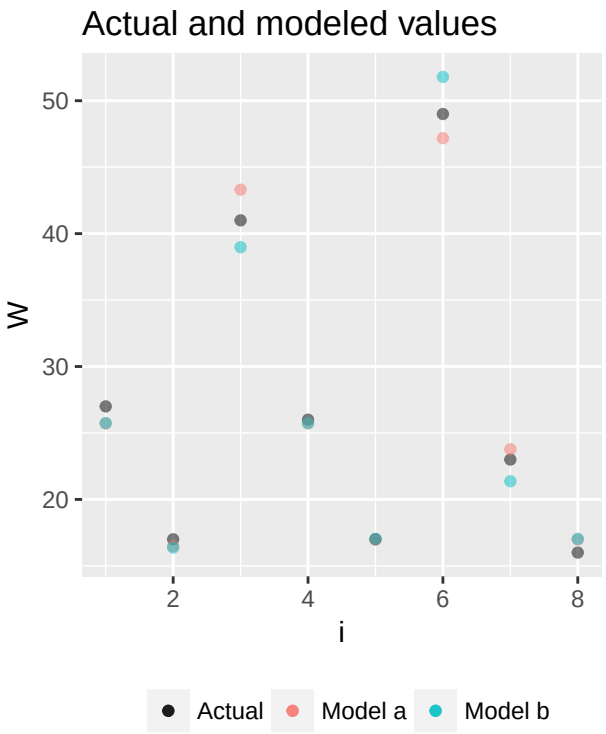
$$W = 0.018675 lg^2$$

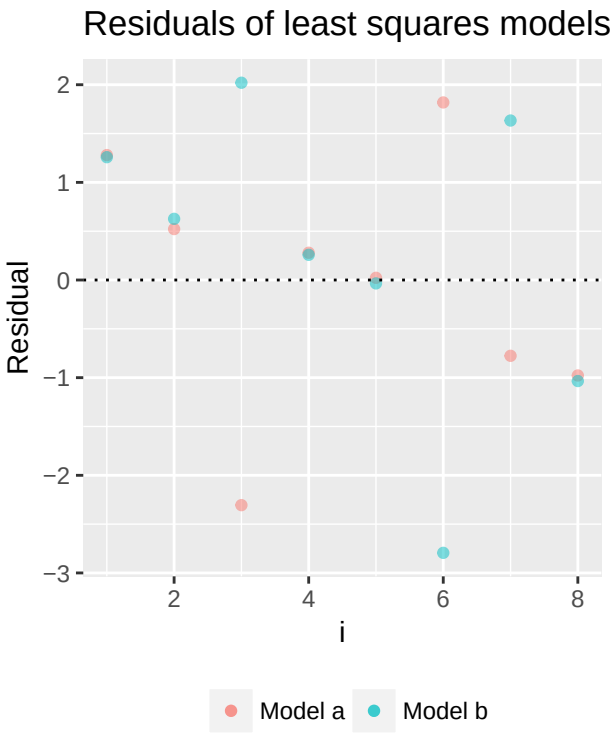
Part c

```
W_a <- k1 * l^3
resid_a <- W - W_a
W_b <- k2 * l * g^2
resid_b <- W - W_b
```

The model from part a ($W = 0.008437l^3$) fits the data better – as shown in the table below, has lower maximum deviation and sum of squared deviations. The plots below show the actual and modeled values, as well as the residuals, for both models, using the index i as the x-axis.

	a	b
Maximum Deviation	2.305	2.794
Sum of Squared Deviations	12.17	17.67





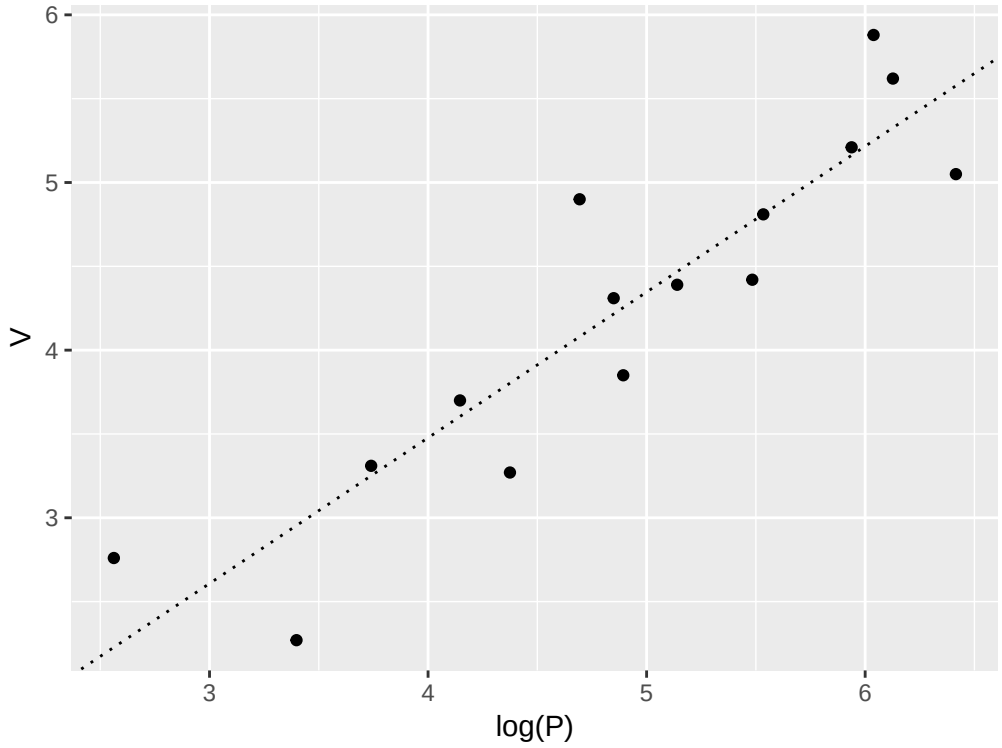
Chapter 4: Experimental Modeling

Section 4.1, Problem 5

Part 1

For the model $V = m(\log P) + b$, the transformation suggested in Problem 1 is not necessary. As such, further steps will be continued using V and $\log P$, using the base-10 logarithm.

V	$\log P$
4.81	5.534
5.88	6.039
3.31	3.74
4.9	4.694
5.62	6.127
2.76	2.562
2.27	3.398
3.85	4.893
5.21	5.938
3.7	4.146
3.27	4.375
4.31	4.849
4.42	5.484
4.39	5.14
5.05	6.415



The least-squares estimate for the slope is given by

$$\begin{aligned} m &= \frac{15\sum(\log P_i)(V_i) - (\sum \log P_i)(\sum V_i)}{15\sum(\log P_i)^2 - (\sum \log P_i)^2} \\ &= 0.860936 \end{aligned}$$

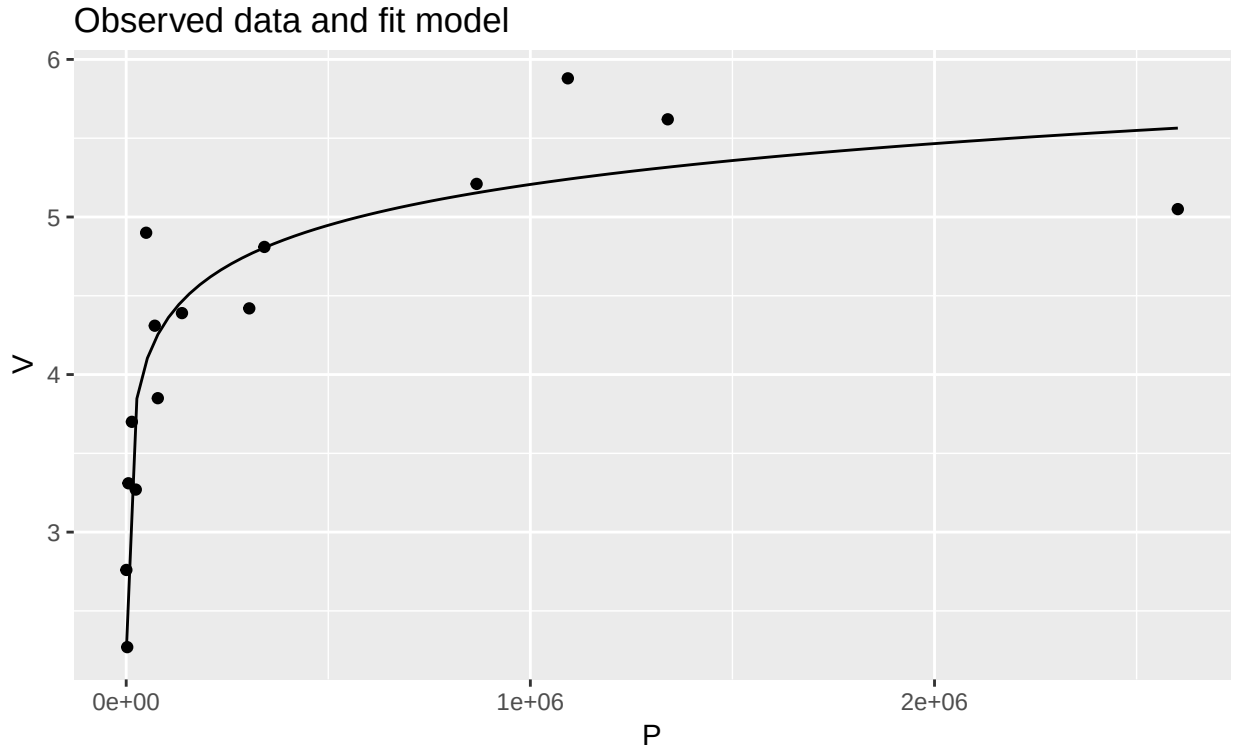
The least-squares estimate for the the intercept is given by

$$\begin{aligned} b &= \frac{\sum(\log P_i^2)\sum V_i - \sum(\log P_i V_i)\sum(\log P_i)}{15\sum(\log P_i)^2 - (\sum \log P_i)^2} \\ &= 0.040981 \end{aligned}$$

The linear equation is given by

$$V = 0.860936 \log P + 0.040981$$

Part 2



Part 3

```
V_pred <- m * log10(P) + b
```

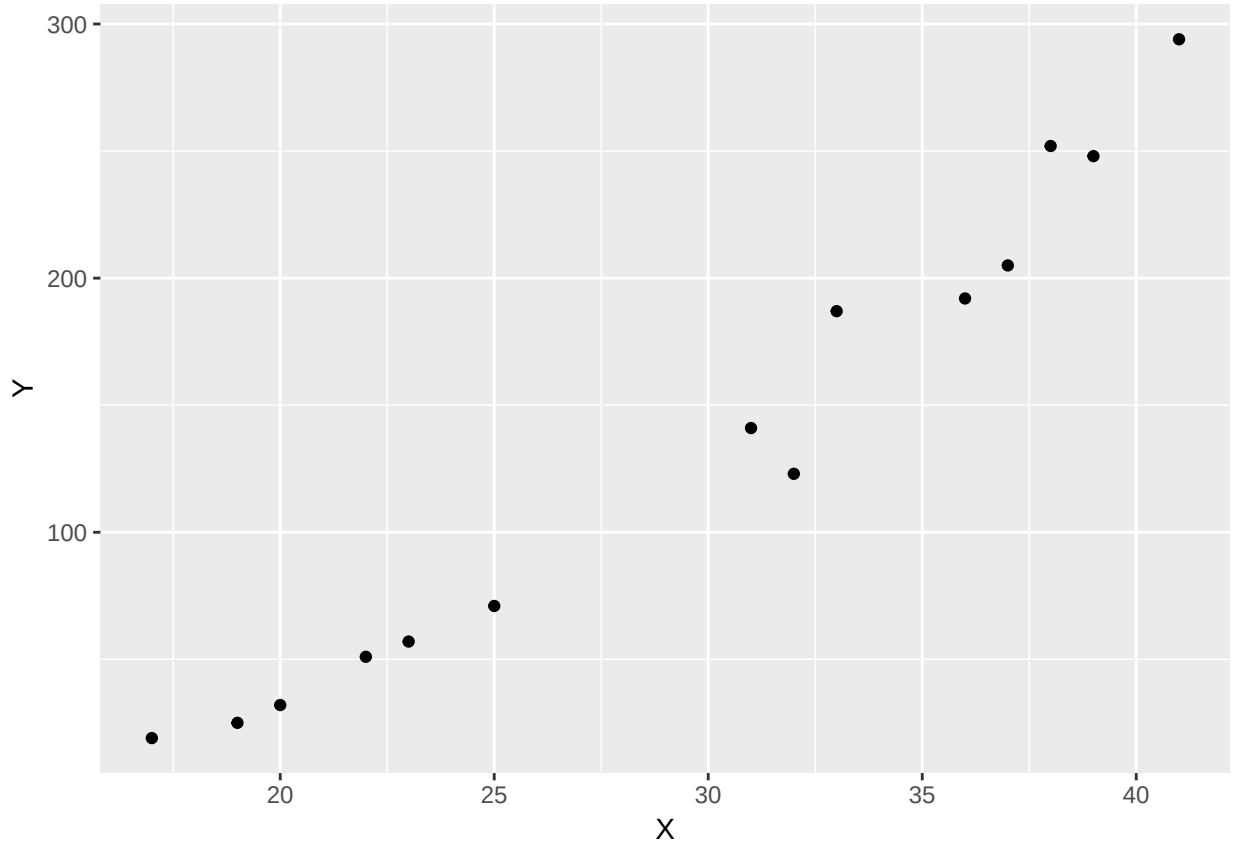
	$V_{observed}$	$V_{predicted}$
Brno, Czechoslovakia	4.81	4.805
Prague, Czechoslovakia	5.88	5.24
Corte, Corsica	3.31	3.261
Bastia, France	4.9	4.082
Munich, Germany	5.62	5.316
Psychro, Crete	2.76	2.247
Itea, Greece	2.27	2.966
Iraklion, Greece	3.85	4.254
Athens, Greece	5.21	5.153
Safed, Israel	3.7	3.611
Dimona, Israel	3.27	3.807
Netanya, Israel	4.31	4.216
Jerusalem, Israel	4.42	4.762
New Haven, USA	4.39	4.466
Brooklyn, USA	5.05	5.564

Part 4

The average Bornstein error for this model is 0.3426 – this is fairly good, as it represents a difference of roughly 8% of the mean of the mean value of V. The model $V = 1.396P^{0.096}$ obtained in Problem 1 returns a slightly better mean Bornstein error of 0.3389.

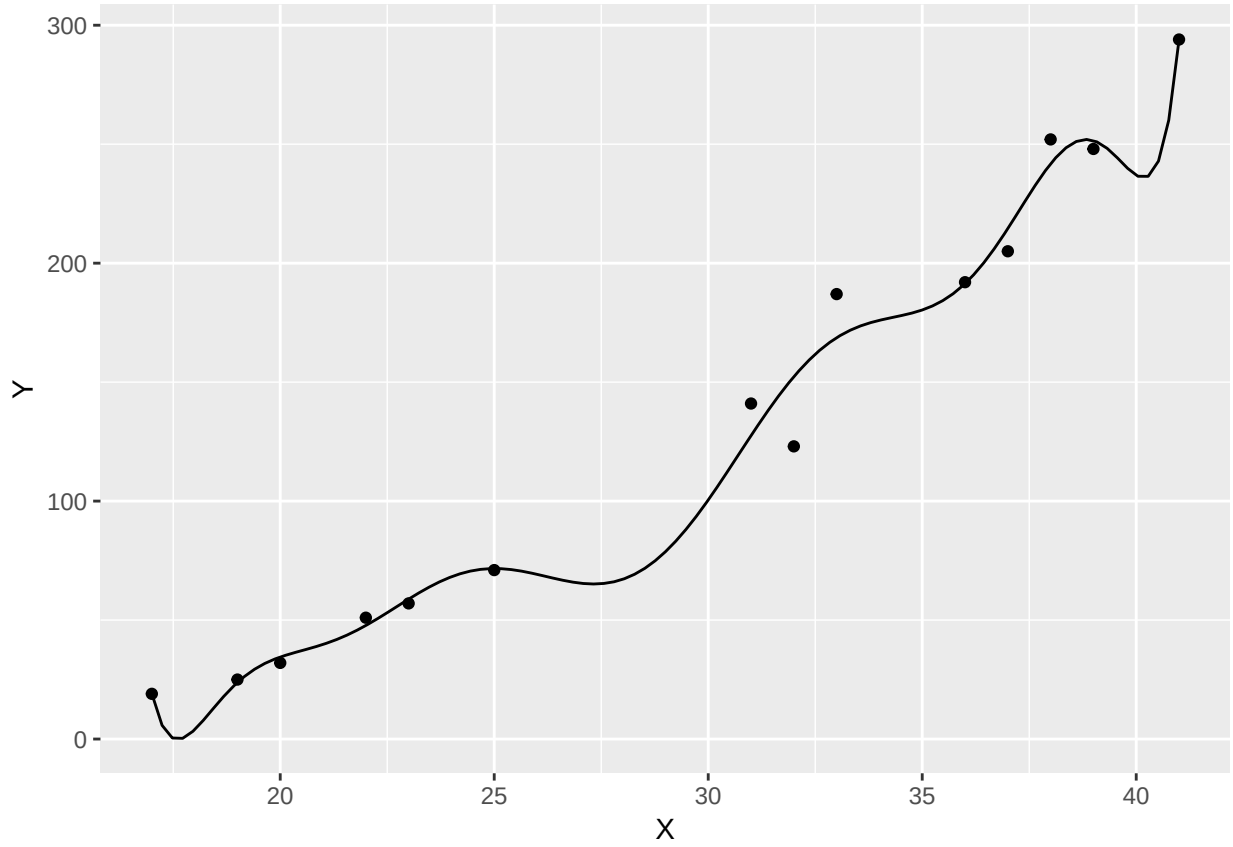
Section 4.2, Problem 4

```
X <- c(17, 19, 20, 22, 23, 25, 31, 32, 33, 36, 37, 38, 39, 41)
Y <- c(19, 25, 32, 51, 57, 71, 141, 123, 187, 192, 205, 252, 248, 294)
```



Since there are 14 data points, fitting a 13th-degree polynomial would likely not be appropriate. There is a polynomial of order 13 that will pass through every data point, but this polynomial will not likely be useful for interpolation between data point or outside the observed range of data. The best-fit 13th-degree polynomial using the least-squares criteria is graphed below:

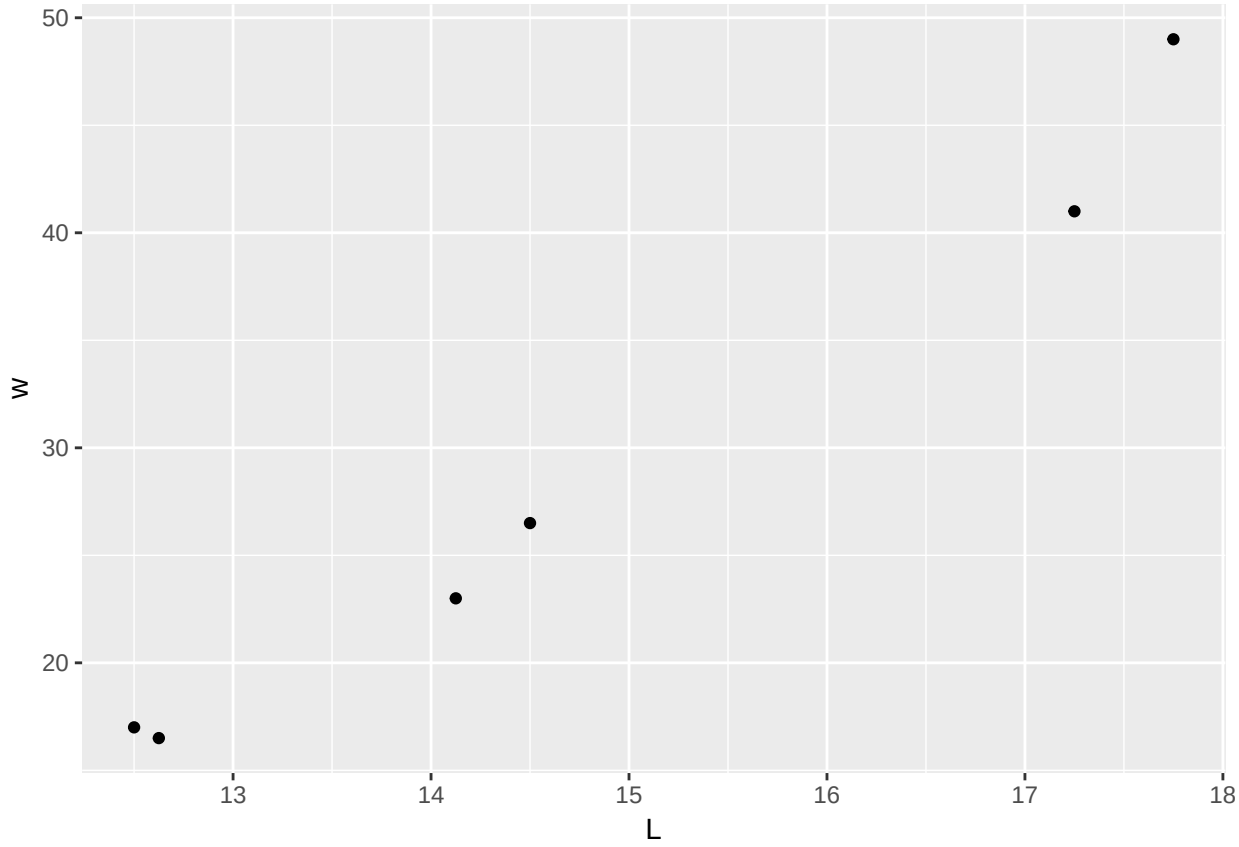
```
poly13 <- lm(Y ~ poly(X, 13, raw = TRUE))
poly13$coefficients[is.na(poly13$coefficients)] <- 0
```



This plot clearly illustrates one of the dangers of fitting high-degree polynomials to data – there is a great deal of oscillation between the points, epecially at the beginning and end of the data range.

Section 4.3, Problem 11

```
L <- c(12.5, 12.625, 14.125, 14.5, 17.25, 17.75)
w <- c(17, 16.5, 23, 26.5, 41, 49)
```



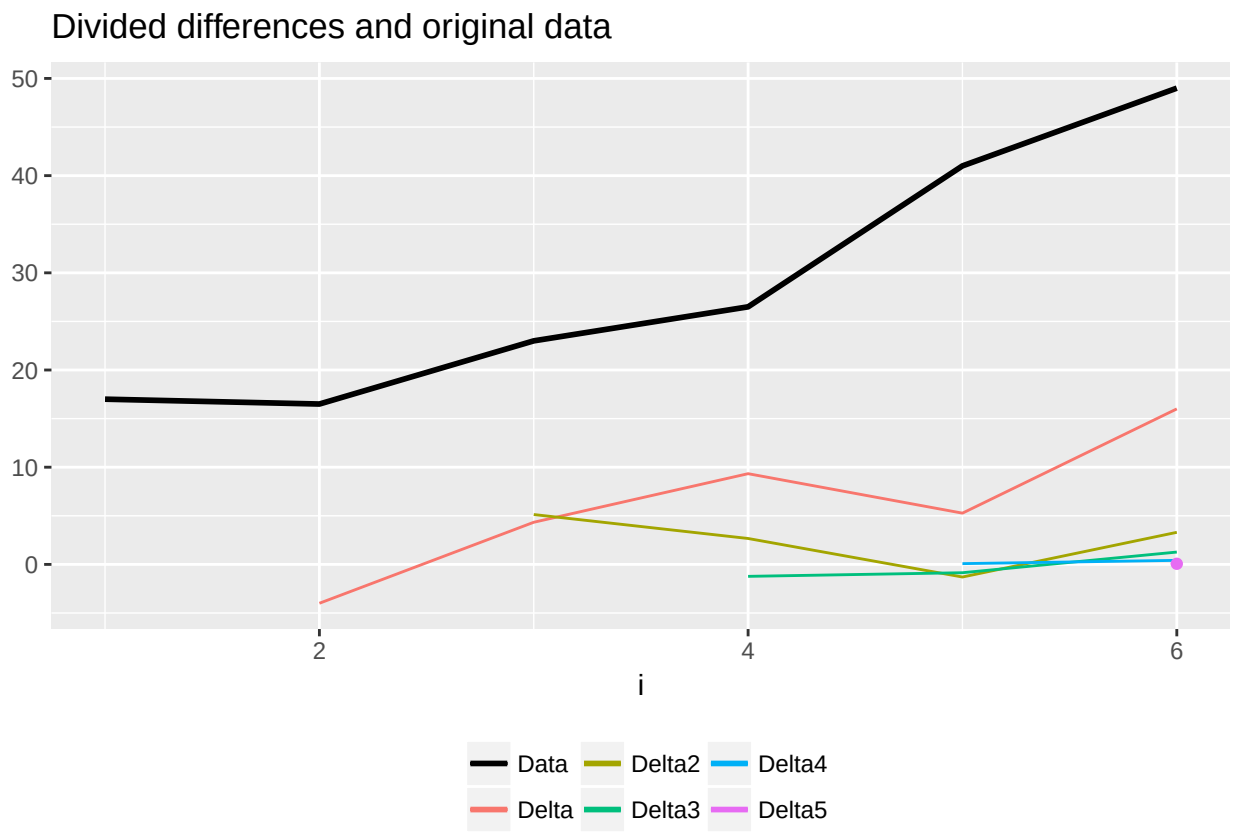
There appears to be an upward trend in the data, but any concavity is not apparent due to the low number of data points. None of the data points immediately appear to be outliers.

A divided difference table is prepared below:

```
library(dplyr)
df_bass <- data.frame(L, w) %>%
  mutate(
    del1 = (w - lag(w)) / (L - lag(L)),
    del2 = (del1 - lag(del1)) / (L - lag(L, 2)),
    del3 = (del2 - lag(del2)) / (L - lag(L, 3)),
    del4 = (del3 - lag(del3)) / (L - lag(L, 4)),
    del5 = (del4 - lag(del4)) / (L - lag(L, 5)))
```

L	w	Δ	Δ^2	Δ^3	Δ^4	Δ^5
12.5	17					
12.62	16.5	-4				
14.12	23	4.3333	5.1282			
14.5	26.5	9.3333	2.6667	-1.2308		
17.25	41	5.2727	-1.2994	-0.8575	0.0786	
17.75	49	16	3.3007	1.269	0.4149	0.0641

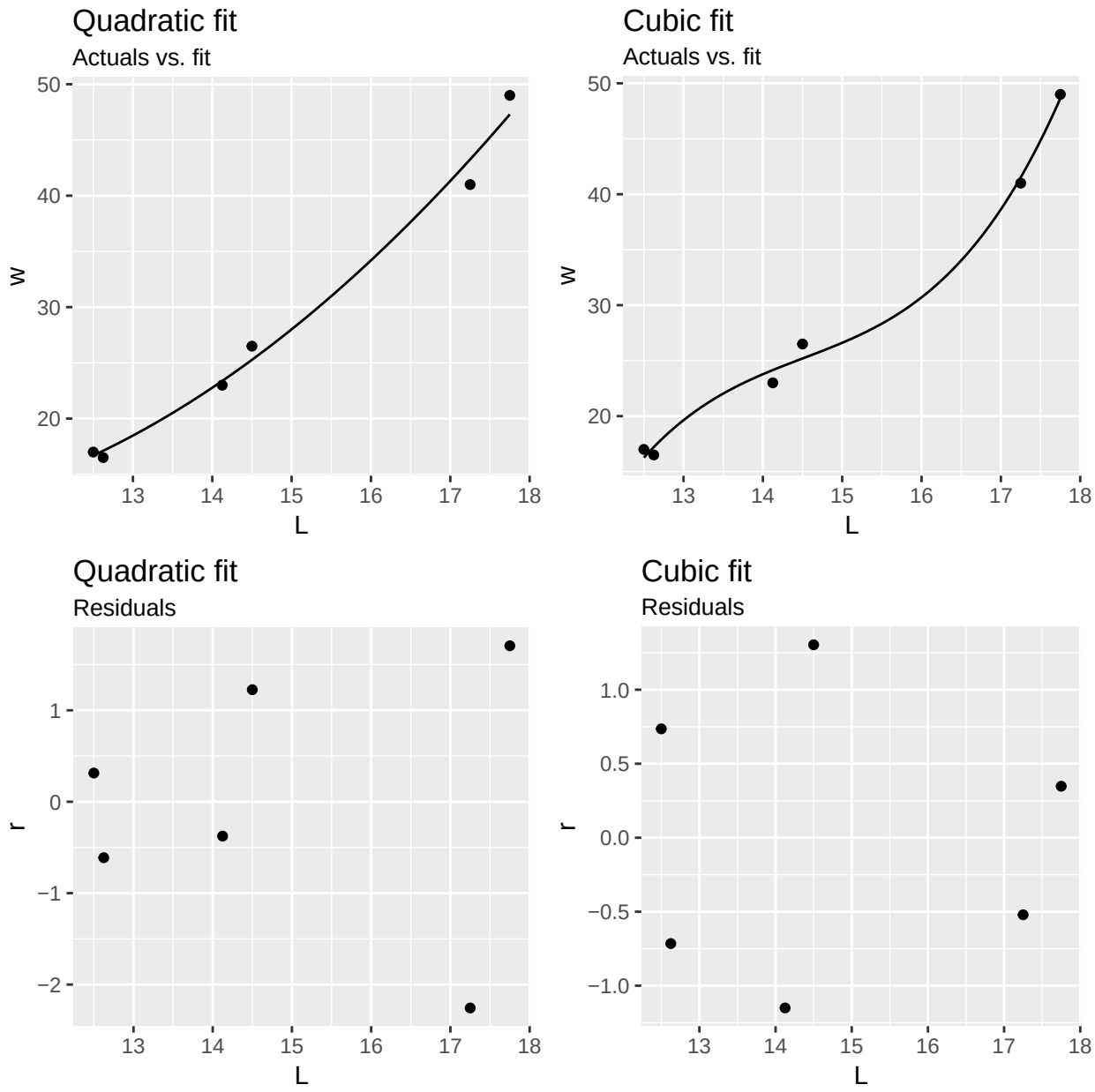
The divided differences are plotted alongside the original data below:



The table and plot show that the first divided difference Δ is roughly constantly increasing. The second divided difference Δ^2 appears to be closer to constant (relative to the data), with a significant change in sign for the third value. The third divided difference Δ^3 is even closer to constant, and has values of low magnitude on either side of zero. The fourth divided difference Δ^4 is definitely constant and very close to zero. Based on this, a second- or third-order polynomial may be a good fit to the data; both are investigated by generating a least-squares estimate.

```
poly2 <- lm(w ~ poly(L, 2, raw = TRUE))
poly3 <- lm(w ~ poly(L, 3, raw = TRUE))
```

The two fits and their residuals are plotted below.



The cubic fit appears to fit the data slightly better – this is confirmed by its lower sum of squared deviations (4.466908 vs. 10.1192309) and maximum deviation (1.3033402 vs. 2.2566506), but it exhibits the oscillation that can be a drawback of higher-order polynomials. Neither fit exhibits any pattern in the residuals. Based on the simplicity of the models and the low number of data points, the best low-order polynomial for the data is the quadratic with equation

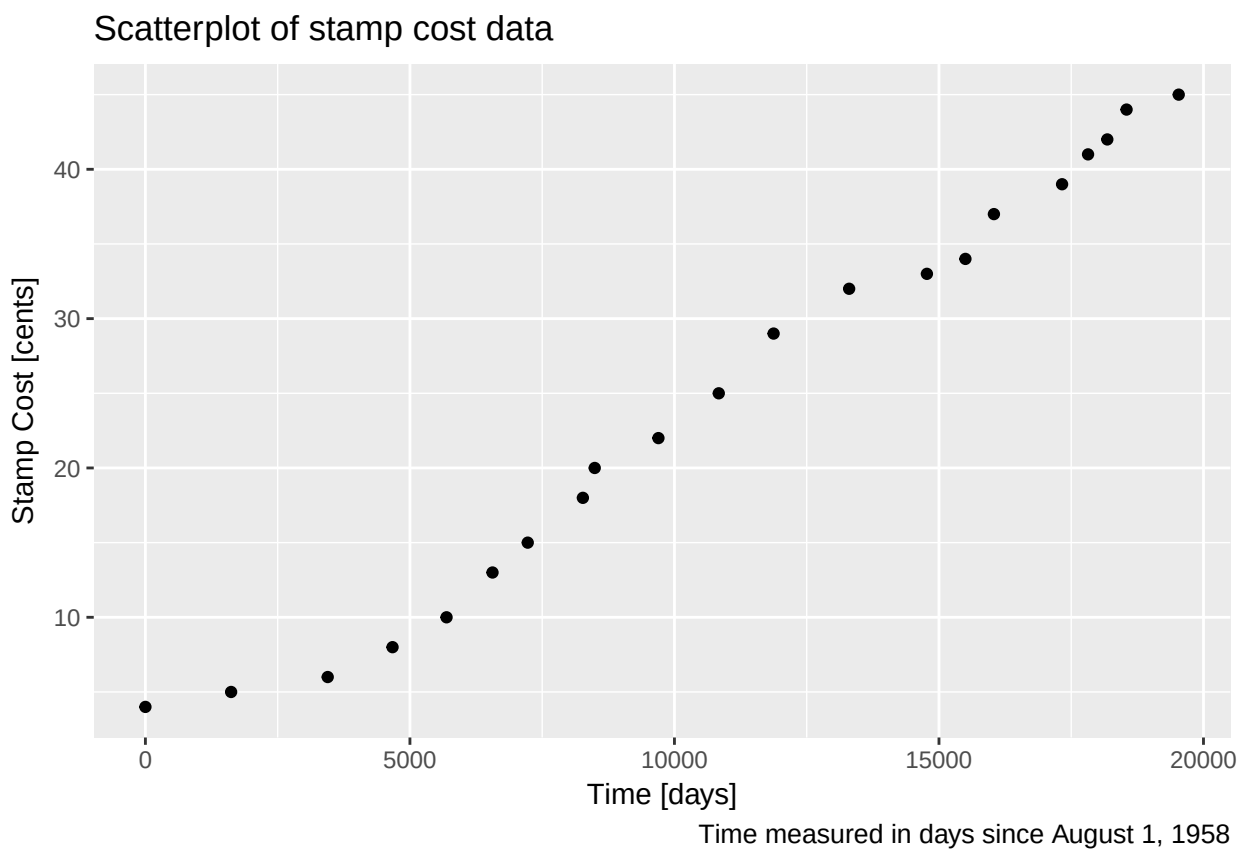
$$w = 48.6796 - 8.4676L + 0.4726L^2$$

Section 4.4, Problem 5

Investigation of Data

The first 34 years of provided data show the same value of a stamp ($0.02), which was then briefly raised for two years before returning to the same level for an additional 13 years. The next price of $0.03 remained for an additional 26 years. This initial stability appears inconsistent with the increasing trend of other data points, so data before 1958 is excluded. The rate of $0.13 on New Years Eve 1975 is also excluded, as it represents a temporary increase.

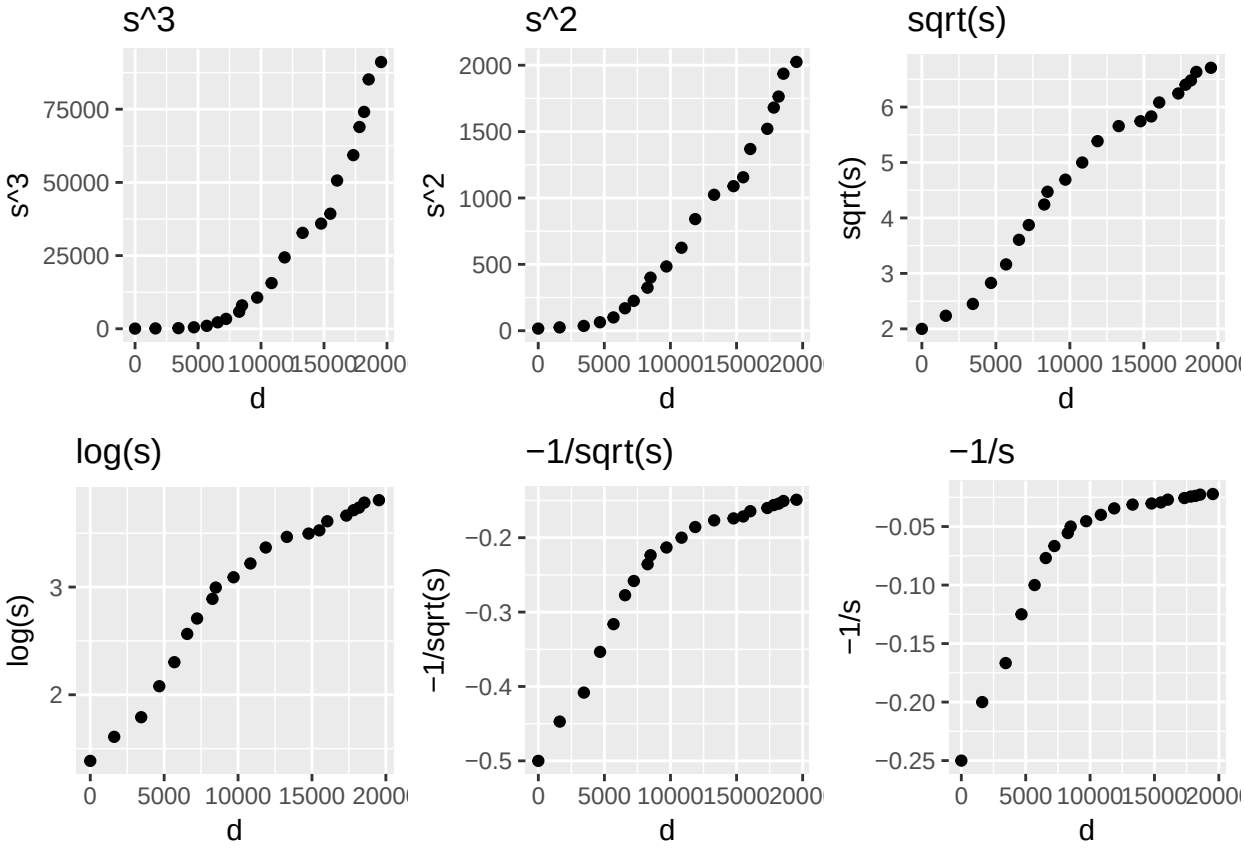
```
d <- c('1958-08-01', '1963-01-07', '1968-01-07', '1971-05-16', '1974-03-02',
       '1976-07-18', '1978-05-15', '1981-03-22', '1981-11-01', '1985-02-17',
       '1988-04-03', '1991-02-03', '1995-01-01', '1999-01-10', '2001-01-07',
       '2002-06-30', '2006-01-08', '2007-05-14', '2008-05-12', '2009-05-11',
       '2012-01-22')
s <- c(0.04, 0.05, 0.06, 0.08, 0.10, 0.13, 0.15, 0.18, 0.20, 0.22, 0.25,
       0.29, 0.32, 0.33, 0.34, 0.37, 0.39, 0.41, 0.42, 0.44, 0.45) * 100

library(lubridate)
# convert dates to days since first recorded date
d <- date(d)
d <- as.numeric(d - d[1])
```

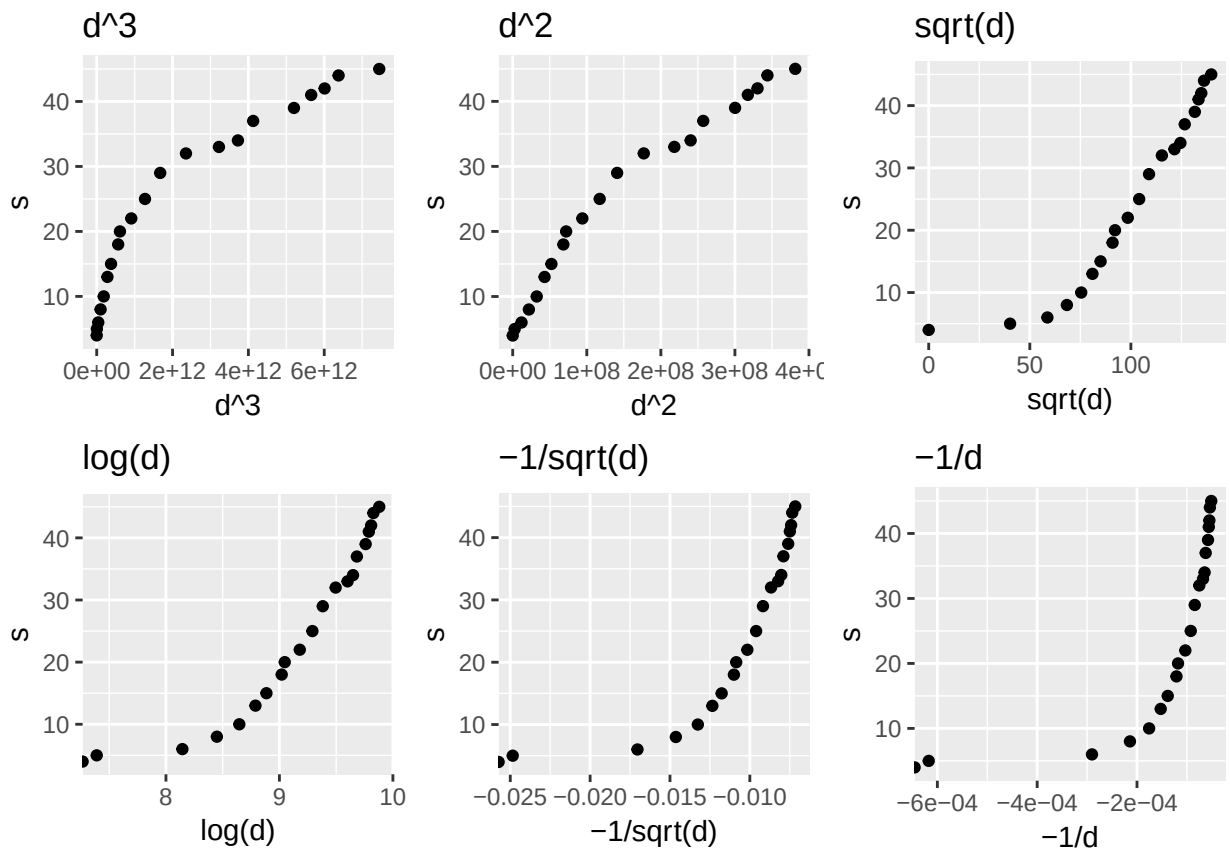


The scatterplot indicates an increasing relationship between the variables; the rate of this increase (and thus concavity of the curve) appears to vary.

One-Term Model Investigation

In order to see if there is a one-term model that may be applicable to this data, the variables d and s are transformed using the ladder of powers; since there is no clear concavity in the data, multiple transformations are attempted. The plots of these transformed variables are shown below:





Of the 12 transformations attempted, only $\sqrt{s}$ appears to create a somewhat linear relationship. This relationship is fit and investigated:

```
lm_sqrt <- lm(sqrt(s) ~ d)
```

The fit for this fit is given by

$$\sqrt{s} = 1.941532 + 0.000257d$$

Rewriting this in terms of s and d, it becomes

$$s = 6.6049 \times 10^{-8}d^2 + 9.97947 \times 10^{-4}d + 3.76955$$

This quadratic fit has a maximum deviation of 4.0685 and a sum of squared deviations of 84.7669. The maximum deviation, in particular, seems to warrant further investigation.

Polynomial Fit Investigation

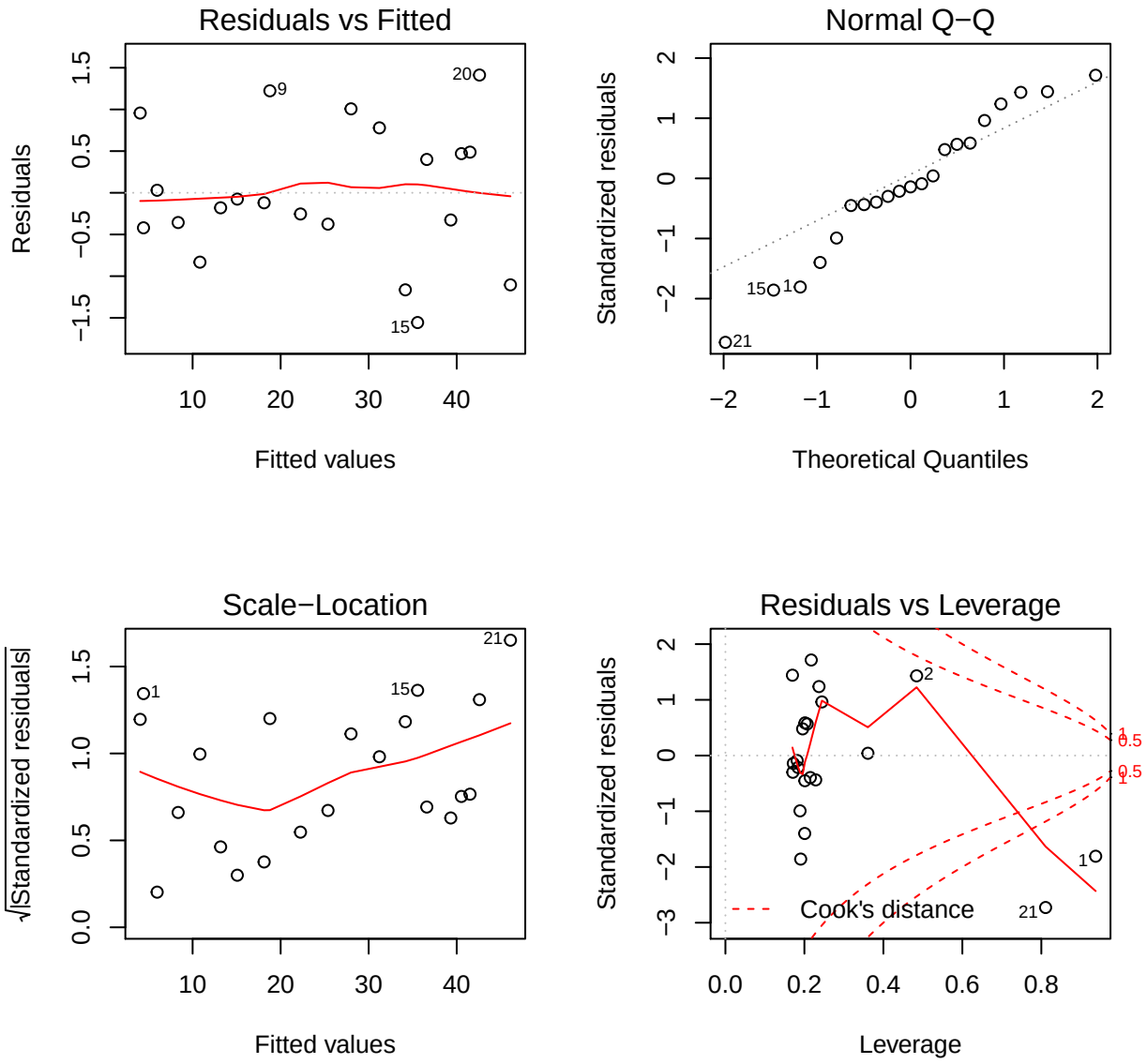
To determine if a low-order polynomial may be an appropriate fit for this data, a divided difference table is calculated:

```
df_stamp <- data.frame(d, s) %>%
  mutate(
    del1 = (s - lag(s)) / (d - lag(d)),
    del2 = (del1 - lag(del1)) / (d - lag(d, 2)),
    del3 = (del2 - lag(del2)) / (d - lag(d, 3)),
    del4 = (del3 - lag(del3)) / (d - lag(d, 4)),
    del5 = (del4 - lag(del4)) / (d - lag(d, 5)))
```

d	s	Δ	Δ^2	Δ^3	Δ^4	Δ^5
0	4	NA	NA	NA	NA	NA
1620	5	0.00062	NA	NA	NA	NA
3446	6	0.00055	-2e-08	NA	NA	NA
4671	8	0.0016	3.6e-07	8e-11	NA	NA
5692	10	0.002	1.5e-07	-5.2e-11	-2.3e-14	NA
6561	13	0.0035	7.9e-07	2.1e-10	5.2e-14	1.2e-17
7227	15	0.003	-2.9e-07	-4.2e-10	-1.7e-13	-3.9e-17
8269	18	0.0029	-7.3e-08	8.5e-11	1.4e-13	6.4e-17
8493	20	0.0089	4.8e-06	2.5e-09	8.7e-13	1.9e-16
9697	22	0.0017	-5.1e-06	-4e-09	-2.1e-12	-7.3e-16
10838	25	0.0026	4.1e-07	2.1e-09	1.7e-12	8.8e-16
11874	29	0.0039	5.7e-07	4.5e-11	-5.8e-13	-4.9e-16
13302	32	0.0021	-7.1e-07	-3.6e-10	-8.3e-14	9.9e-17
14772	33	0.00068	-4.9e-07	5.7e-11	8.1e-14	2.6e-17
15500	34	0.0014	3.2e-07	2.2e-10	3.5e-14	-7.9e-18
16039	37	0.0056	3.3e-06	1.1e-09	2.1e-13	3.3e-17
17327	39	0.0016	-2.2e-06	-2.2e-09	-8.1e-13	-1.9e-16
17818	41	0.0041	1.4e-06	1.6e-09	1.2e-12	4.5e-16
18182	42	0.0027	-1.6e-06	-1.4e-09	-1.1e-12	-6.8e-16
18546	44	0.0055	3.8e-06	4.4e-09	2.3e-12	1.1e-15
19532	45	0.001	-3.3e-06	-4.1e-09	-3.9e-12	-1.8e-15

There appear to be a high number in sign changes at Δ^5, indicating that there are variations at the fifth-order polynomial that will not be captured in lower-order polynomials. As such, a fifth-order polynomial is created using the least-squares estimate:

```
poly5 <- lm(s ~ poly(d, 5, raw = TRUE))
```



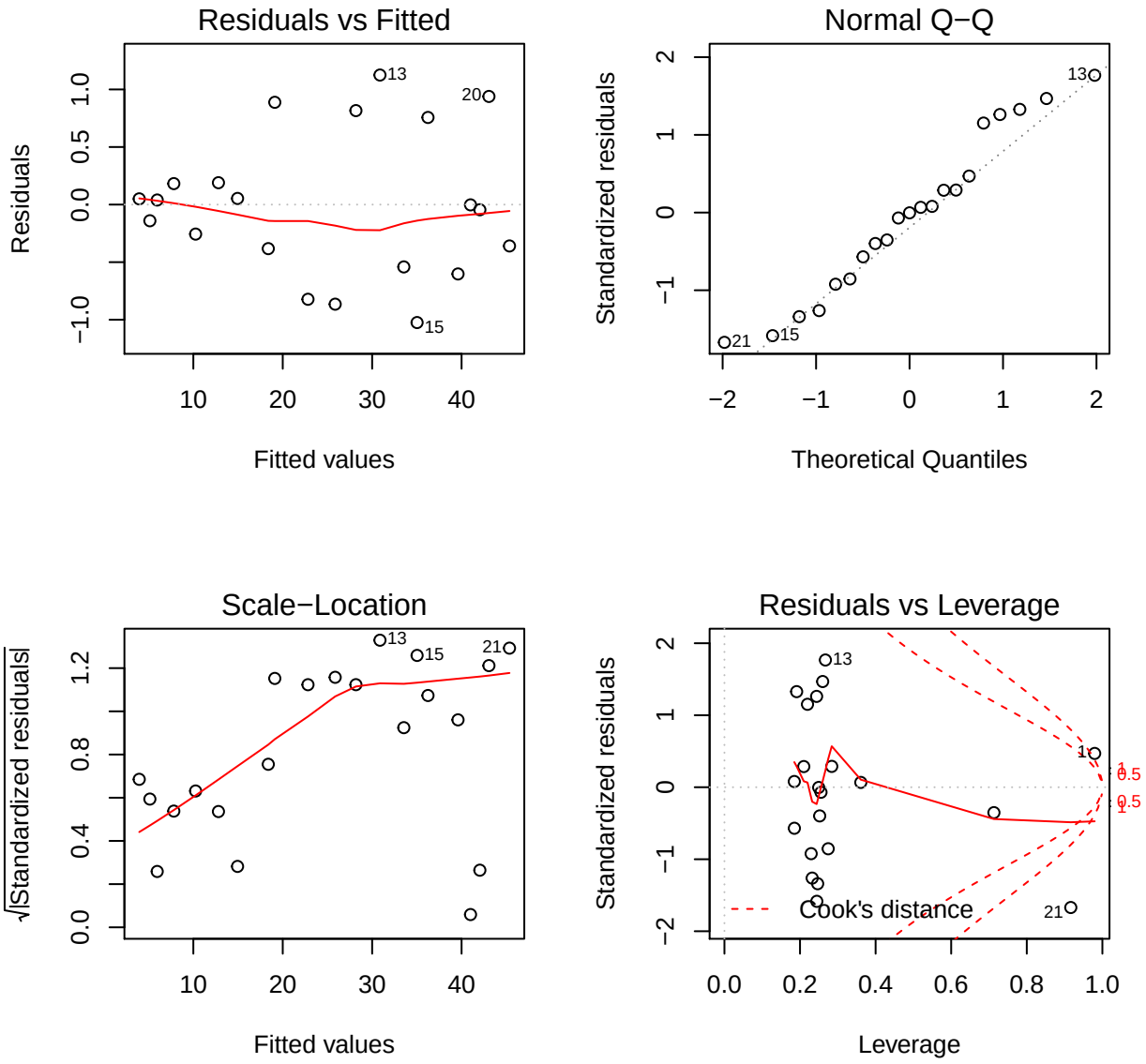
The normal Q-Q plot fit shows a significant problem with this fit – it underestimates at high values and overestimates at low values. To look into this, higher-level polynomials are considered:

```
df_stamp <- df_stamp %>%
  mutate(
    del6 = (del5 - lag(del5)) / (d - lag(d, 6)),
    del7 = (del6 - lag(del6)) / (d - lag(d, 7)),
    del8 = (del7 - lag(del7)) / (d - lag(d, 8)),
    del9 = (del8 - lag(del8)) / (d - lag(d, 9)))
```

d	s	Δ^6	Δ^7	Δ^8	Δ^9
0	4	NA	NA	NA	NA
1620	5	NA	NA	NA	NA
3446	6	NA	NA	NA	NA
4671	8	NA	NA	NA	NA
5692	10	NA	NA	NA	NA
6561	13	NA	NA	NA	NA
7227	15	-7e-21	NA	NA	NA
8269	18	1.5e-20	2.7e-24	NA	NA
8493	20	2.5e-20	1.4e-24	-1.6e-28	NA
9697	22	-1.8e-19	-3.3e-23	-4.3e-27	-4.3e-31
10838	25	3.1e-19	8.1e-23	1.5e-26	2.1e-30
11874	29	-2.6e-19	-9.3e-23	-2.4e-26	-4.7e-30
13302	32	9.7e-20	5.3e-23	1.9e-26	5e-30
14772	33	-1.1e-20	-1.4e-23	-8.2e-27	-3e-30
15500	34	-4.9e-21	8.8e-25	1.8e-27	1.1e-30
16039	37	6.5e-21	1.5e-24	8.1e-29	-2e-31
17327	39	-3.4e-20	-5.3e-24	-7.7e-28	-9.4e-32
17818	41	1.1e-19	2e-23	3.1e-27	4.2e-31
18182	42	-2.3e-19	-5.4e-23	-1e-26	-1.6e-30
18546	44	4.8e-19	1.3e-22	2.8e-26	5e-30
19532	45	-7.1e-19	-2.5e-22	-6.2e-26	-1.2e-29

The values of Δ^7 are very nearly zero, so a polynomial of degree 6 is calculated:

```
poly6 <- lm(s ~ poly(d, 6, raw = TRUE))
```

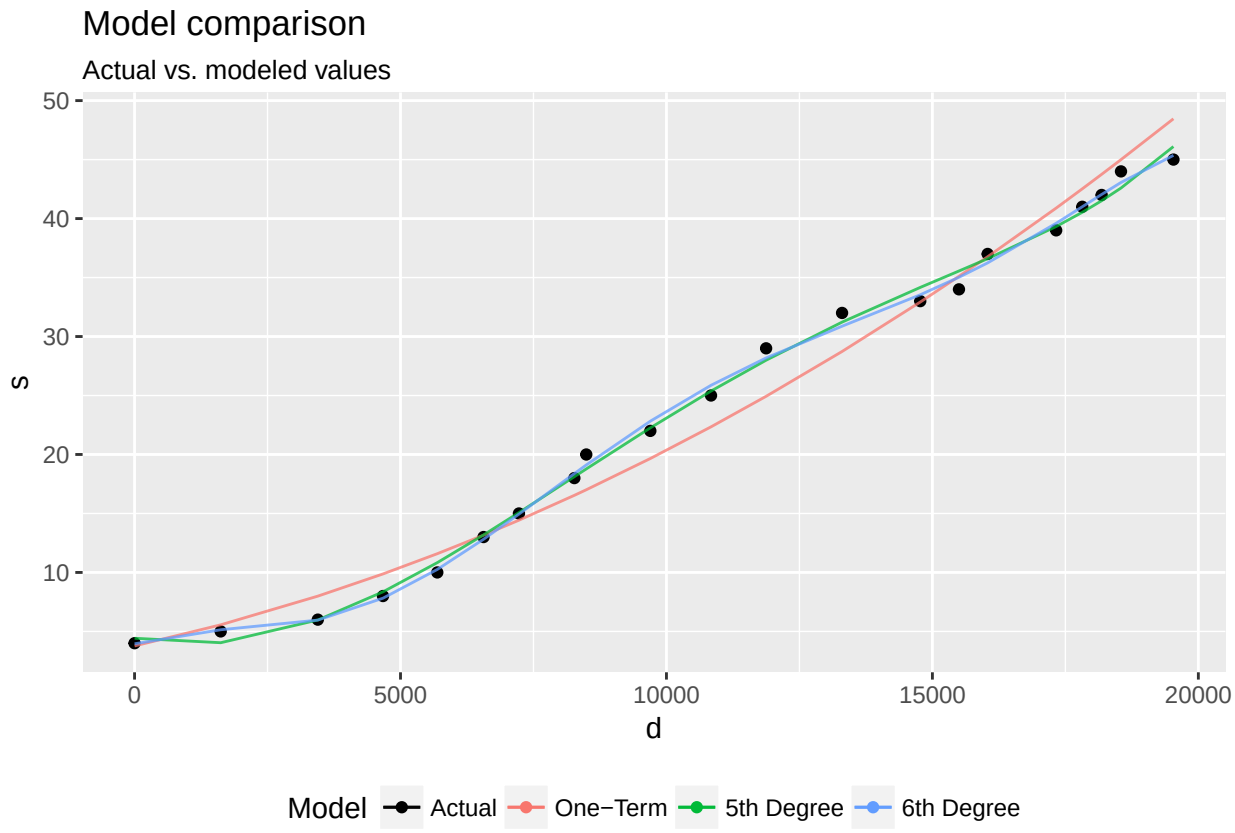


The normal Q-Q and residual vs. fitted plots look far better for this fit, so it is investigated further.

Model Selection

Measures of goodness of fit are shown below, followed by a plot of measured vs. predicted values

	One-Term	5th Degree	6th Degree
Largest Deviation	4.068	1.556	1.125
Sum of Squared Deviations	84.77	13.01	7.745



The goodness of fit measures and the plot indicate that the 6th-degree polynomial is the best model for the data. The equation for this model is

$$s = 3.9502 + 0.0019d - 1.2031 \times 10^{-6}d^2 + 3.3632 \times 10^{-10}d^3 - 3.4313 \times 10^{-14}d^4 + 1.5243 \times 10^{-18}d^5 - 2.4858 \times 10^{-23}d^6$$

Prediction

The three models investigated predict the following values for the value of a stamp on January 1, 2010 ($d = 18781$):

- One Term: \$0.4581
- 5th Degree: \$0.4334
- 6th Degree: \$0.4369

To get the date at which the value of a stamp will be \$1.00, the roots of the equation $\hat{s}(d) - 100$ are found.

The resulting dates for the three models are:

- One Term: June 5, 2044 ($d = 31355.91$)
- 5th Degree: October 15, 2026 ($d = 24912.76$)
- 6th Degree: *No solution*

As shown above, the 6th degree polynomial fit does not have a real solution – just beyond the range of the given data, the model very quickly predicts decreasing, and then highly negative, values. While this model does a good job of explaining the relationship between the variables d and s in the range of observed data, the high degree of the model leads to the model not being useful for forecasting. If longer-term forecasting is desired, a lower-level model should be used.