DATA 609: Homework 3

Model Fitting & Experimental Modeling

Aaron Grzasko

September 12, 2017

```r
# load libraries
library(dplyr)
library(tidyr)
library(knitr)
library(ggplot2)
library(latex2exp)
library(ggthemes)
library(lpSolve)
```

Page 113: Question 2

The following table gives the elongation e in inches per inch (in./in.) for a given stress S on a steel wire measured in pounds per square inch (lb/in^2). Test the the model $e = c_1 S$ by plotting the data. Estimate c_1 graphically.

```r
S <- c(5, seq(10, 100, 10))
e <- c(0, 19, 57, 94, 134, 173, 216, 256, 297, 343, 390)
mydf <- data.frame(S = S, e = e)
```

S	5	10	20	30	40	50	60	70	80	90	100
e	0	19	57	94	134	173	216	256	297	343	390

The proposed model assumes $e \propto S$. In other words, the model is a line passing through the origin. We can estimate the slope parameter, c_1, using a variety of methods.

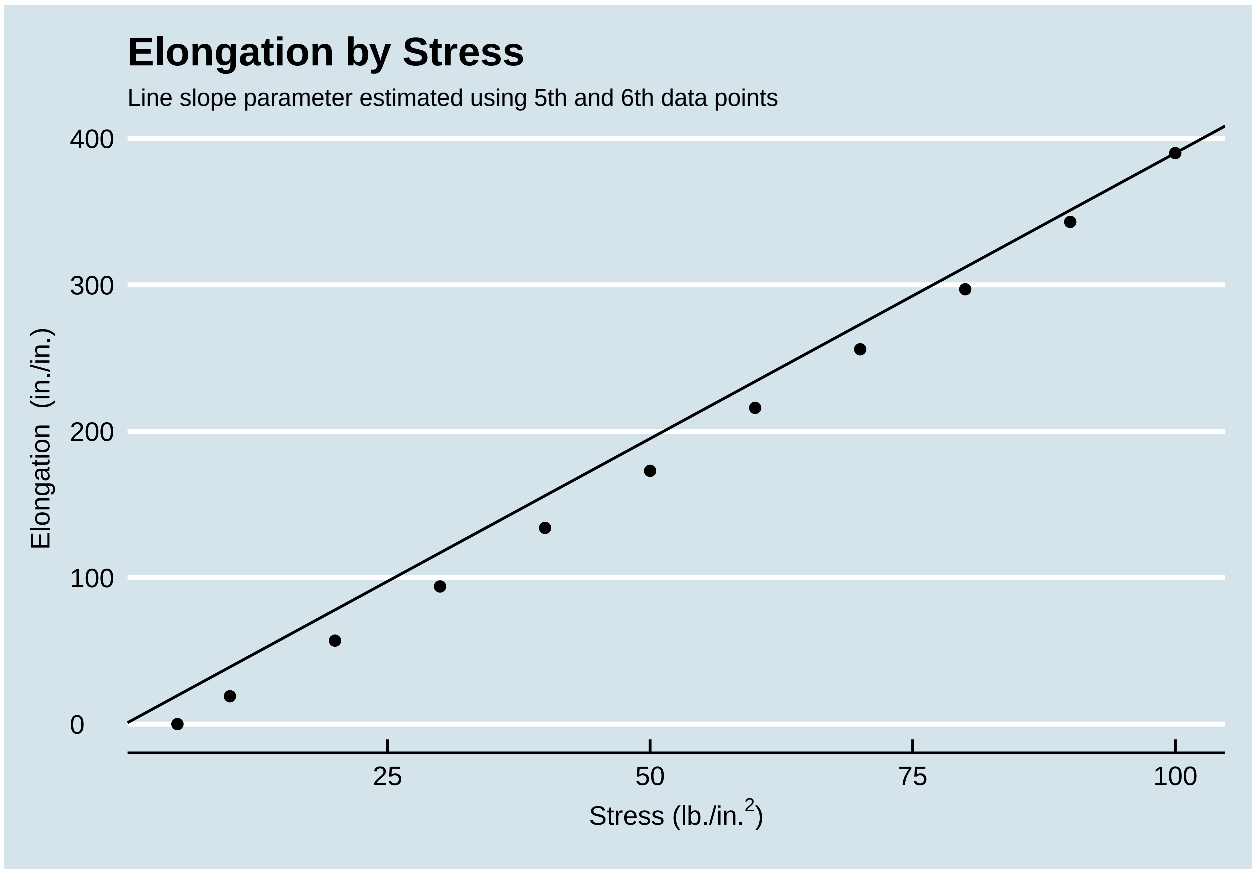
For now, we will use a simpler method that is consistent with the examples in section 3.1. That is, we will use two observed points in our data to estimate the slope.

$$c_1 = \frac{e_6 - e_5}{S_6 - S_5} = \frac{173 - 134}{50 - 40} = 3.9$$

```r
# estimated slope parameter using 5th and 6th data points
c1 <- (mydf[6, 2] - mydf[5, 2])/(mydf[6, 1] - mydf[5, 1])
c1
```

```
[1] 3.9
```

```r
# plot data with estimated line using 5th and 6th data points
g <- ggplot(mydf, aes(x = S, y = e)) + geom_point() + theme_economist()
g <- g + labs(title = "Elongation by Stress ")
g <- g + labs(subtitle = "Line slope parameter estimated using 5th and 6th data points",
    x = TeX("Stress ($lb./in.^2$)"), y = "Elongation  (in./in.)")
g <- g + geom_abline(slope = c1)
g
```



This line fits the data somewhat closely, but the fitted line overstates the observed elongation amount for at most stress levels.

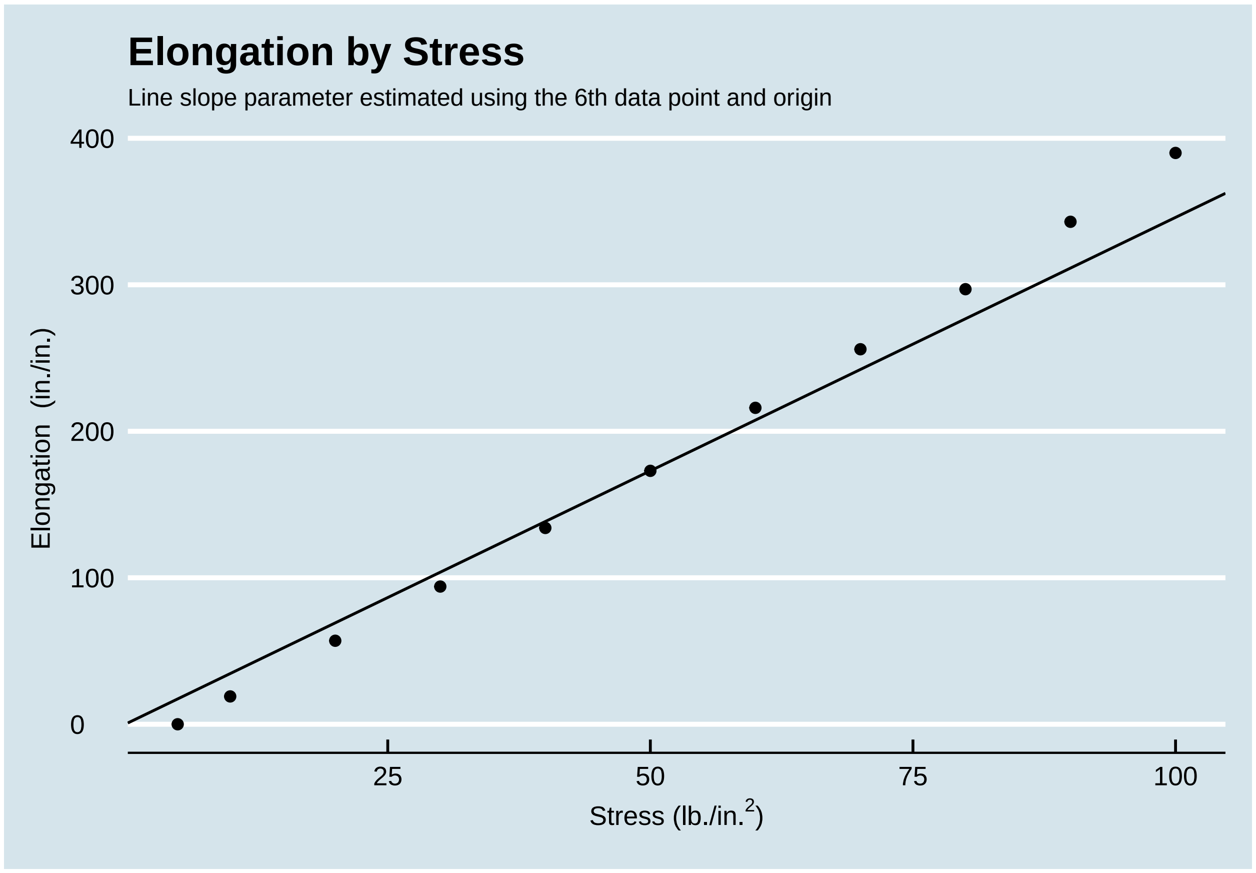
As alternative, we can estimate the slope again, using the 6th data point and the point (0,0), as our proportionality model assumes the line passes through the origin.

$$c_1 = \frac{e_6 - 0}{S_6 - 0} = \frac{173}{50} = 3.46$$

```
# estimated slope parameter using 6th data point and origin
c1 <- (mydf[6, 2])/(mydf[6, 1])
c1
```

```
[1] 3.46
```

```
# plot data with estimated line uisng 6th data point and origin
g <- ggplot(mydf, aes(x = S, y = e)) + geom_point() + theme_economist()
g <- g + labs(title = "Elongation by Stress ")
g <- g + labs(subtitle = "Line slope parameter estimated using the 6th data point and origin",
    x = TeX("Stress ($lb./in.^2$)"), y = "Elongation  (in./in.)")
g <- g + geom_abline(slope = c1)
g
```



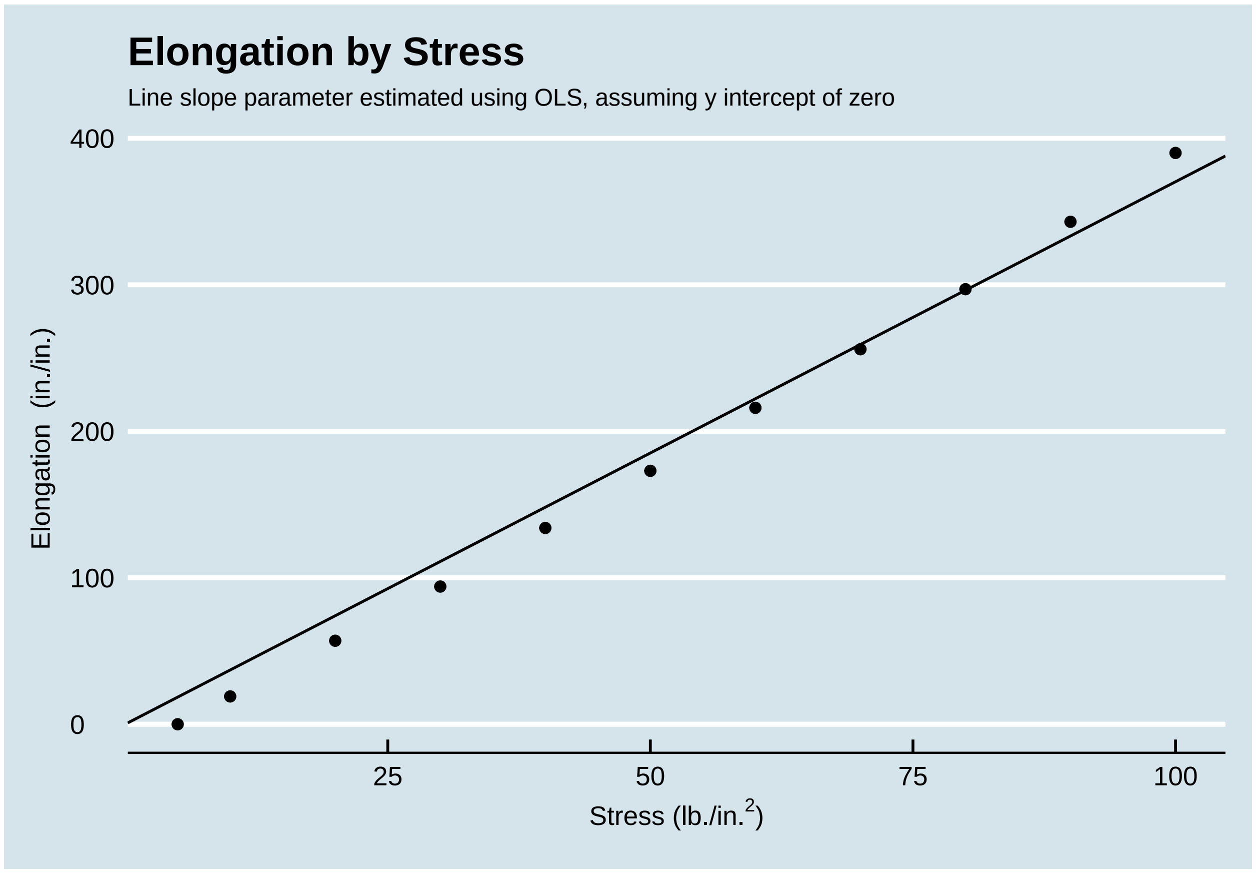
Once again, the line fits the reasonably well, but there are issues. While the fitted line passes through the 6th data point–as expected–the line overstates the observed elongation for stress amounts below 50 lb/in^2 and understates elongation for Stress amounts over 50 lb/in^2.

Now, let's attempt to fit a line using ordinary least squares, but force the y-intercept to be zero to be consistent with our proposed, proportionality model.

```
# fit ols model
mylm <- lm(e ~ S + 0, data = mydf)

# estimated slope parameter using OLS with zero intercept
c1 <- mylm$coefficients
c1
```

```
      S
3.70331
```

```
# plot data with estimated OLS line with zero y intercept
g <- ggplot(mydf, aes(x = S, y = e)) + geom_point() + theme_economist()
g <- g + labs(title = "Elongation by Stress ")
g <- g + labs(subtitle = "Line slope parameter estimated using OLS, assuming y intercept of zero",
    x = TeX("Stress ($lb./in.^2$)"), y = "Elongation  (in./in.)")
g <- g + geom_abline(slope = c1)
g
```



This model appears to be an improvement compared to the previous model, but the line is overstating elongation vis-a-vis the observed values for the majority of stress levels.

Finally, let's abandon the proportionality model in favor of a linear model with non-zero intercept, i.e. $y = c_1 x + c_0$.

```
# fit ols model
mylm <- lm(e ~ S, data = mydf)

# estimated slope parameter using OLS
c1 <- mylm$coefficients[2]
c1
```

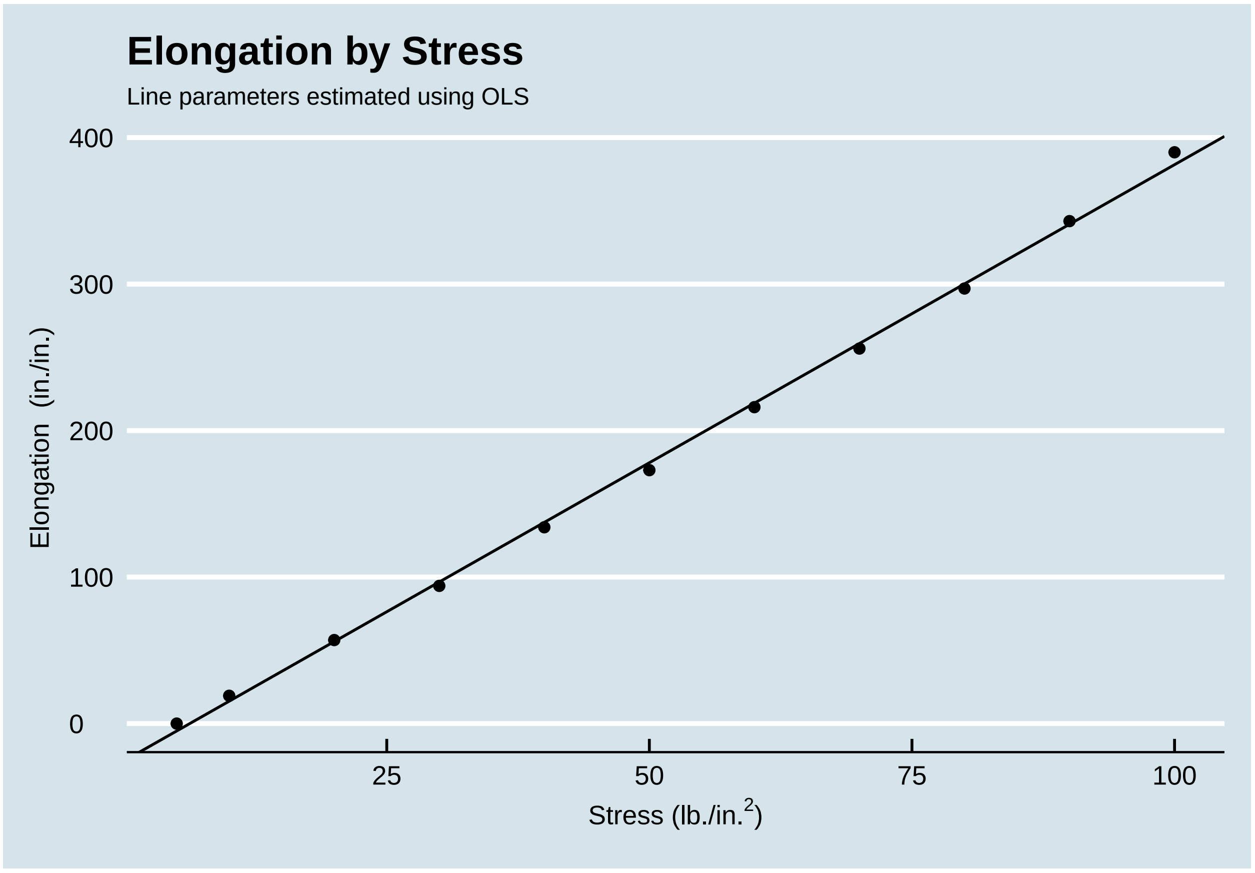
```
      S
4.06933
```

```
# estimated intercept using OLS
c0 <- mylm$coefficients[1]
c0
```

```
(Intercept)
  -25.40713
```

```
# plot data with estimated OLS line
g <- ggplot(mydf, aes(x = S, y = e)) + geom_point() + theme_economist()
g <- g + labs(title = "Elongation by Stress ")
g <- g + labs(subtitle = "Line parameters estimated using OLS", x = TeX("Stress ($lb./in.^2$)"),
    y = "Elongation  (in./in.)")
g <- g + geom_abline(slope = c1, intercept = c0)
```

```
g
```



The OLS model fits very closely with the observed data, but we have a new issue: our model assumes negative elongation amounts for stress amounts between 0 and roughly 5 lb/in^2. These negative elongation estimates are inconsistent with our physical understanding of the process being modeled. We should be cautious, therefore, in using this model for estimates involving low stress levels.

Page 121: Question 2.a

Formulate the mathematical model that minimizes the largest deviation between the data and the line $y = ax + b$. If a computer is available, solve for the estimates of a and b.

```
# data
x <- c(1, 2.3, 3.7, 4.2, 6.1, 7)
y <- c(3.6, 3, 3.2, 5.1, 5.3, 6.8)
mydf <- data.frame(x = x, y = y)
kable(mydf)
```

x	y
1.0	3.6
2.3	3.0
3.7	3.2
4.2	5.1
6.1	5.3
7.0	6.8

x	y

We are asked to fit a line to data using the Chebyshev approximation criterion.

This is linear programming problem in which we minimize r subject to the following constraints:

$$\begin{cases} r + 1.0a + 1.0b - 3.6 \geq 0 & (r - r_1 \geq 0) \\ r - 1.0a - 1.0b + 3.6 \geq 0 & (r + r_1 \geq 0) \\ r + 1.0a + 2.3b - 3.0 \geq 0 & (r - r_2 \geq 0) \\ r - 1.0a - 2.3b + 3.0 \geq 0 & (r + r_2 \geq 0) \\ r + 1.0a + 3.7b - 3.2 \geq 0 & (r - r_3 \geq 0) \\ r - 1.0a - 3.7b + 3.2 \geq 0 & (r + r_3 \geq 0) \\ r + 1.0a + 4.2b - 5.1 \geq 0 & (r - r_4 \geq 0) \\ r - 1.0a - 4.2b + 5.1 \geq 0 & (r + r_4 \geq 0) \\ r + 1.0a + 6.1b - 5.3 \geq 0 & (r - r_5 \geq 0) \\ r - 1.0a - 6.1b + 5.3 \geq 0 & (r + r_5 \geq 0) \\ r + 1.0a + 7.0b - 6.8 \geq 0 & (r - r_6 \geq 0) \\ r - 1.0a - 7.0b + 6.8 \geq 0 & (r + r_6 \geq 0) \end{cases}$$

The *R* language has a variety of libraries dedicated to solving linear programming problems. Below, we use the `lp()` function in the lpSolve package.

```r
# objective function: r + 0*a + 0*b
obj <- c(1, 0, 0)

# matrix of coefficients in constraint list above
col1 <- rep(1, 12)
col2 <- c(1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1)
col3 <- c(1, -1, 2.3, -2.3, 3.7, -3.7, 4.2, -4.2, 6.1, -6.1, 7, -7)
mymatrix <- cbind(col1, col2, col3)

# constants in constraint inequalities; signs reversed to indicate move to
# right hand side of inequalities
rightvec <- c(3.6, -3.6, 3, -3, 3.2, -3.2, 5.1, -5.1, 5.3, -5.3, 6.8, -6.8)

# solve linear program using lp() in lpSolve package
mylp <- lp("min", obj, mymatrix, rep(">=", 12), rightvec)

# print coefficients r, a, and b, respectively
mylp$solution
```

```
[1] 0.9200000 2.1466667 0.5333333
```

We can also solve this problem using the more general, `optim()` function in the stats package. Solving the problem in this manner is easier to set up, as we do not need to explicitly describe initial constraints.

```r
# function to calculate maximum residual, given linear model parameters
# inputs: parameter vector (intercept, slope), independent var vector,
# dependent var vector
max_resid_calc <- function(params, x, y) {
    r_max <- 0  # initialize maximum residual at 0

    for (i in 1:length(x)) {
        r <- abs(y[i] - params[1] - params[2] * x[i])
        r_max <- max(r_max, r)
    }
    r_max
}

# use optim() and max_resid_calc to optimize parameters to minimize the max
# residual Note: optim() requires initial guesses for slope and intercept
optim_solve <- optim(c(1, 1), max_resid_calc, x = x, y = y)

# print intercept and slope parameters, respectively
optim_solve$par
```

```
[1] 2.1466666 0.5333333
```

```
# print r value
optim_solve$value
```

```
[1] 0.92
```

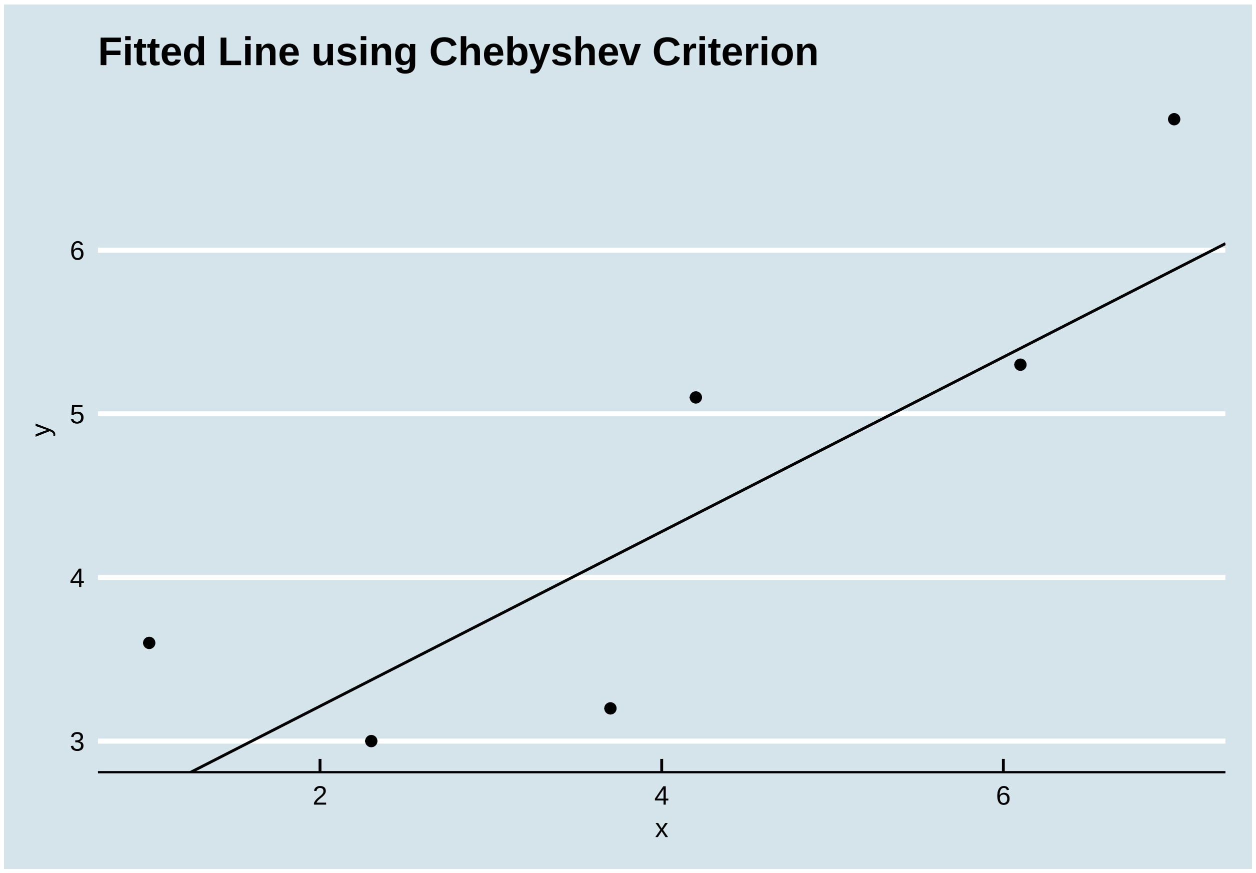
```
# See https://stackoverflow.com/questions/21845429/r-minimize-absolute-error
# for additional information
```

Our best-fit line using the Chebyshev criterion is:

$$y = 0.533x + 2.1467$$

Now let's plot the fitted line:

```
# plot using Chebyshev criterion
g <- ggplot(mydf, aes(x = x, y = y)) + geom_point() + theme_economist()
g <- g + labs(title = "Fitted Line using Chebyshev Criterion ")
g <- g + geom_abline(slope = mylp$solution[3], intercept = mylp$solution[2])
g
```



For comparison purposes, let's now fit an OLS line to our data:

```r
# set up OLS model
mylm <- lm(y ~ x, data = mydf)

# OLS model Intercept and slope parameter, respectively
mylm$coefficients
```

```
(Intercept)           x 
  2.2148534   0.5642337 
```

```r
# maximum absolute value of residuals
max(abs(mylm$residuals))
```

```
[1] 1.102518
```

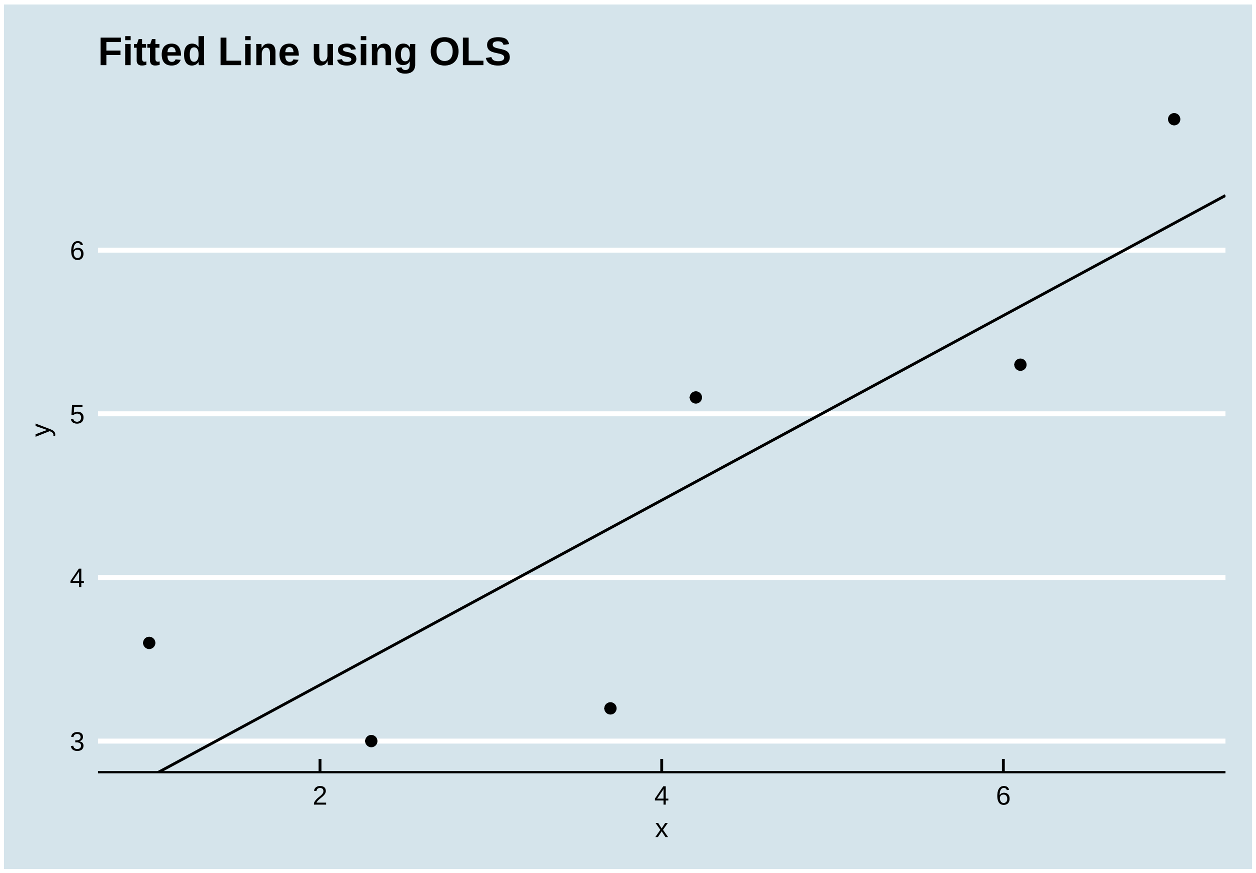
Our model using standard OLS is:

$$y = 0.5642 + 2.2149$$

We see that the fitted slope and intercept for the Chebyshev and OLS models are fairly similar. As expected, the maximum residual under the OLS model is higher than the Chebyshev model: 1.10 vs. 0.92, respectively.

Now, let's plot the OLS model:

```r
# plot OLS model
g <- ggplot(mydf, aes(x = x, y = y)) + geom_point() + theme_economist()
g <- g + labs(title = "Fitted Line using OLS")
g <- g + geom_abline(slope = mylm$coefficients[2], intercept = mylm$coefficients[1])
g
```



Page 127, Question 10

Fit the data with the models given, using least squares

```
# data
body <- c("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranus",
    "Jupiter")

period <- c(7.6 * 10^6, 1.94 * 10^7, 3.16 * 10^7, 5.94 * 10^7, 3.74 * 10^8,
    9.35 * 10^8, 2.64 * 10^9, 5.22 * 10^9)

distance <- c(5.79 * 10^10, 1.08 * 10^11, 1.5 * 10^11, 2.28 * 10^11, 7.79 *
    10^11, 1.43 * 10^12, 2.87 * 10^12, 4.5 * 10^12)

mydf <- data.frame(body = body, period = period, distance = distance)

# print table
kable(mydf)
```

body	period	distance
Mercury	7.60e+06	5.79e+10
Venus	1.94e+07	1.08e+11
Earth	3.16e+07	1.50e+11
Mars	5.94e+07	2.28e+11

body	period	distance
Jupiter	3.74e+08	7.79e+11
Saturn	9.35e+08	1.43e+12
Uranus	2.64e+09	2.87e+12
Jupiter	5.22e+09	4.50e+12

Fit the model $y = ax^{3/2}$

For this problem, we assume *period* is the x variable, and *distance* is the y variable.

To solve for a, we need to minimize the following:

$$S = \sum_{i=1}^{m}[y_i - f(x_i)]^2 = \sum_{i=1}^{m}[y_i - ax_i^{3/2}]^2$$

We find the minimum by taking the derivative of the above equation with respect to a:

$$\frac{dS}{da} = -2\sum_{i=1}^{m} x_i^{3/2}[y_i - ax_i^{3/2}] = 0$$

With basic algebraic manipulation, we can solve for a:

$$a = \frac{\sum_{i=1}^{m} x_i^{3/2} y_i}{\sum_{i=1}^{m} x_i^3}$$

```r
# solve for a
a <- sum((mydf$period^(3/2) * mydf$distance))/sum(mydf$period^3)
a
```

```
[1] 0.01320756
```

Let's check our work using the nonlinear least squares, `nls()`, function in the stats package:

```r
# nonlinear
my_model <- nls(distance ~ a * period^(3/2), data = mydf, start = list(a = 0.5))

# model output
my_model
```

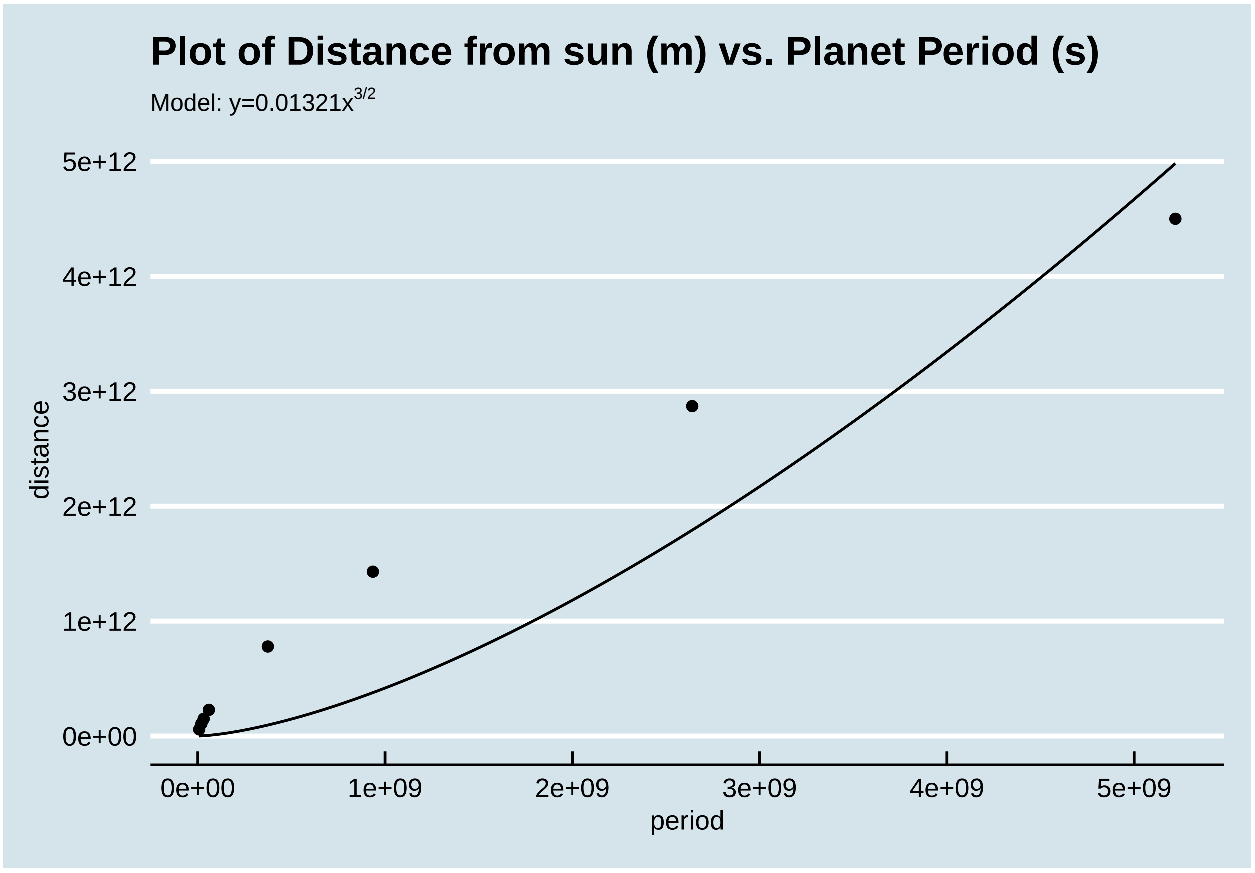
```
Nonlinear regression model
  model: distance ~ a * period^(3/2)
   data: mydf
      a
0.01321
 residual sum-of-squares: 3.055e+24

Number of iterations to convergence: 1
Achieved convergence tolerance: 2.825e-07
```

The estimate for a using the `nls()` function, 0.01321, is consistent with our analytical solution above.

Our final model is $y = 0.01321x^{3/2}$

Let's plot our model to visually assess the quality of fit:

```r
# model stored as function
myfun <- function(x) 0.01320756 * x^(3/2)
g <- ggplot(mydf, aes(x = period, y = distance)) + geom_point() + theme_economist()
g <- g + labs(title = "Plot of Distance from sun (m) vs. Planet Period (s)")
g <- g + labs(subtitle = TeX("Model: $y=0.01321x^{3/2}$"))
g <- g + stat_function(fun = myfun)
g
```



The model does not fit tightly to the observed data.

As an alternative solution, we can fit another power function to the data, but leverage the least-squares model to solve for both the unknown coefficient a and the power n in the equation $y = ax^n$.

First, take the natural log of both sides for the original equation:

$$ln\ y = ln\ a\ +\ n\ ln\ x$$

Now use the OLS linear model to solve for lna and n.

```r
# log transformations
xlog <- log(mydf$period)
ylog <- log(mydf$distance)

# OLS linear model
mylm <- lm(ylog ~ xlog)
```

```r
# model parameters ln(a) and n
mylm$coefficients
```

```
(Intercept)        xlog
 14.2191681   0.6667012
```

Solve for a as follows:

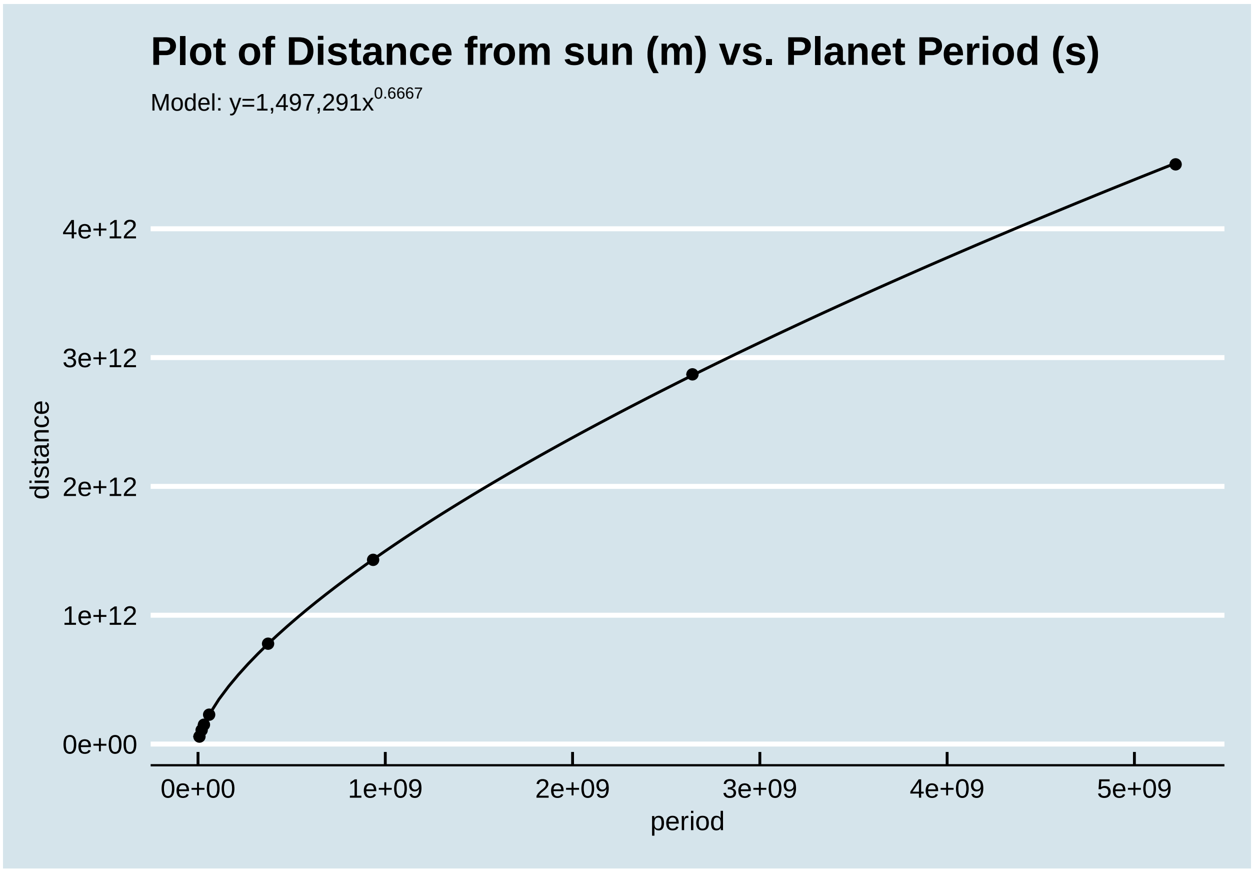
```r
# solve for a
a <- exp(mylm$coefficients[1])
a
```

```
(Intercept)
    1497291
```

Our fitted model is:

$$y = 1497291x^{0.6667}$$

```r
# model stored as function
myfun <- function(x) 1497291 * x^(0.6667012)
g <- ggplot(mydf, aes(x = period, y = distance)) + geom_point() + theme_economist()
g <- g + labs(title = "Plot of Distance from sun (m) vs. Planet Period (s)")
g <- g + labs(subtitle = TeX("Model: $y=1,497,291x^{0.6667}$"))
g <- g + stat_function(fun = myfun)
g
```



The new model appears to fit very well to the observed data.