Week 3 HW 609

Charley Ferrari

Friday, September 11, 2015

Page 127, Number 10

```
planets <- data.frame(body = c("Mercury", "Venus", "Earth", "Mars", "Jupiter",
                               "Saturn", "Uranus", "Neptune"),
                      period = c(7.6e6, 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))
```

we're looking for a least squares solution to the formula $y = Ax^n$

In this case, our formula is:

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

Where $n = \frac{3}{2}$

```
a <- sum(planets$period^(3/2)*planets$distance)/sum(planets$period^(6/2))

planets$distancest <- a*planets$period^(3/2)

a
```

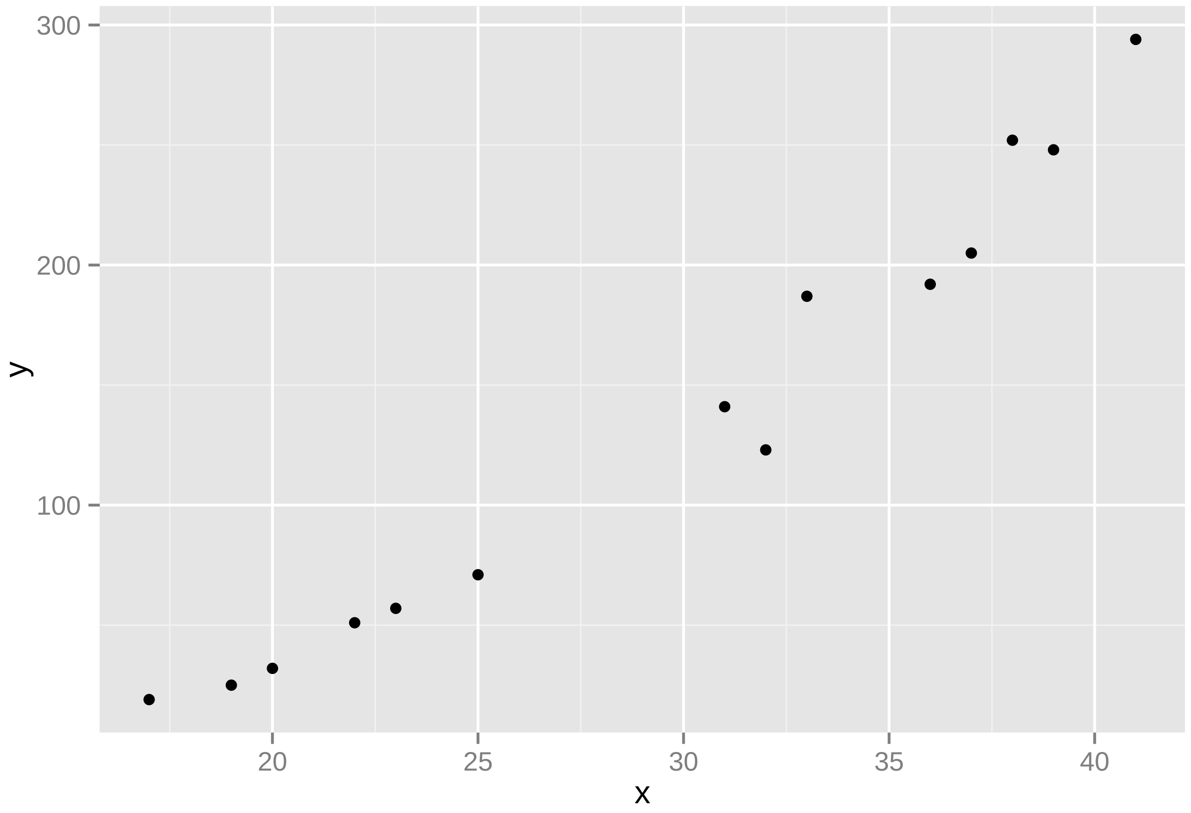
```
## [1] 0.01320756
```

So, our equation becomes $distance = 0.01320756 * x^{\frac{3}{2}}$

page 157, question 4

In the following data, X represents the diameter of a Ponderosa Pine measured at breast height, and Y is a measure of volume - number of bard feet divided by 10. Make a scatterplot of the data.

```
ponderosa <- data.frame(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))

library(ggplot2)

ggplot(ponderosa, aes(x=x, y=y)) + geom_point()
```



Discuss the appropriateness of using a 13th degree polynomial that passes through the data points as an empirical model:

Since we have 14 datapoints, a 13th degree polynomial exists that would pass through every data point. However, this would be an example of overfitting: it would fit the data perfectly, but woudln't be applicable to new observations.

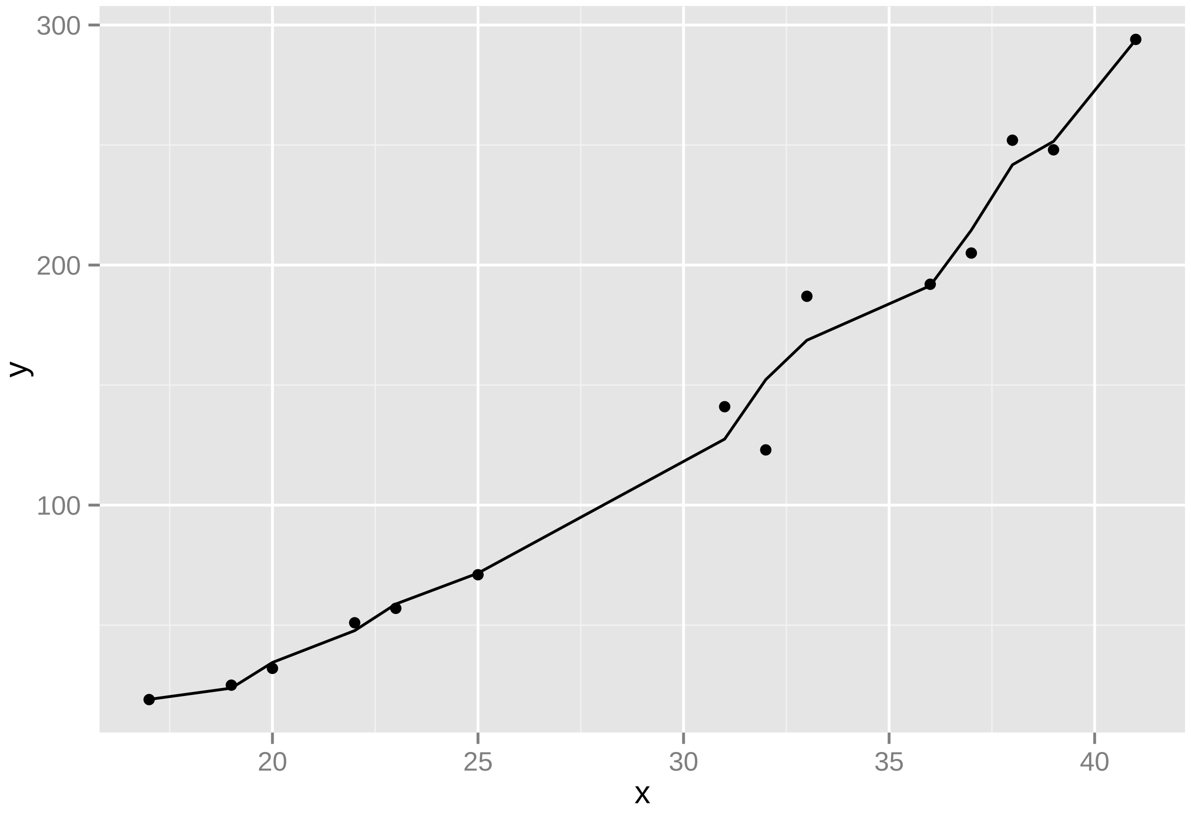
```r
summary(lm(y ~ poly(x, 13, raw=TRUE), data=ponderosa))
```

```
##
## Call:
## lm(formula = y ~ poly(x, 13, raw = TRUE), data = ponderosa)
##
## Residuals:
##        1        2        3        4        5        6        7        8
##  -0.0777   1.2350  -2.4560   3.2869  -1.7901  -0.6550  13.4945 -29.2786
##        9       10       11       12       13       14
##  18.3321   0.6008  -9.5549  10.2363  -3.5314   0.1583
##
## Coefficients: (3 not defined because of singularities)
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 1.670e+07  2.823e+07   0.592    0.596
## poly(x, 13, raw = TRUE)1  -6.202e+06  1.031e+07  -0.602    0.590
## poly(x, 13, raw = TRUE)2   1.018e+06  1.666e+06   0.611    0.584
## poly(x, 13, raw = TRUE)3  -9.699e+04  1.563e+05  -0.621    0.579
## poly(x, 13, raw = TRUE)4   5.906e+03  9.383e+03   0.629    0.574
## poly(x, 13, raw = TRUE)5  -2.380e+02  3.732e+02  -0.638    0.569
## poly(x, 13, raw = TRUE)6   6.309e+00  9.774e+00   0.646    0.565
## poly(x, 13, raw = TRUE)7  -1.038e-01  1.590e-01  -0.653    0.561
## poly(x, 13, raw = TRUE)8   8.663e-04  1.315e-03   0.659    0.557
## poly(x, 13, raw = TRUE)9          NA         NA      NA       NA
## poly(x, 13, raw = TRUE)10 -4.741e-08  7.078e-08  -0.670    0.551
## poly(x, 13, raw = TRUE)11         NA         NA      NA       NA
## poly(x, 13, raw = TRUE)12  2.555e-12  3.768e-12   0.678    0.546
## poly(x, 13, raw = TRUE)13         NA         NA      NA       NA
##
## Residual standard error: 23.14 on 3 degrees of freedom
## Multiple R-squared:  0.9862, Adjusted R-squared:  0.9401
## F-statistic:  21.4 on 10 and 3 DF,  p-value: 0.01419
```

```
coefficients <- as.vector(lm(y ~ poly(x, 13, raw=TRUE), data=ponderosa)$coefficients)

coefficients # some are NA, so I'll remove those terms
```

```
##  [1]  1.670122e+07 -6.202161e+06  1.018156e+06 -9.698887e+04  5.906309e+03
##  [6] -2.380261e+02  6.309286e+00 -1.037739e-01  8.662536e-04            NA
## [11] -4.740887e-08            NA  2.554665e-12            NA
```

```
ggplot(ponderosa) + geom_point(aes(x=x, y=y)) + geom_line(aes(x=x, y=
  coefficients[1] + coefficients[2]*x + coefficients[3]*x^2 +
    coefficients[4]*x^3 + coefficients[5]*x^4 + coefficients[6]*x^5 +
    coefficients[7]*x^6 + coefficients[8]*x^7 + coefficients[9]*x^8 +
    coefficients[11]*x^10 +
    coefficients[13]*x^12))
```



While not quite a lagrangian polynomial (which would actually be passing through every one of the points,) one can see that the curve is influenced by the data points laying a bit further from the straight line fit in order to minimize the residuals. I'm not sure why the coefficients are NAs for the 9th, 11th, and 13th powers of X. perhaps if those were calculated, I'd have the lagrangian I would have expected.

Either way, having this many variables is definitely overfitting the data, and probably isn't best to use as a model for Ponderosa Pines.

page 169 question 11

The following data represent the length of a bass fish and its weight:

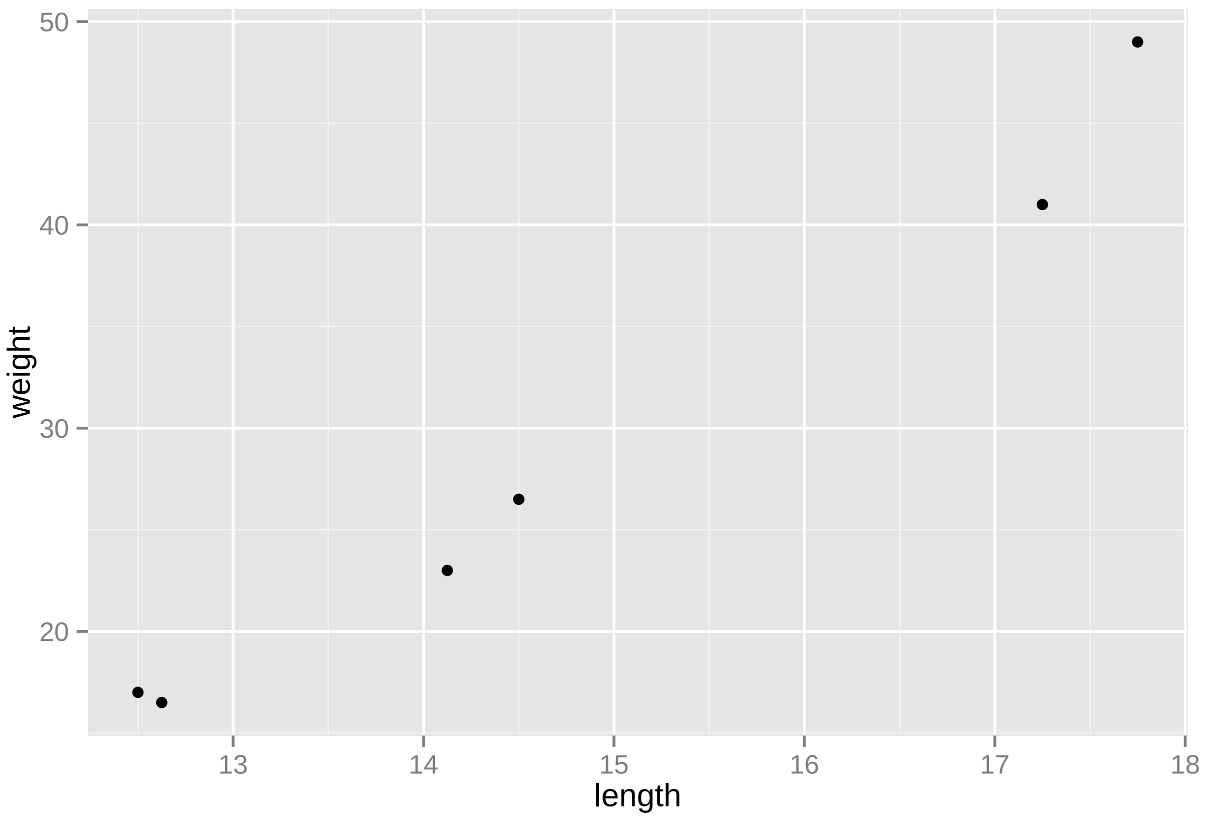
```
bass <- data.frame(length = c(12.5, 12.625, 14.125, 14.5, 17.25, 17.75),
                   weight = c(17, 16.5, 23, 26.5, 41, 49))

bassdelta1 <- (tail(bass$weight, -1) - head(bass$weight, -1))/
  (tail(bass$length, -1) - head(bass$length, -1))

bassdelta2 <- (tail(bassdelta1, -1) - head(bassdelta1, -1))/
  (tail(bass$length, -2) - head(bass$length, -2))

bassdelta3 <- (tail(bassdelta2, -1) - head(bassdelta2, -1))/
  (tail(bass$length, -3) - head(bass$length, -3))

bassdelta4 <- (tail(bassdelta3, -1) - head(bassdelta3, -1))/
  (tail(bass$length, -4) - head(bass$length, -4))
```

```
bassdelta5 <- (tail(bassdelta4, -1) - head(bassdelta4, -1))/
  (tail(bass$length, -5) - head(bass$length, -5))


bass$delta1 <- c(bassdelta1, 0)
bass$delta2 <- c(bassdelta2, rep(0,2))
bass$delta3 <- c(bassdelta3, rep(0,3))
bass$delta4 <- c(bassdelta4, rep(0,4))
bass$delta5 <- c(bassdelta5, rep(0,5))

bass
```

```
##   length weight    delta1    delta2     delta3     delta4     delta5
## 1 12.500   17.0 -4.000000  5.128205 -1.2307692 0.07857739 0.06406722
## 2 12.625   16.5  4.333333  2.666667 -0.8575266 0.41493031 0.00000000
## 3 14.125   23.0  9.333333 -1.299394  1.2689912 0.00000000 0.00000000
## 4 14.500   26.5  5.272727  3.300699  0.0000000 0.00000000 0.00000000
## 5 17.250   41.0 16.000000  0.000000  0.0000000 0.00000000 0.00000000
## 6 17.750   49.0  0.000000  0.000000  0.0000000 0.00000000 0.00000000
```

First lets plot the data:

```
ggplot(bass, aes(x=length, y=weight)) + geom_point()
```



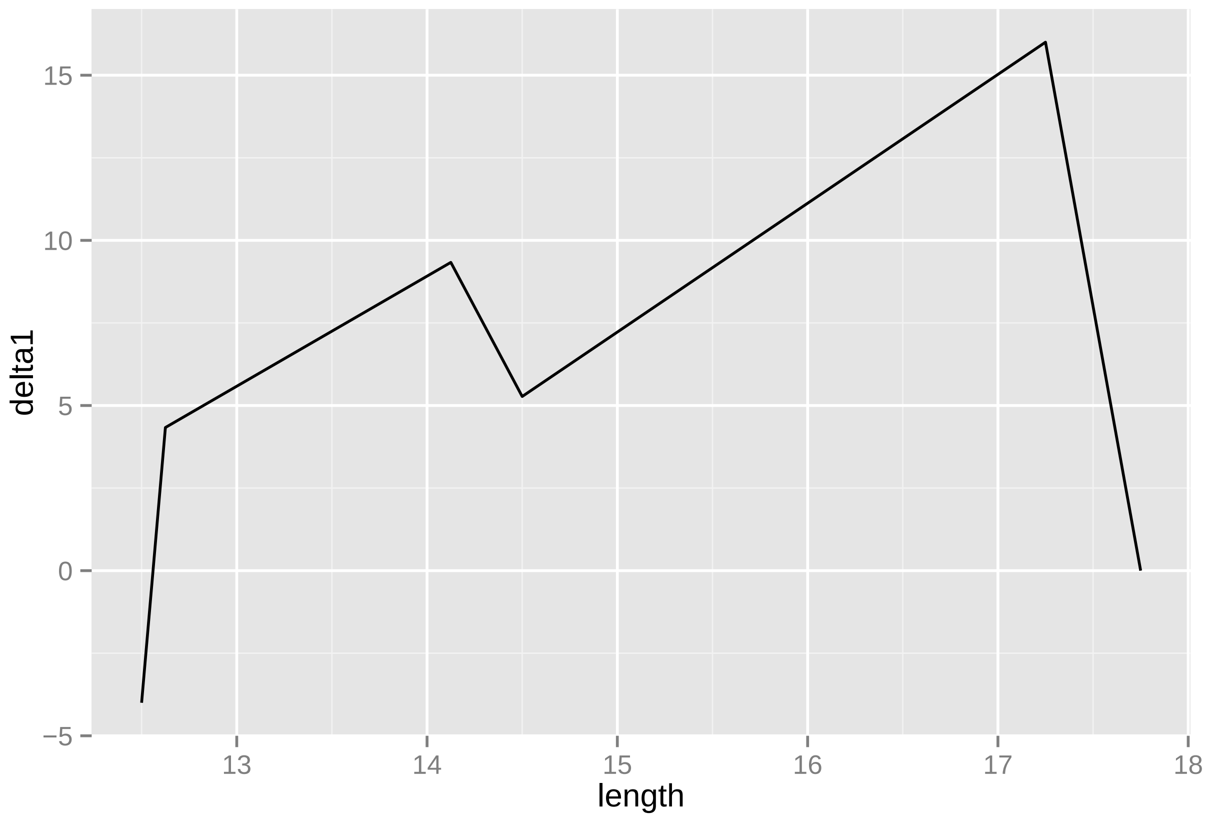
The data appears to be linear. Lets look at the difference table:

```
bass
```

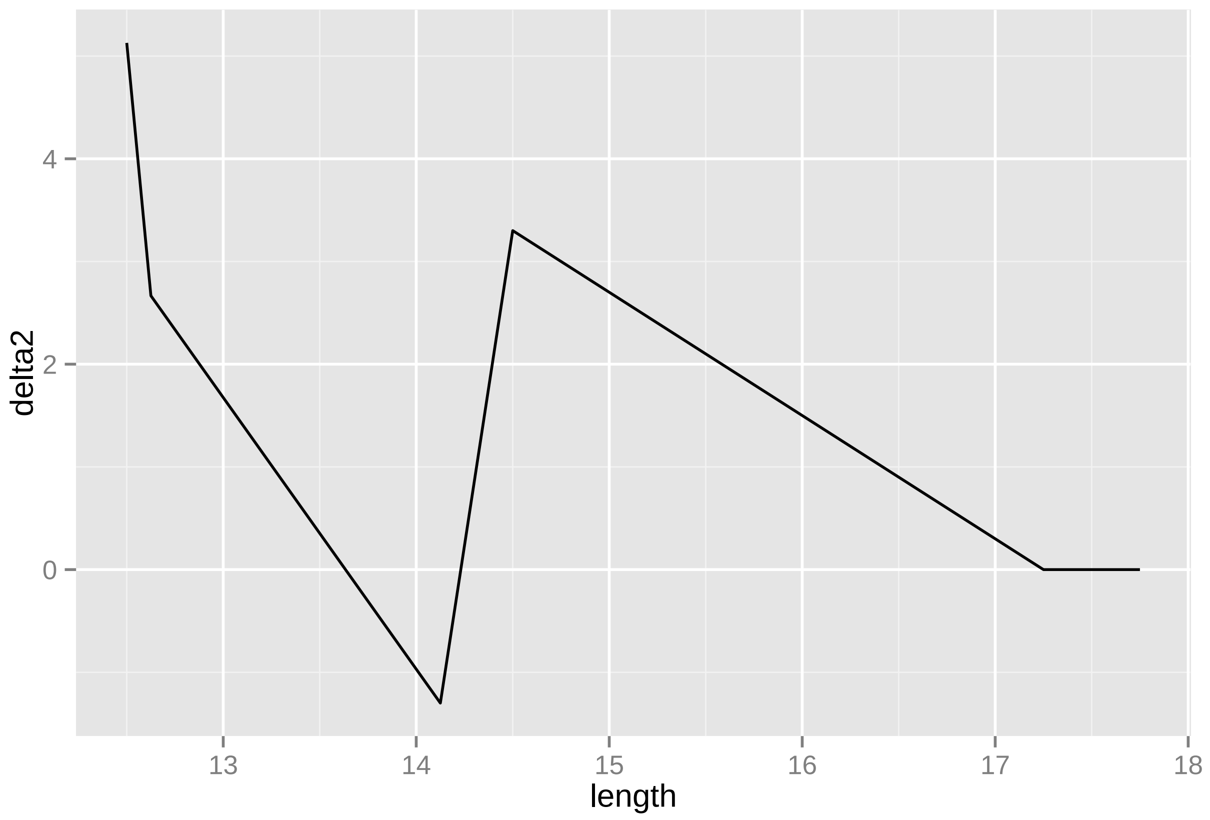
```
##   length weight    delta1    delta2     delta3     delta4     delta5
## 1 12.500   17.0 -4.000000  5.128205 -1.2307692 0.07857739 0.06406722
## 2 12.625   16.5  4.333333  2.666667 -0.8575266 0.41493031 0.00000000
## 3 14.125   23.0  9.333333 -1.299394  1.2689912 0.00000000 0.00000000
## 4 14.500   26.5  5.272727  3.300699  0.0000000 0.00000000 0.00000000
## 5 17.250   41.0 16.000000  0.000000  0.0000000 0.00000000 0.00000000
## 6 17.750   49.0  0.000000  0.000000  0.0000000 0.00000000 0.00000000
```

In order to get a handle on what the difference tables are doing, lets take a look at line graphs of each of the deltas:

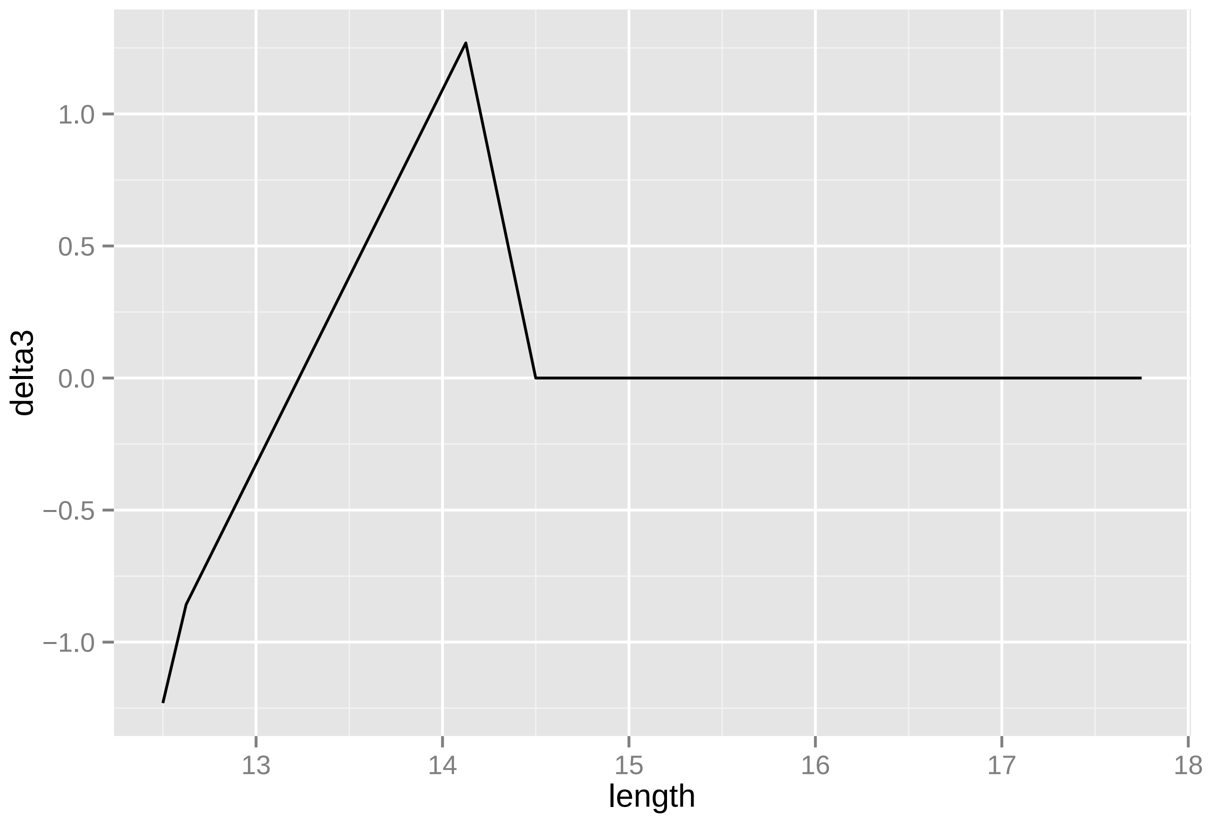
```
ggplot(bass, aes(x=length, y=delta1)) + geom_line()
```



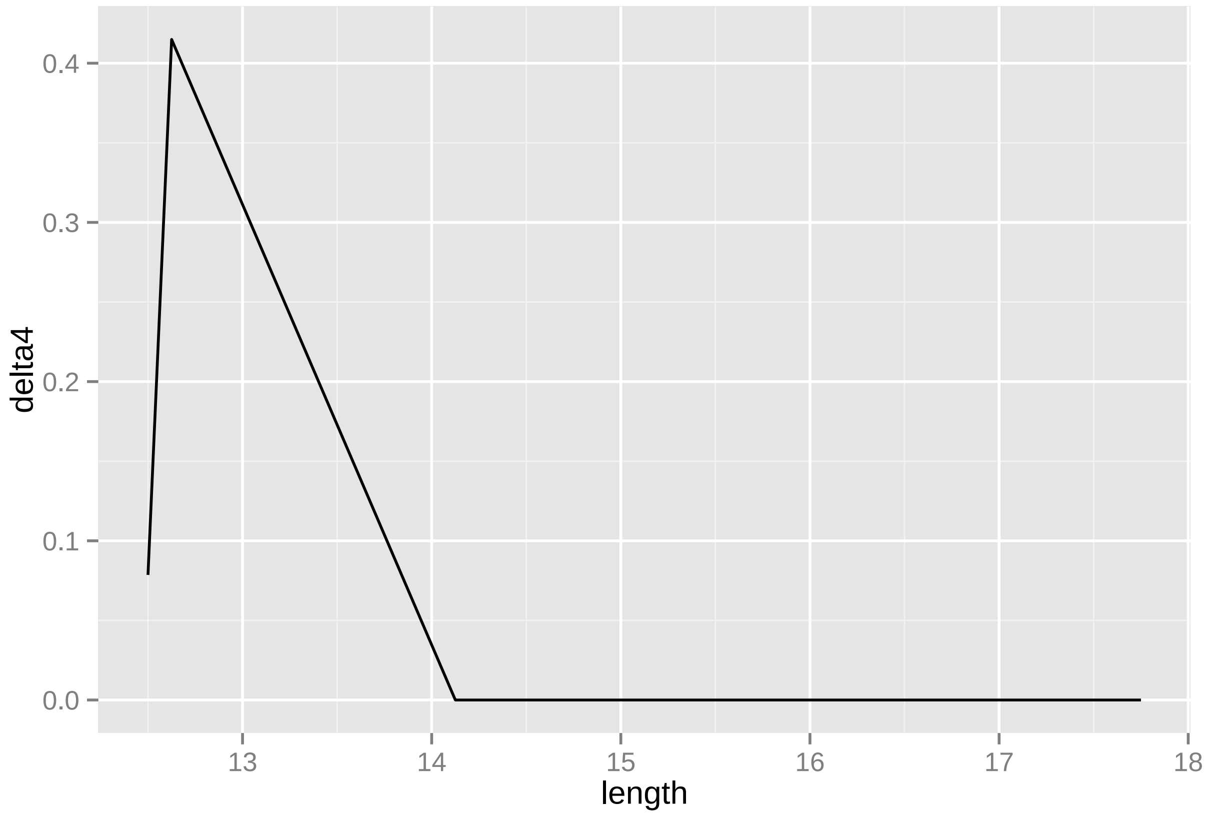
```
ggplot(bass, aes(x=length, y=delta2)) + geom_line()
```



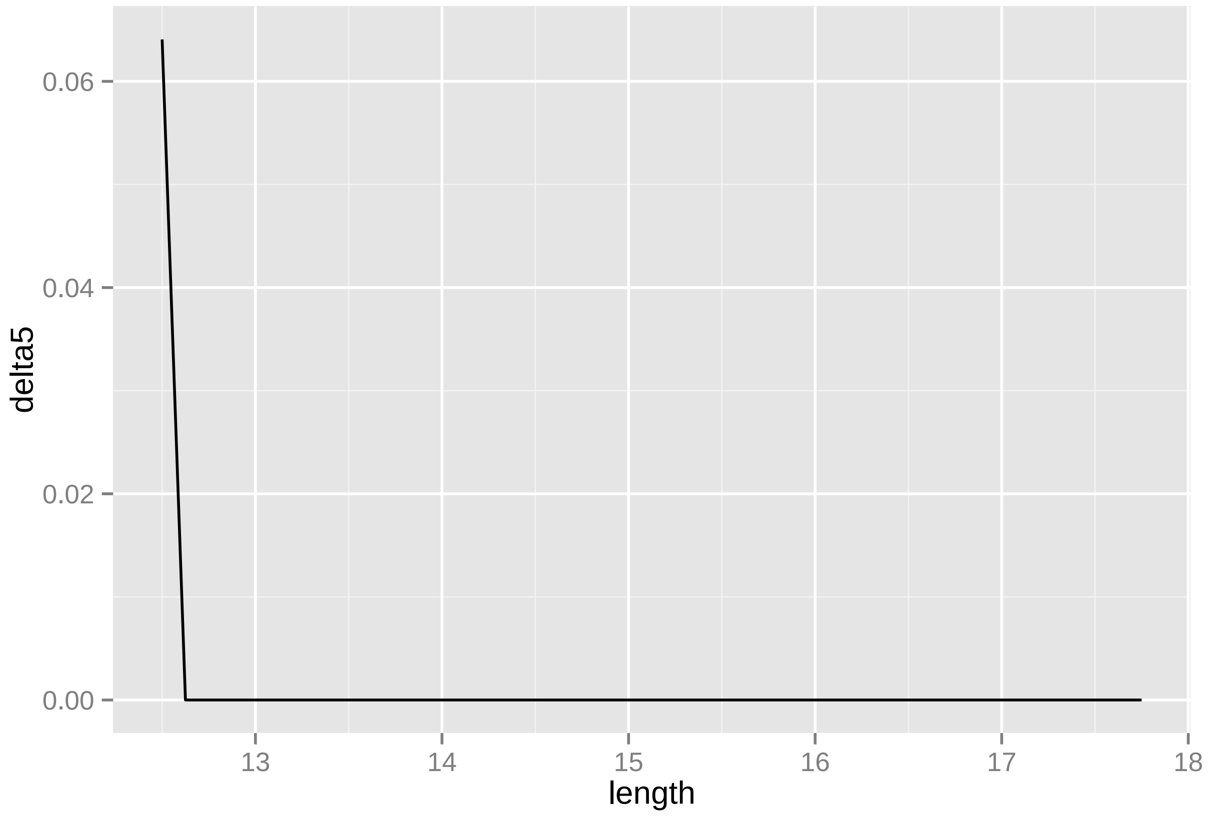
```r
ggplot(bass, aes(x=length, y=delta3)) + geom_line()
```



```r
ggplot(bass, aes(x=length, y=delta4)) + geom_line()
```



```
ggplot(bass, aes(x=length, y=delta5)) + geom_line()
```

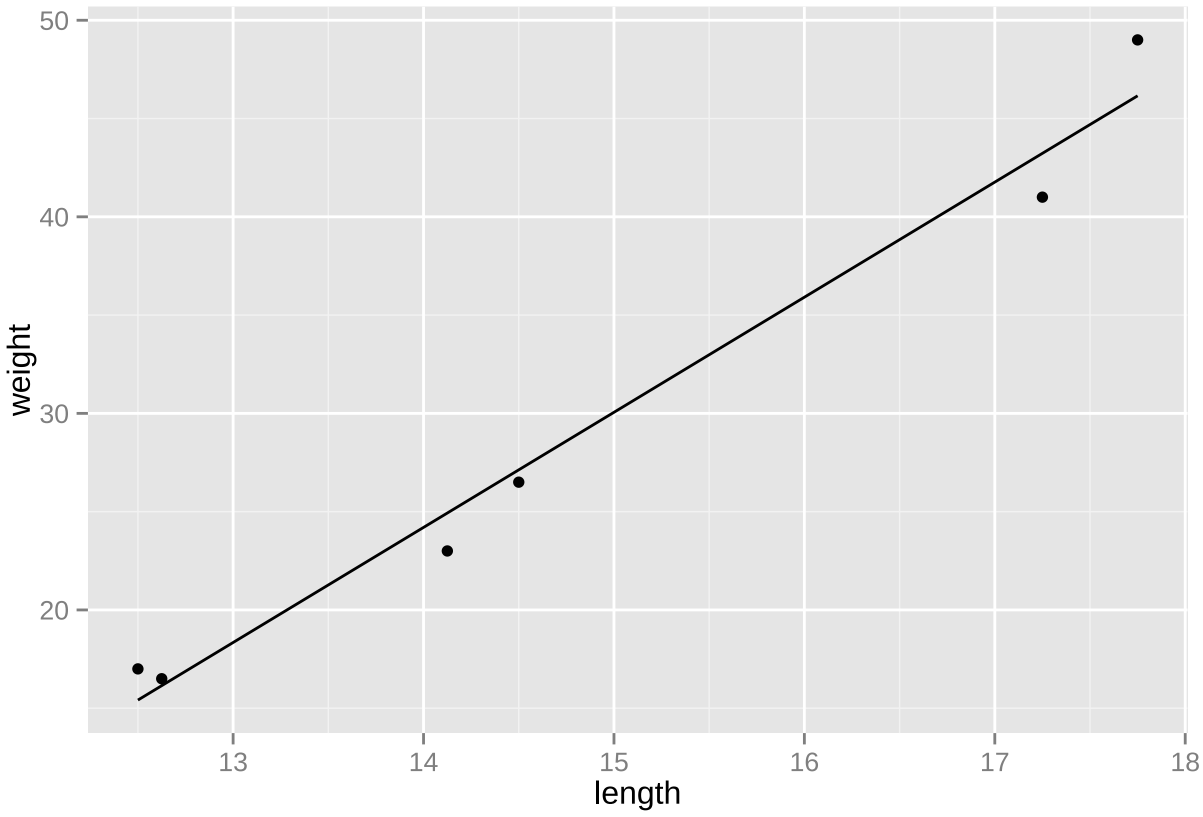


The change in signs could be errors propagating through the data, but what stuck out to me more is the shapes of each of the graph. It seems like the first derivative is a cubic, the second derivative is quadratic, and the third derivative is linear.

The original scatterplot, however, heavily suggests a linear relationship. The first datapoint, where the weight is higher even though the length is lower than the second datapoint, might be the error in the data that is propagating throughout and giving me spurious results.

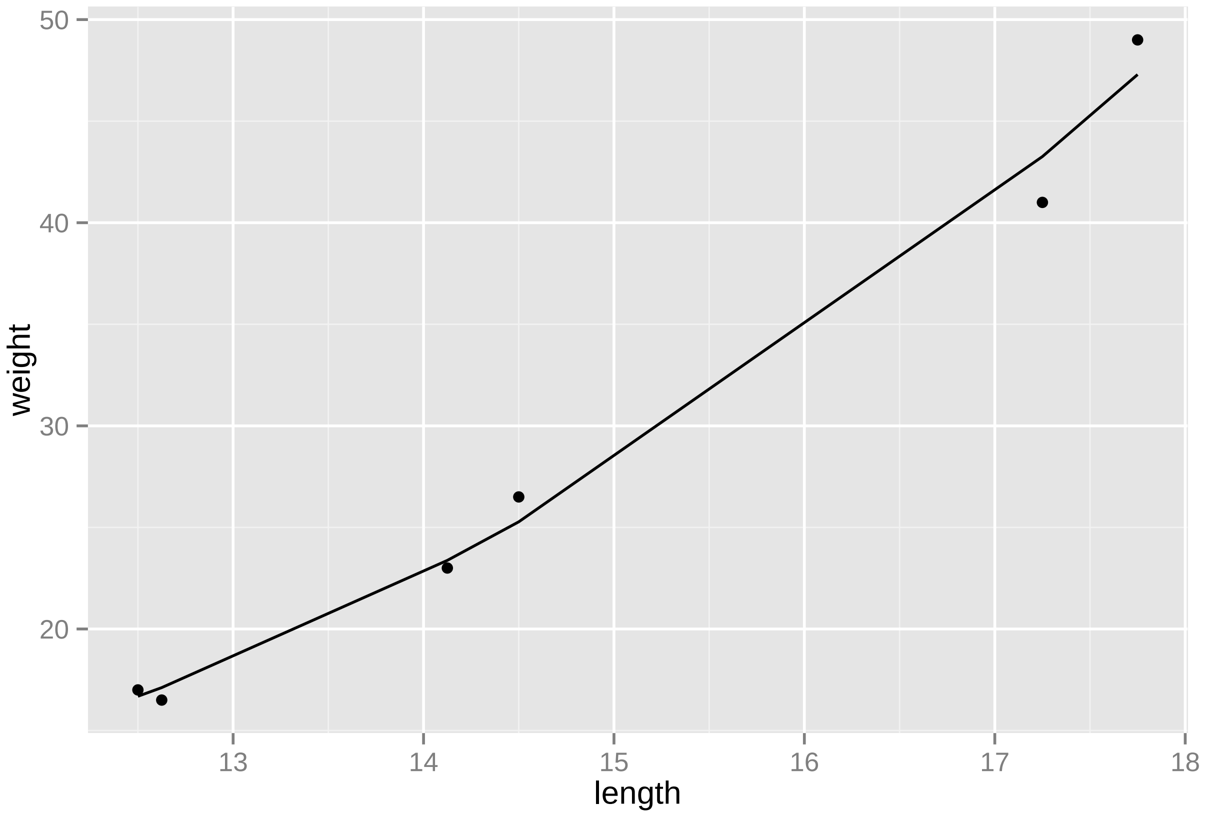
Lets create a linear model:

```
bassmodel <- lm(weight ~ length, data=bass)

coefficients <- bassmodel$coefficients

bass$bassest <- coefficients[1] + coefficients[2]*bass$length

ggplot(bass) + geom_point(aes(x=length, y=weight)) + geom_line(aes(x=length, y=bassest))
```



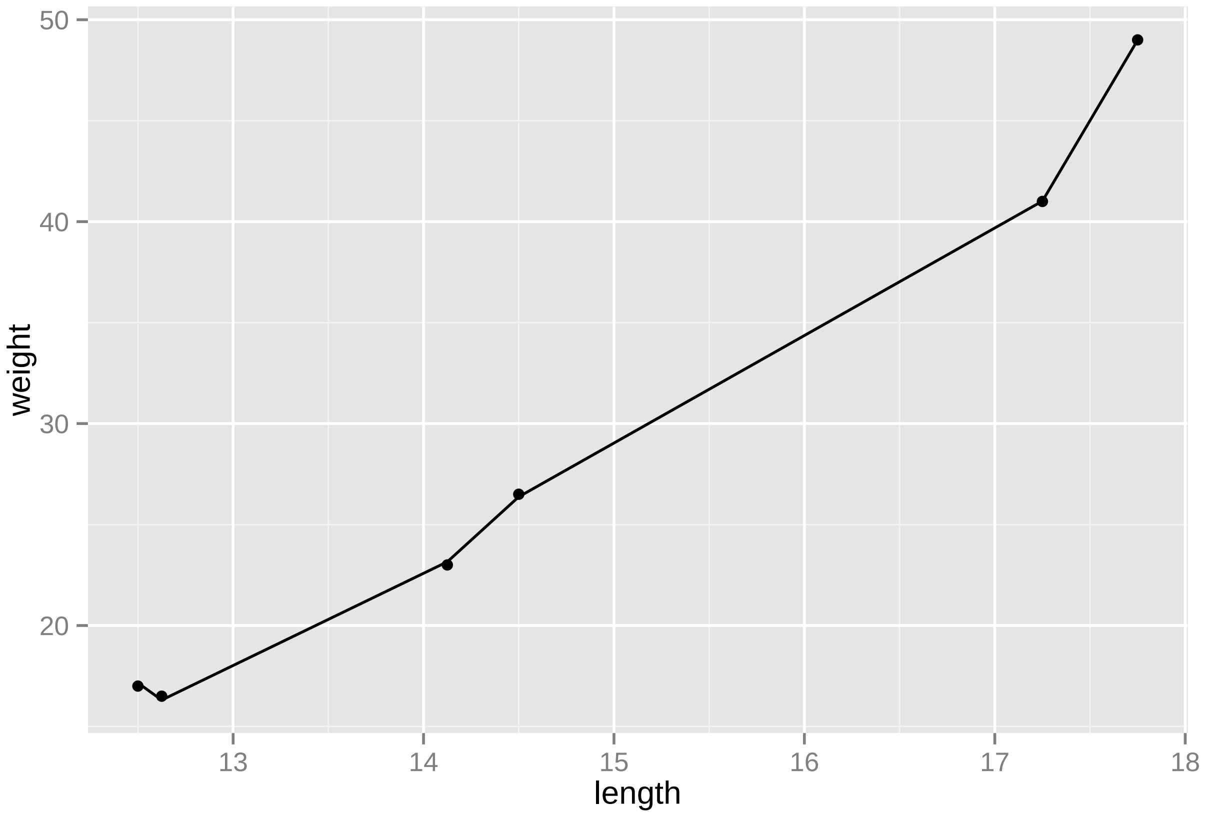
Looking at the plot, I'm now rethinking my choice in model. I at least see the potential for a quadratic estimation. Lets try that as our model

```
bassmodelquad <- lm(weight ~ poly(length, 2, raw=TRUE), data=bass)

coefficients <- bassmodelquad$coefficients

bass$bassestquad <- coefficients[1] + coefficients[2]*bass$length + coefficients[3]*bass$length^2

ggplot(bass) + geom_point(aes(x=length, y=weight)) + geom_line(aes(x=length, y=bassestquad))
```



That looks a bit more satisfying. Lets try the quartic equation though just to see what it looks like

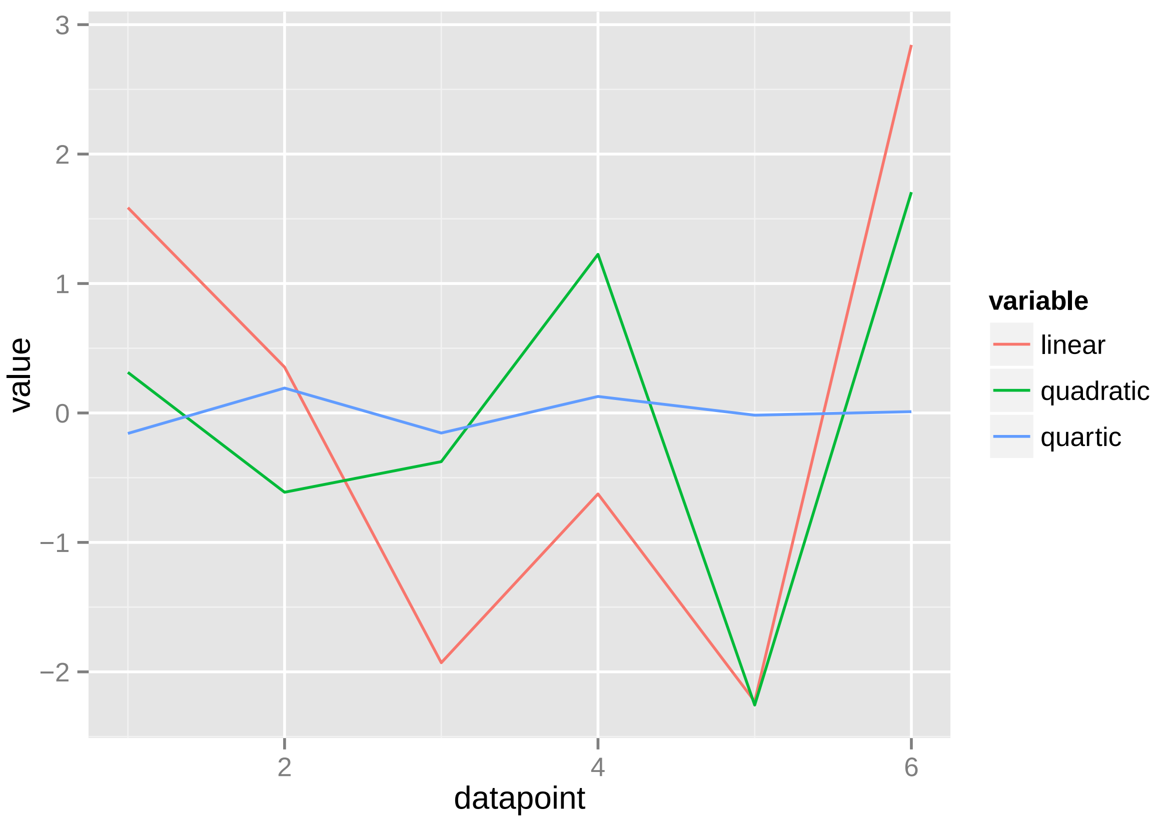
```
bassmodelquart <- lm(weight ~ poly(length, 4, raw=TRUE), data=bass)

coefficients <- bassmodelquart$coefficients

bass$bassestquart <- coefficients[1] + coefficients[2]*bass$length + coefficients[3]*bass$length^2 +
  coefficients[4]*bass$length^3 + coefficients[5]*bass$length^4

ggplot(bass) + geom_point(aes(x=length, y=weight)) + geom_line(aes(x=length, y=bassestquart))
```



Here, we're getting what we expected, the data looks overfit.

Lets chart the residuals

```r
residuals <- data.frame(linear = bassmodel$residuals, quadratic = bassmodelquad$residuals,
                        quartic = bassmodelquart$residuals, datapoint = seq(1,6))

library(reshape2)

ggplot(melt(residuals, id="datapoint"), aes(x=datapoint, y=value, color=variable)) + geom_line()
```



The quartic model has the least residuals by far, but I believe that it is overfit. I believe the quadratic model is a more generalizable fit.