Data 609 Assignment #2

Joseph Simone

2/19/2020

```
library(ggplot2)
library(tidyr)
library(dplyr)
library(knitr)
library(gridExtra)
library(latex2exp)
library(reshape2)
library(kableExtra)
```

Section 2.1

Page 121: #2

For each of the following data sets, formulate the mathematical model that minimizes the largest deviation between the data and the line y D axCb. If a computer is available, solve for the estimates of a and b.

```
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)
mydf <- data.frame(x, y)
```

```
mydf %>%
  kable() %>%
  kable_styling(full_width = F)
```

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

Part A.

```
lm1 <- lm(y ~ x)
lm1
```

solve the estimates for a and b using Least Square Method

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##      2.2149       0.5642
```

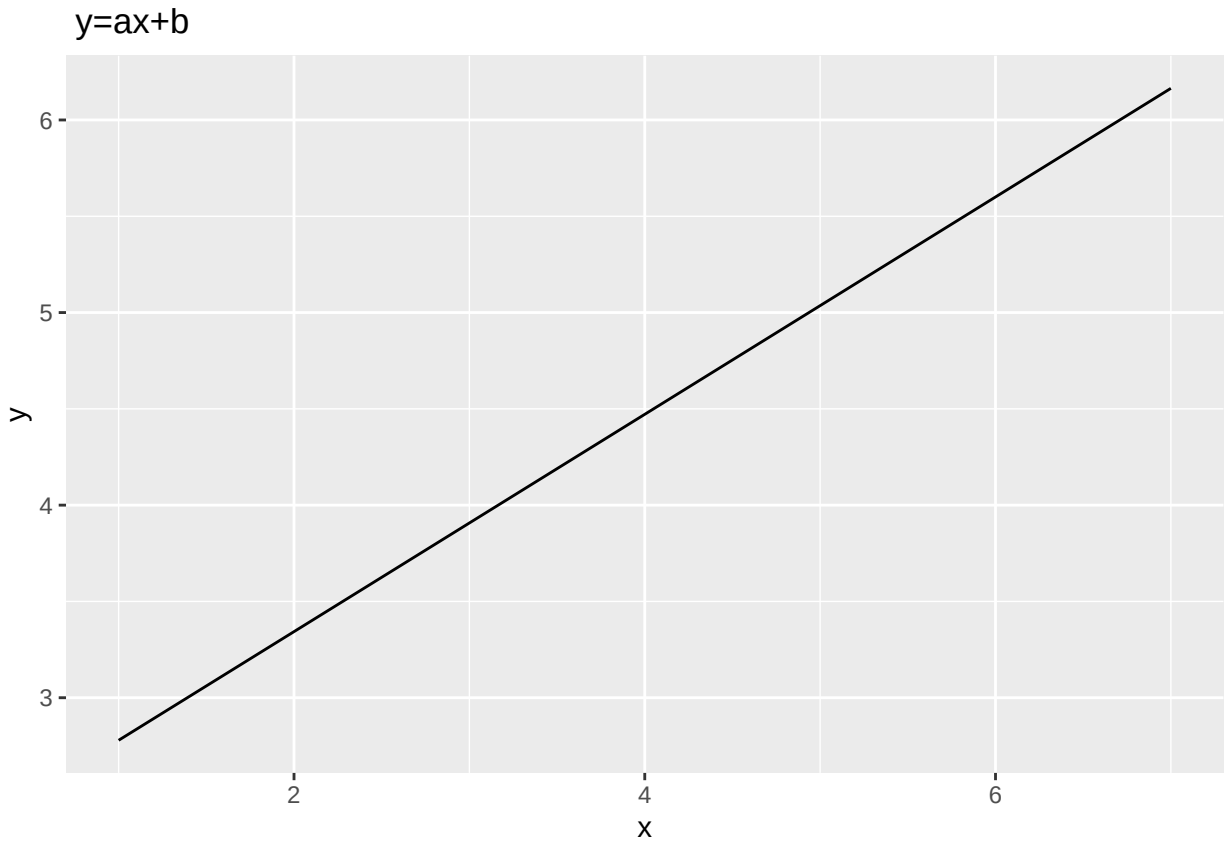
Slope : 0.5642

Intercept : 2.2148

```
mydf2<- data.frame(x,y)
mydf2$fx<- (0.5642*mydf2$x) + 2.2148
```

```
p<- ggplot(data = mydf2, aes(x = x, y = fx)) +
    geom_line()
```

```
p <- p + labs(title=" y=ax+b", x="x", y="y")
p
```



Plot

Page 135: #7

Part A. In the following data, W represents the weight of a fish (bass) and l represents its length. Fit the model W D kl3 to the data using the least-squares criterion.

```
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)
mydf2 <- data.frame(l, w)
```

```
mydf2 %>%
  kable() %>%
  kable_styling(full_width = F)
```

l	w
14.500	27
12.500	17
17.250	41
14.500	26
12.625	17
17.750	49
14.125	23
12.625	16

```
k <- sum(l^3 * w)/sum(l^6)
model1<- k*l^3
resi1 <- w - model1
```

```
sse1 <- sum(resi1^2)
sse1
```

```
## [1] 12.16834
```

Part B. In the following data, g represents the girth of a fish. Fit the model W D klg2 to the data using the least-squares criterion.

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

```
mydf3 <- data.frame(mydf2, g)
```

```
mydf3 %>%
  kable() %>%
  kable_styling(full_width = F)
```

l	w	g
14.500	27	9.750
12.500	17	8.375
17.250	41	11.000
14.500	26	9.750
12.625	17	8.500
17.750	49	12.500
14.125	23	9.000
12.625	16	8.500

$w = kl^3$ $k = \frac{\Sigma l_i^3 w_i}{\Sigma l_i^6}$

```
k2 <- sum(l*g^2*w)/sum(l^2*g^4)
model2 <- k2*l*g^2
resi2 <- w - model2
```

```
sse2 <- sum(resi2^2)
sse2
```

```
## [1] 17.6711
```

Part C. Which of the two models fits the data better? Justify fully. Which model do you prefer? Why?

Solution: The model that fits the data better, would have to be the first Model. This is due to the fact that the Sum of Squares Error is lower for Model 1.

Page 135: #7

construct a scatterplot of the given data. Is there a trend in the data? Are any of the data points outliers? Construct a divided difference table. Is smoothing with a low-order polynomial appropriate? If so, choose an appropriate polynomial and fit using the least-squares criterion of best fit. Analyze the goodness of fit by examining appropriate indicators and graphing the model, the data points, and the deviations.

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))
```

```
bass %>%
  kable() %>%
  kable_styling(full_width = F)
```

length	weight
12.500	17.0
12.625	16.5
14.125	23.0
14.500	26.5
17.250	41.0
17.750	49.0

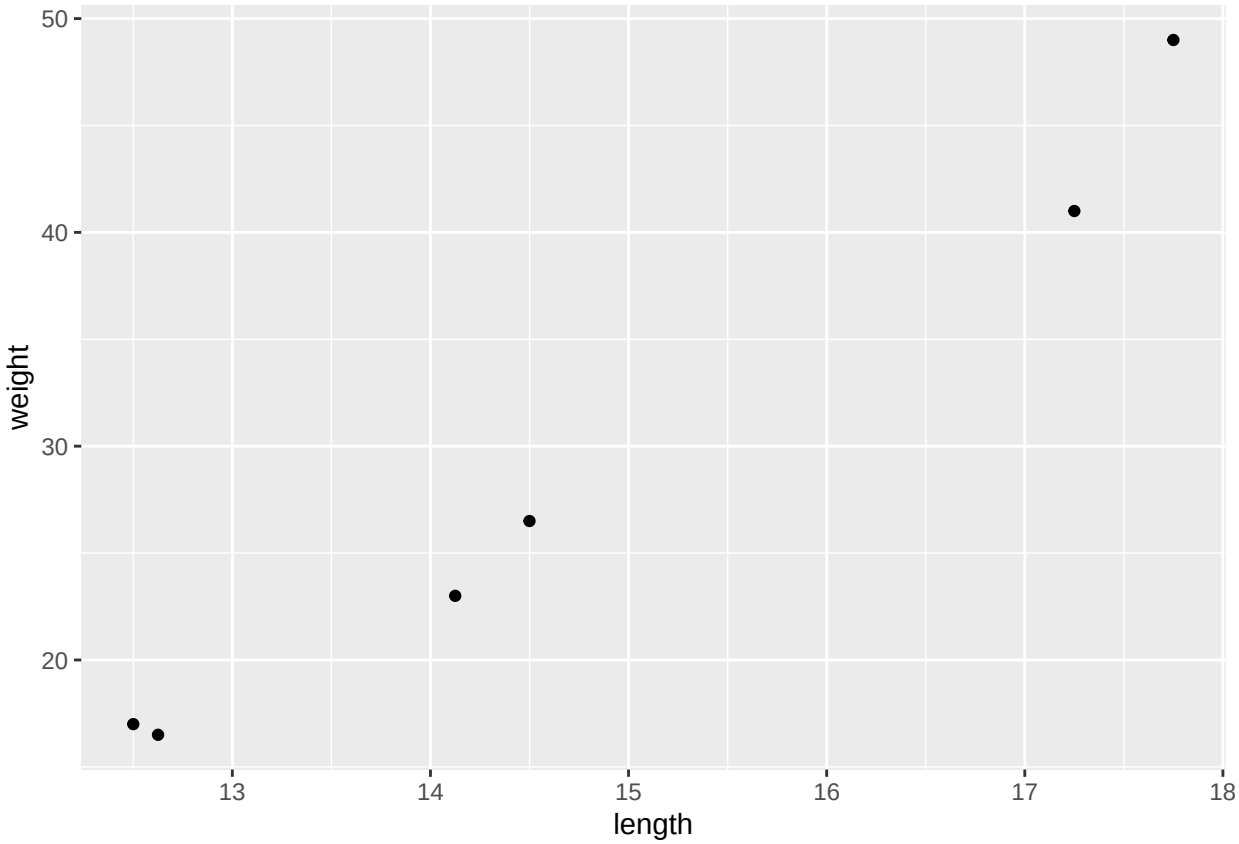
```
bassdelta1 <- (tail(bass$weight, -1) - head(bass$weight, -1))/
  (tail(bass$length, -1) - head(bass$length, -1))

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 %>%
  kable() %>%
  kable_styling(full_width = F)
```

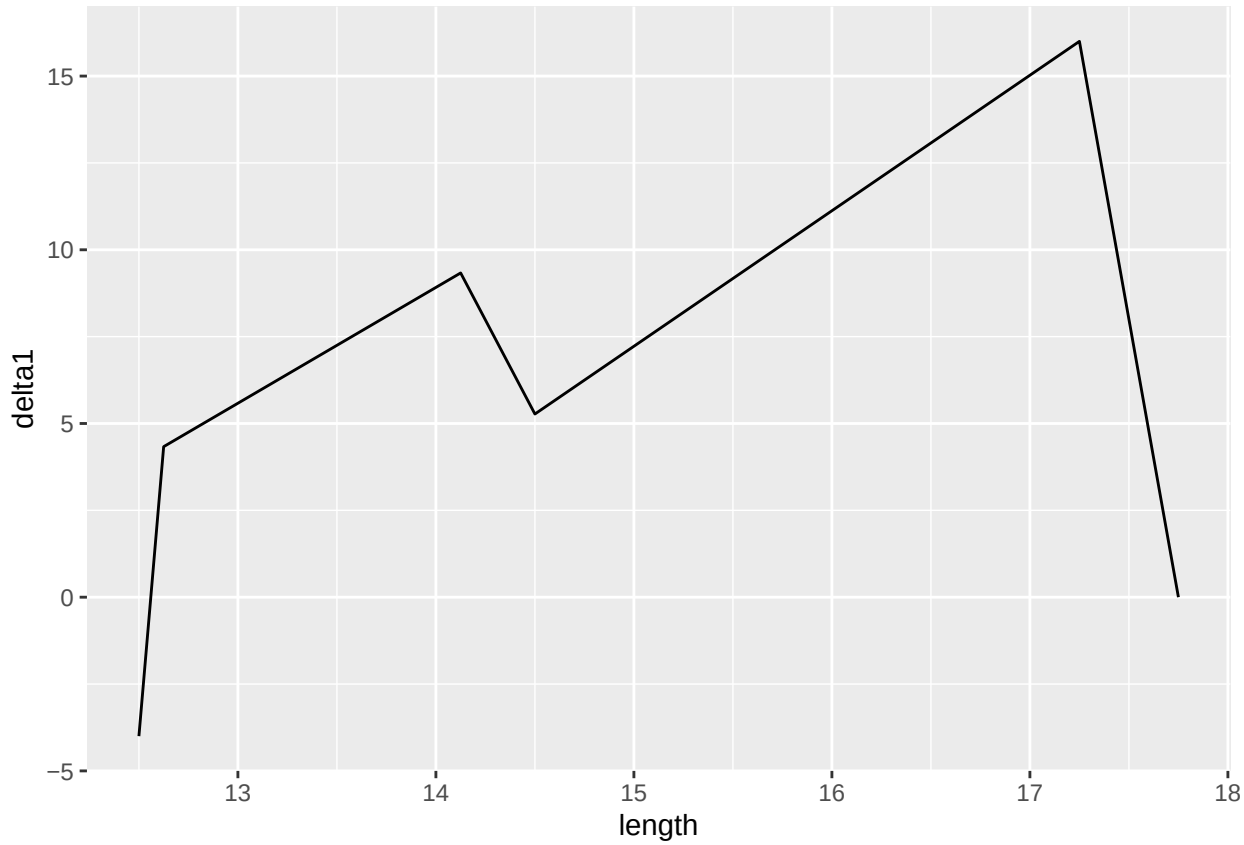
length	weight	delta1	delta2	delta3	delta4	delta5
12.500	17.0	-4.000000	5.128205	-1.2307692	0.0785774	0.0640672
12.625	16.5	4.333333	2.666667	-0.8575266	0.4149303	0.0000000
14.125	23.0	9.333333	-1.299394	1.2689912	0.0000000	0.0000000
14.500	26.5	5.272727	3.300699	0.0000000	0.0000000	0.0000000
17.250	41.0	16.000000	0.000000	0.0000000	0.0000000	0.0000000
17.750	49.0	0.000000	0.000000	0.0000000	0.0000000	0.0000000

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

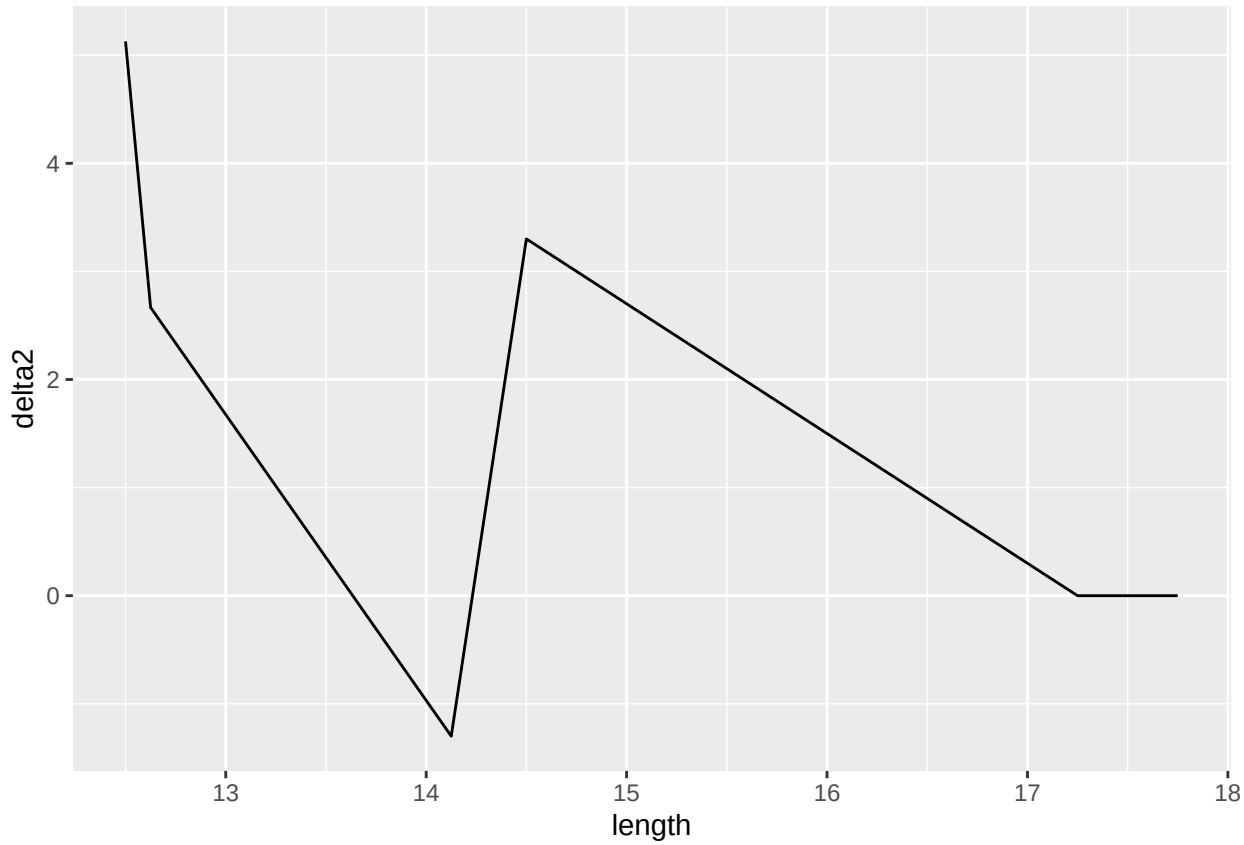


As seen from the plot the number of observations are less, so it is difficult to detect outliers in this scatter plot

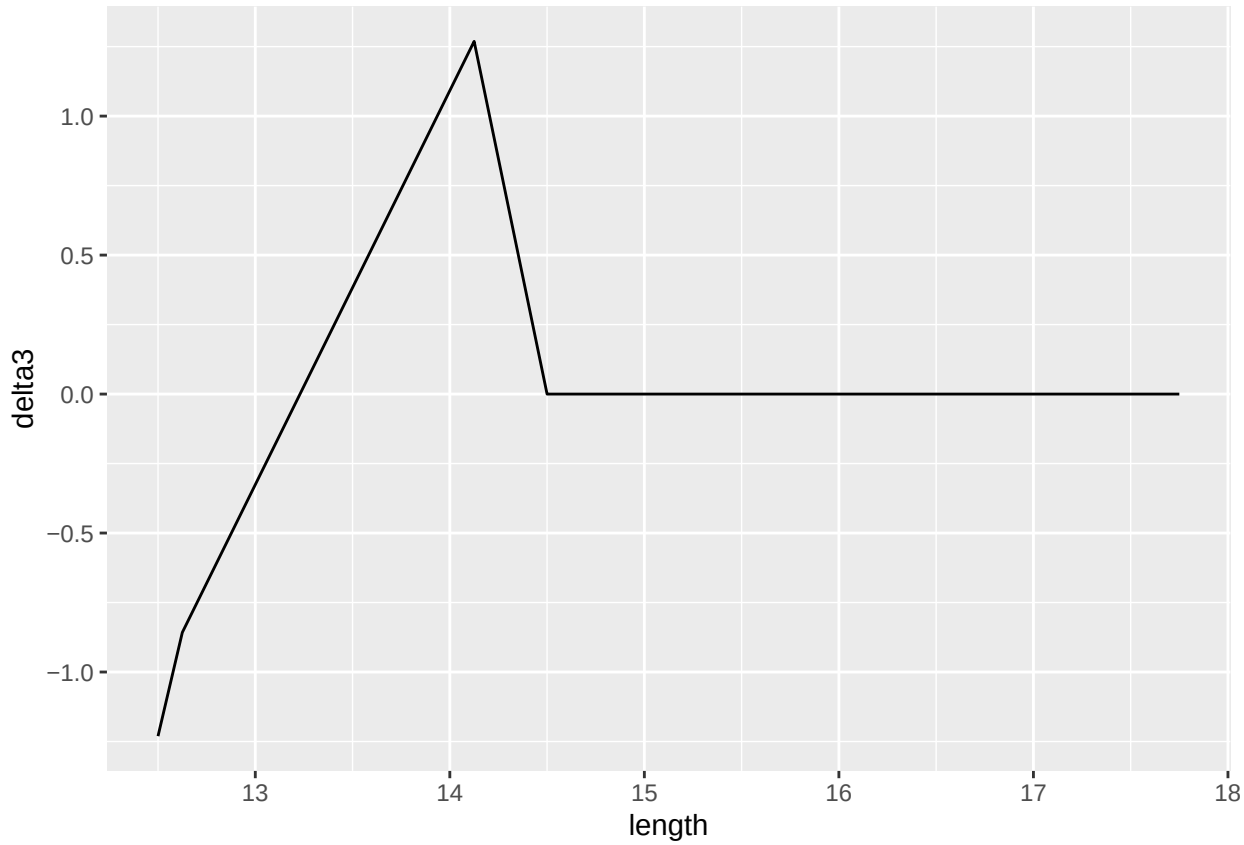
```
ggplot(bass, aes(x=length, y=delta1)) + geom_line()
```



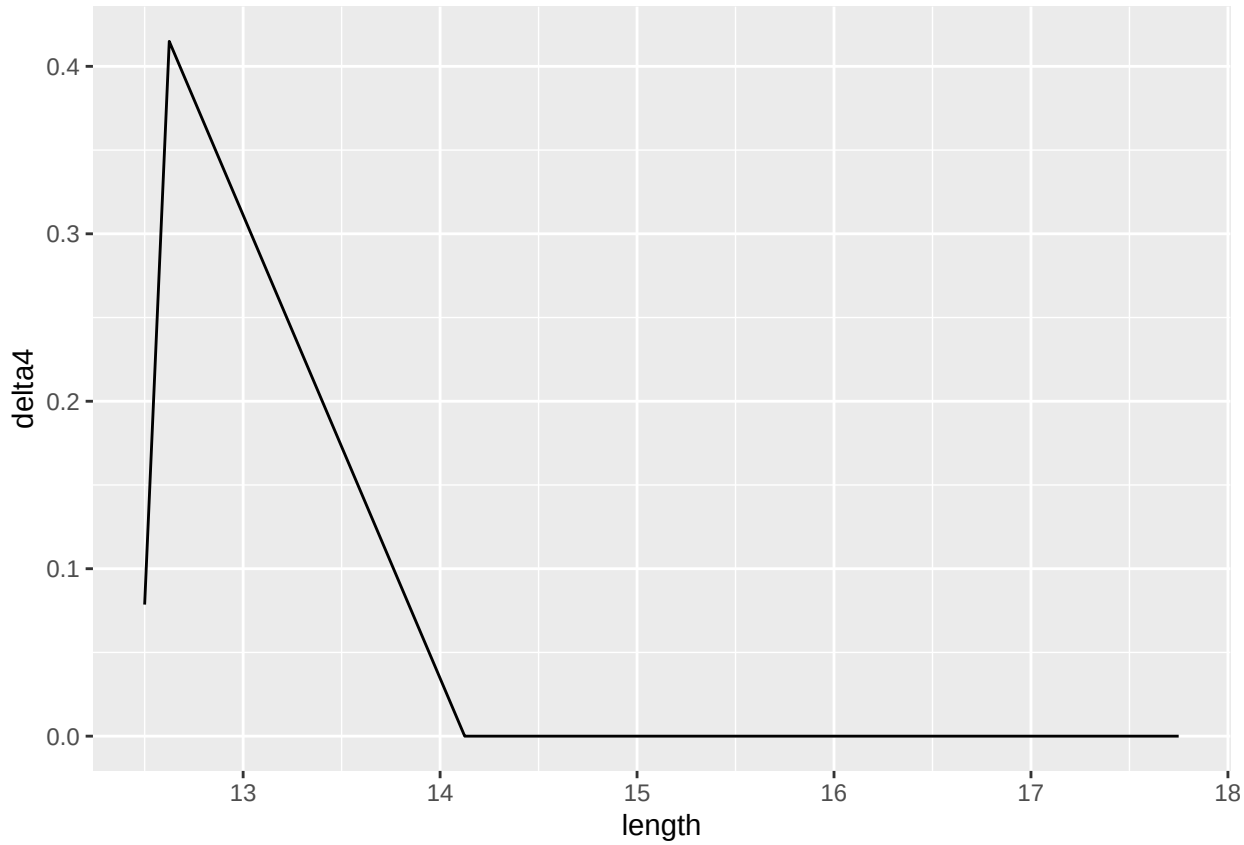

```
ggplot(bass, aes(x=length, y=delta2)) + geom_line()
```



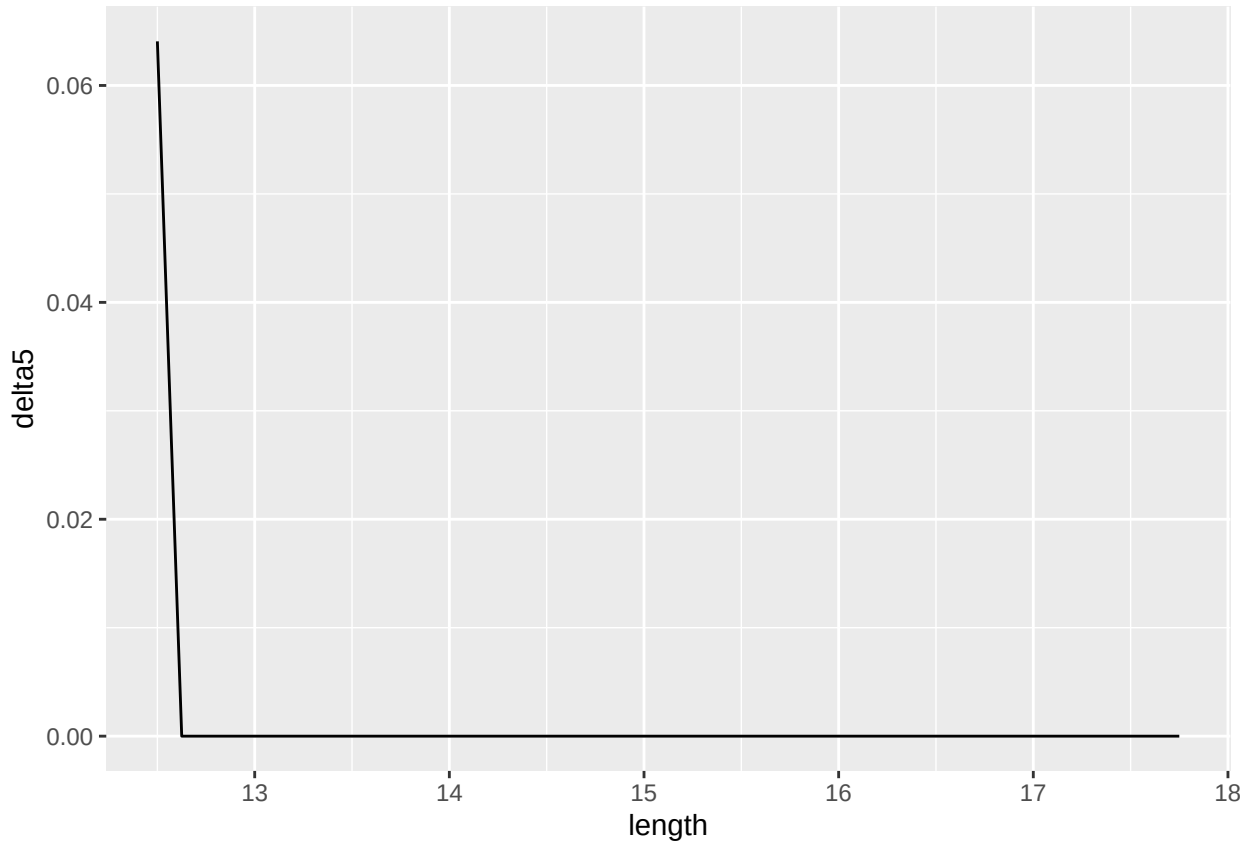

```
ggplot(bass, aes(x=length, y=delta3)) + geom_line()
```




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




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

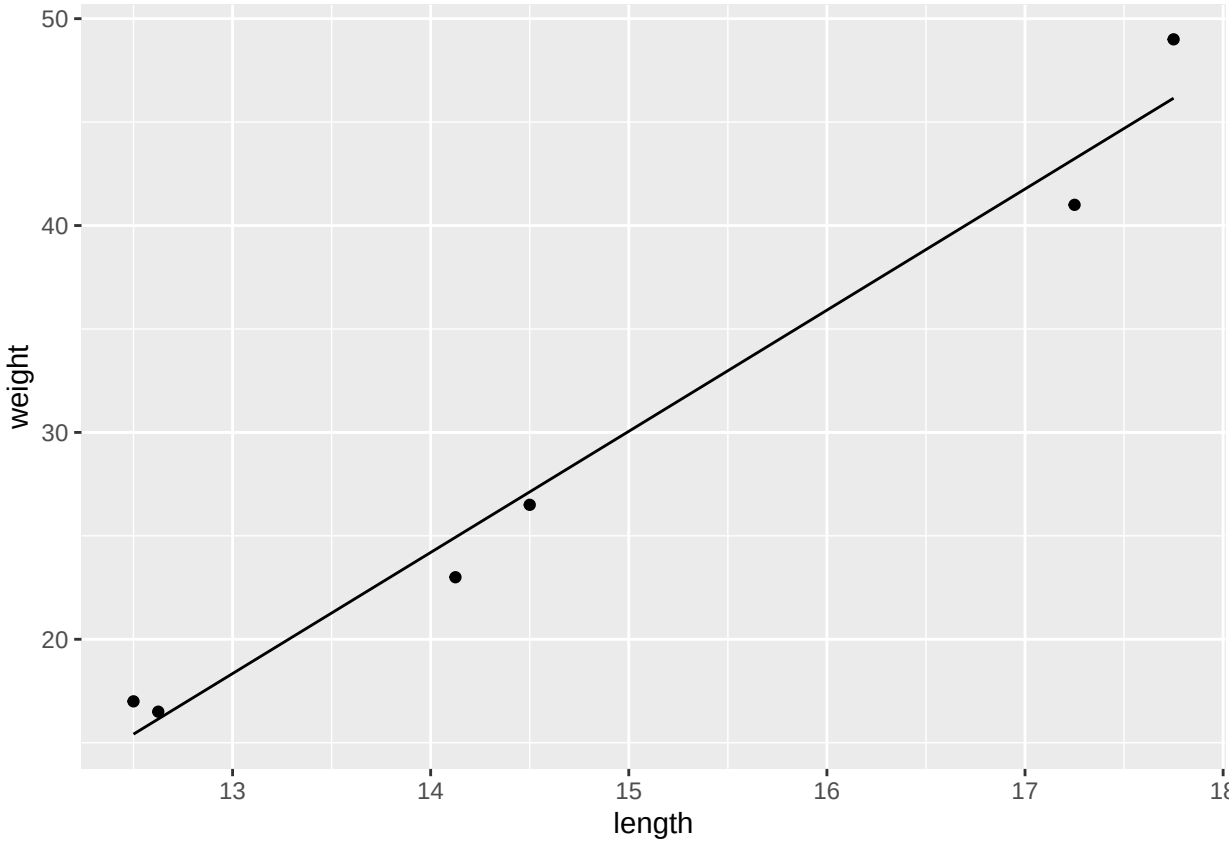



```
bassmodel <- lm(weight ~ length, data=bass)
```

```
coefficients <- bassmodel$coefficients
```

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

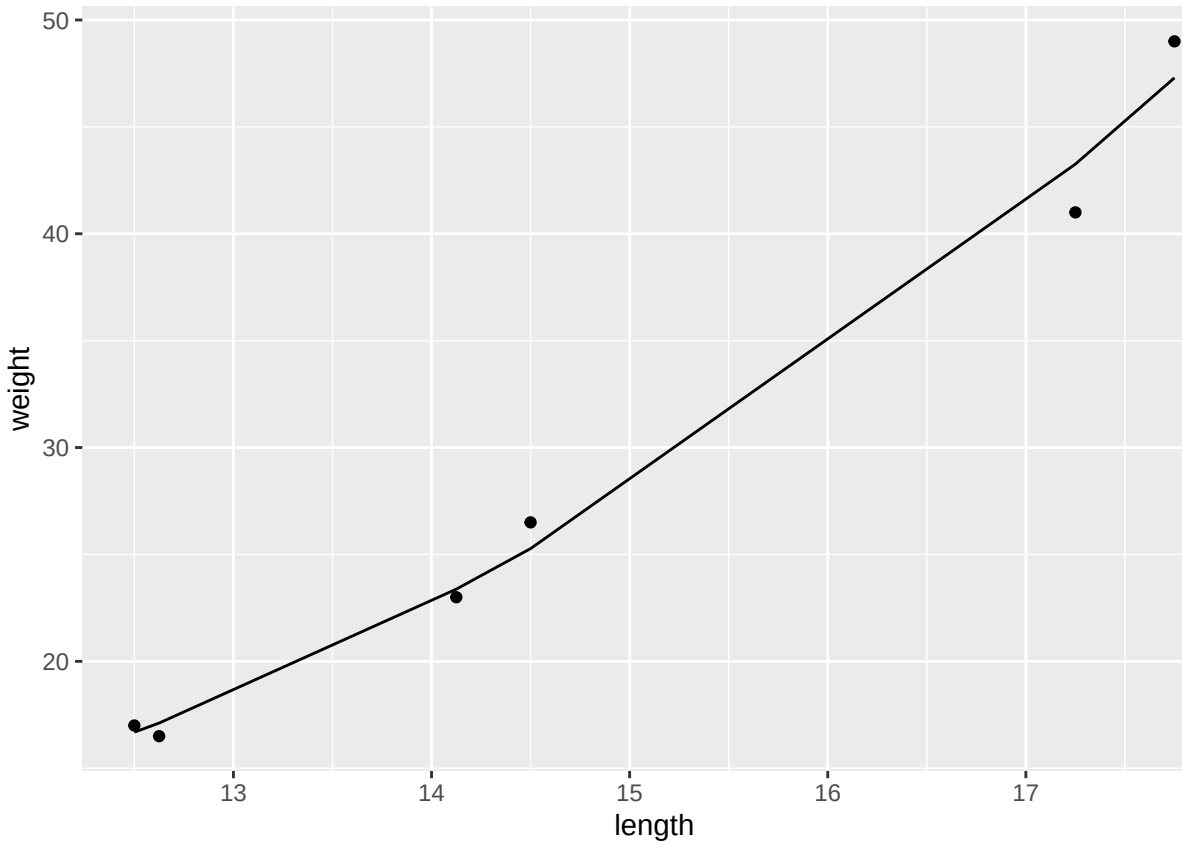
```
ggplot(bass) + geom_point(aes(x=length, y=weight)) + geom_line(aes(x=length, y=bassest))
```



Linear Model

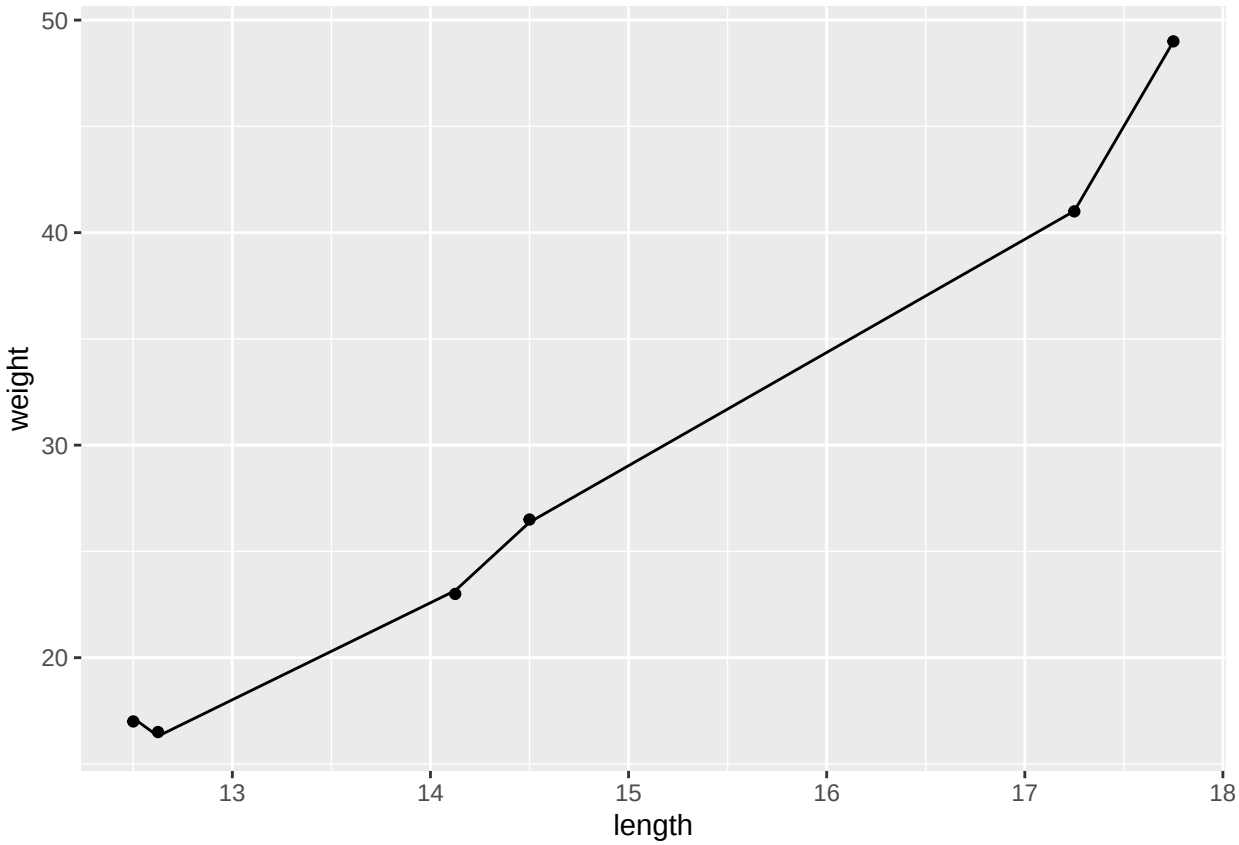
```
e <- ggplot(mydf, aes(x = n, y = error)) + geom_point()
```

```
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))
```

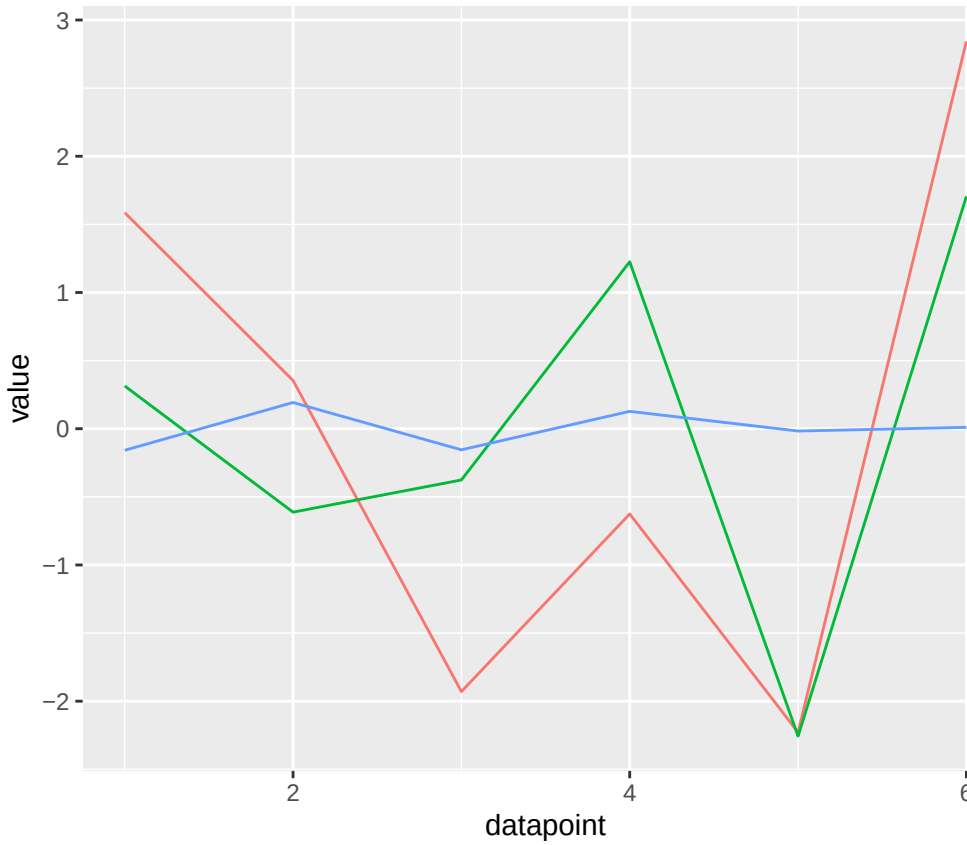


Quadratic Model

```
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 + co

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




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


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



Residuals of the Quadratic Model

The model that fits the data the bets would be the Quadratic model. This appears to be a case of overfit.

Section 2.1

Page 194: #1

```
ms <- function(x, len) {

  sq <- x^2
  len_sq <- len * 2

  sq <- as.character(stringr::str_pad(as.character(sq), len_sq, pad = "0"))

  start <- len_sq/2 - len/2 + 1
  end <- len_sq/2 + len/2

  return (as.numeric(substring(sq, start, end)))
}
```

Use the middle-square method to generate:

Part A. 10 random numbers using $x_0 = 1009$.

```
solution  <- 1009

for (i in 1:10) {
   solution[i + 1] <- ms(solution[i], 4)
}

print(solution)
```

```
##  [1] 1009  180  324 1049 1004   80   64   40   16    2    0
```

Part B. 20 random numbers using $x_0 = 653217$.

```
solution  <- 653217

for (i in 1:20) {
   solution[i + 1] <- ms(solution[i], 6)
}

print(solution)
```

```
##  [1] 653217 692449 485617 823870 761776 302674 611550 993402 847533 312186
## [11] 460098 690169 333248  54229 940784  74534 555317 376970 106380 316704
## [21] 301423
```

Part C. 15 random numbers using $x_0 = 3043$.

```
solution  <- 3043

for (i in 1:15) {
   solution[i + 1] <- ms(solution[i], 4)
}

print(solution)
```

```
##  [1] 3043 2598 7496 1900 6100 2100 4100 8100 6100 2100 4100 8100 6100 2100 4100
## [16] 8100
```

Part D. Comment about the results of each sequence. Was there cycling? Did each sequence degenerate rapidly?

Solution: In sequence 'a', degenerated to 0 was fairly quick within 10 iterations.

In sequence 'b', did not degenerate or cycle after 20 iterations.

In Sequence 'c', it cycles after hitting 6100 on the 4th iteration. Then repeats the cycle at 6100, 2100, 4100, 8100.

Page 211: #3

In many situations, the time T between deliveries and the order quantity Q is not fixed. Instead, an order is placed for a specific amount of gasoline. Depending on how many orders are placed in a given time interval, the time to fill an order varies. You have no reason to believe that the performance of the delivery operation will change. Therefore, you have examined records for the past 100 deliveries and found the following lag times, or extra days, required to fill your order:

Lag Time	# of Occurrences
2	10
3	25
4	30
5	20
7	13
7	2
	-
Total	100

Construct a Monte Carlo simulation for the lag time submodel. If you have a handheld calculator or computer available, test your submodel by running 1000 trials and comparing the number of occurrences of the various lag times with the historical data.

```
simulation <- function(iterations) {

  n <- iterations
  counter <- rep(0, 6)

  for (i in 1:n) {

    random <- runif(1)

    counter[1] <- counter[1] + (random >= 0    & random <= 0.1)
    counter[2] <- counter[2] + (random >  0.1  & random <= 0.35)
    counter[3] <- counter[3] + (random >  0.35 & random <= 0.65)
    counter[4] <- counter[4] + (random >  0.65 & random <= 0.85)
    counter[5] <- counter[5] + (random >  0.85 & random <= 0.98)
    counter[6] <- counter[6] + (random >  0.98 & random <= 1.0)
  }
  return (counter)
}
```

```
mydf6 <- data.frame(lag =  as.factor(2:7), occurrences = c(10, 25, 30, 20, 13, 2))
```

```
mydf6$probability  <- with(mydf6, occurrences / sum(occurrences))
mydf6$c_prob <- cumsum(mydf6$probability)
```

```


n <- 10000
counter <- simulation(10000)
```

```
mydf6$simulated_number[1] <- counter[1]
mydf6$simulated_number[2] <- counter[2]
mydf6$simulated_number[3] <- counter[3]
mydf6$simulated_number[4] <- counter[4]
mydf6$simulated_number[5] <- counter[5]
mydf6$simulated_number[6] <- counter[6]
```

```
mydf6$simulated_prob[1] <- counter[1]/n
mydf6$simulated_prob[2] <- counter[2]/n
mydf6$simulated_prob[3] <- counter[3]/n
mydf6$simulated_prob[4] <- counter[4]/n
mydf6$simulated_prob[5] <- counter[5]/n
mydf6$simulated_prob[6] <- counter[6]/n
```

```
mydf6 %>%
  kable() %>%
  kable_styling(full_width = F)
```

lag	occurrences	probability	c_prob	simulated_number	simulated_prob
2	10	0.10	0.10	975	0.0975
3	25	0.25	0.35	2574	0.2574
4	30	0.30	0.65	2969	0.2969
5	20	0.20	0.85	2026	0.2026
6	13	0.13	0.98	1261	0.1261
7	2	0.02	1.00	195	0.0195

Simulation