HW3

Raphael Nash

9/5/2017

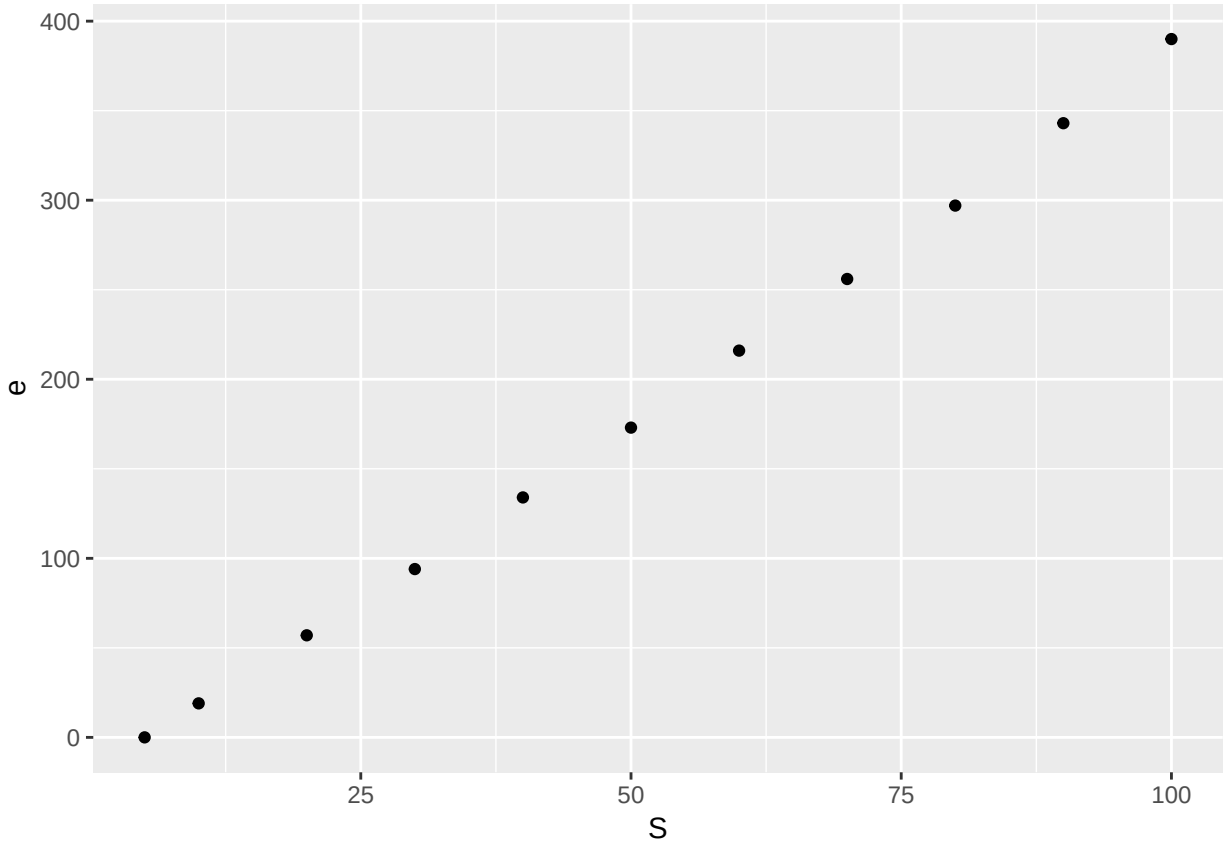
```
library("ggplot2")
```

p113 #2

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 model $e = c_1 S$ by plotting the data. Estimate c_1 graphically.

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

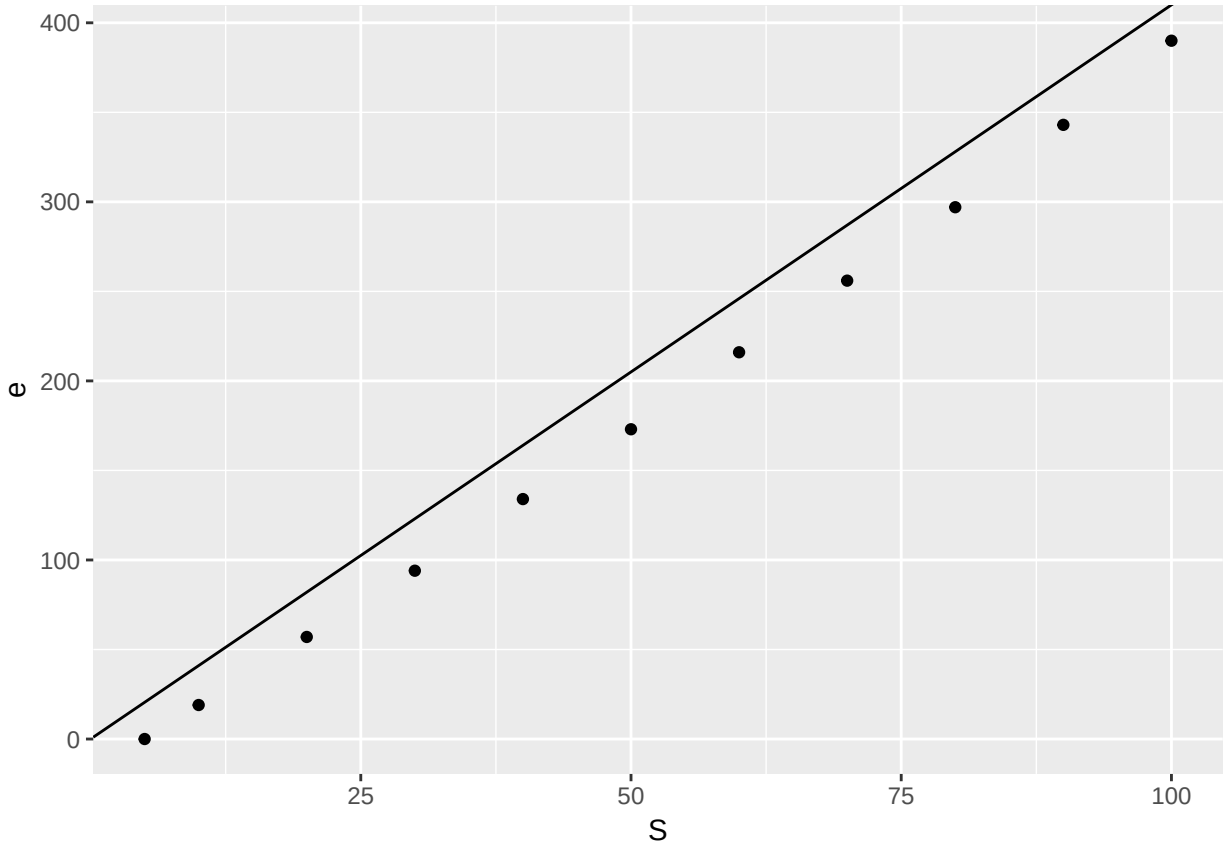
```
ggplot( data= problem_data , aes( x=S, y=e) ) + geom_point()
```




```
#c1 <- (297-19)/(90-10)
c1 <- (297-256)/(90-80)
c1
```

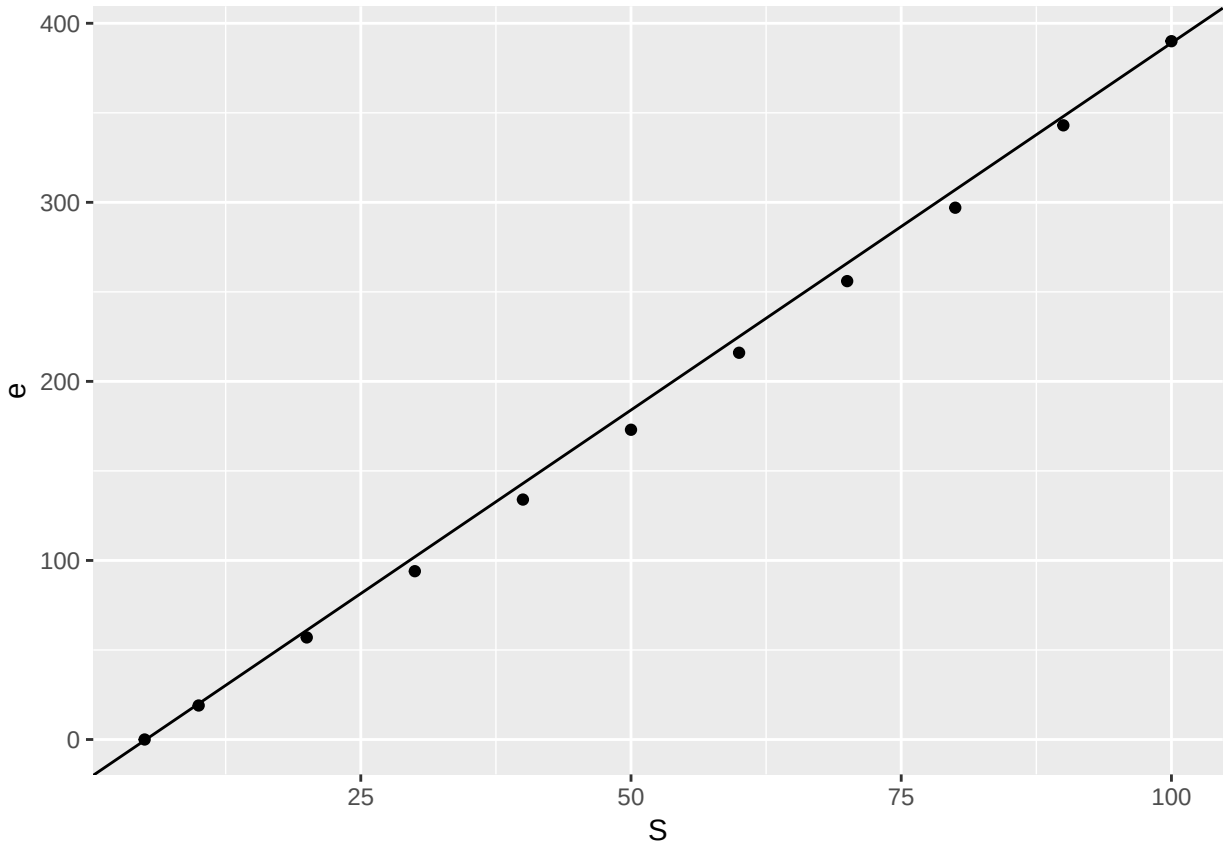
```
## [1] 4.1
```

```
ggplot( data= problem_data , aes( x=S, y=e) ) + geom_point() + geom_abline(intercept=0, slope=c1)
```



Moving intercept manually

```
ggplot( data= problem_data , aes( x=S, y=e) ) + geom_point() + geom_abline(intercept=-21, slope=c1)
```



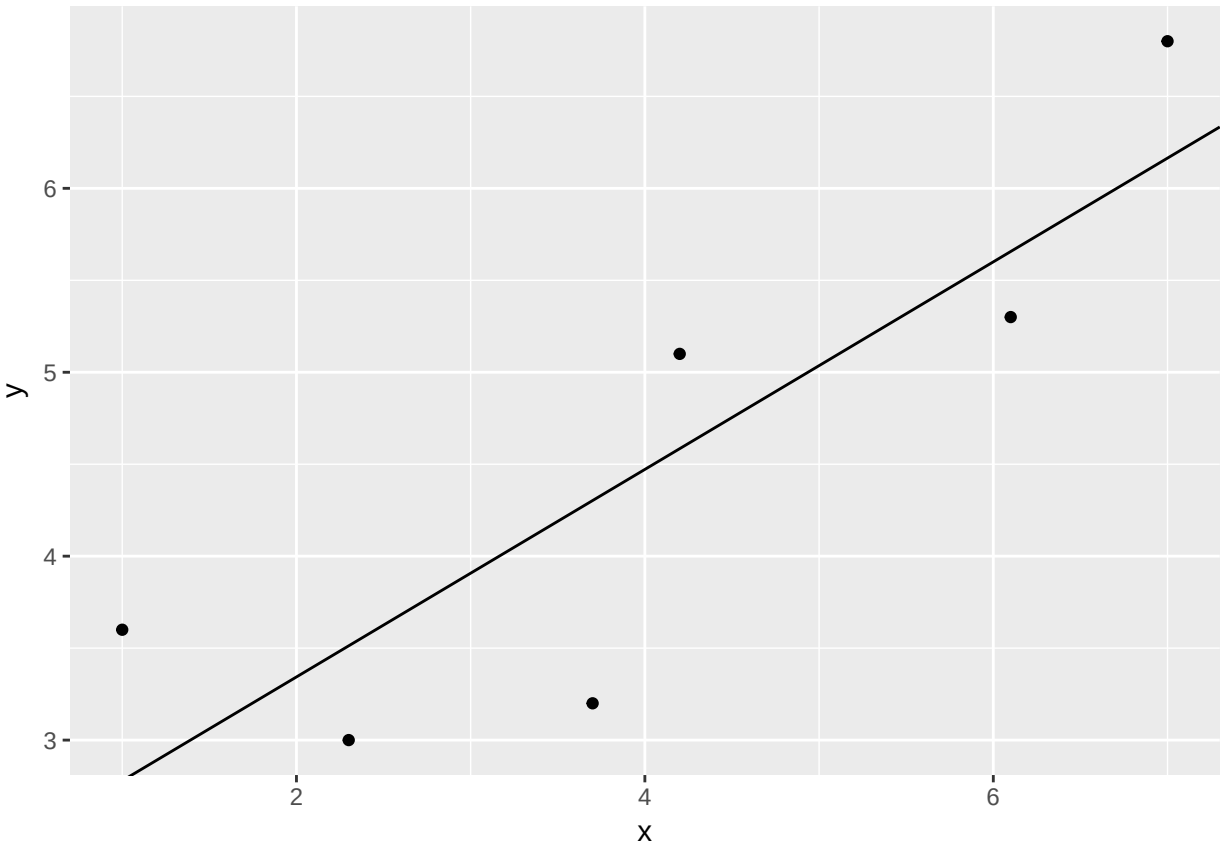
p 121 2.a

```
problem_data_121_2a <- data.frame(
  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)
)

lm <- lm(y ~ x, problem_data_121_2a)
lm$coefficients
```

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

```
ggplot(data=problem_data_121_2a) +
  geom_point(aes(x=x, y=y)) +
  geom_abline(intercept= lm$coefficients[[1]], slope=lm$coefficients[[2]])
```



The equation that theese points fit is: y = 0.6x +2.2

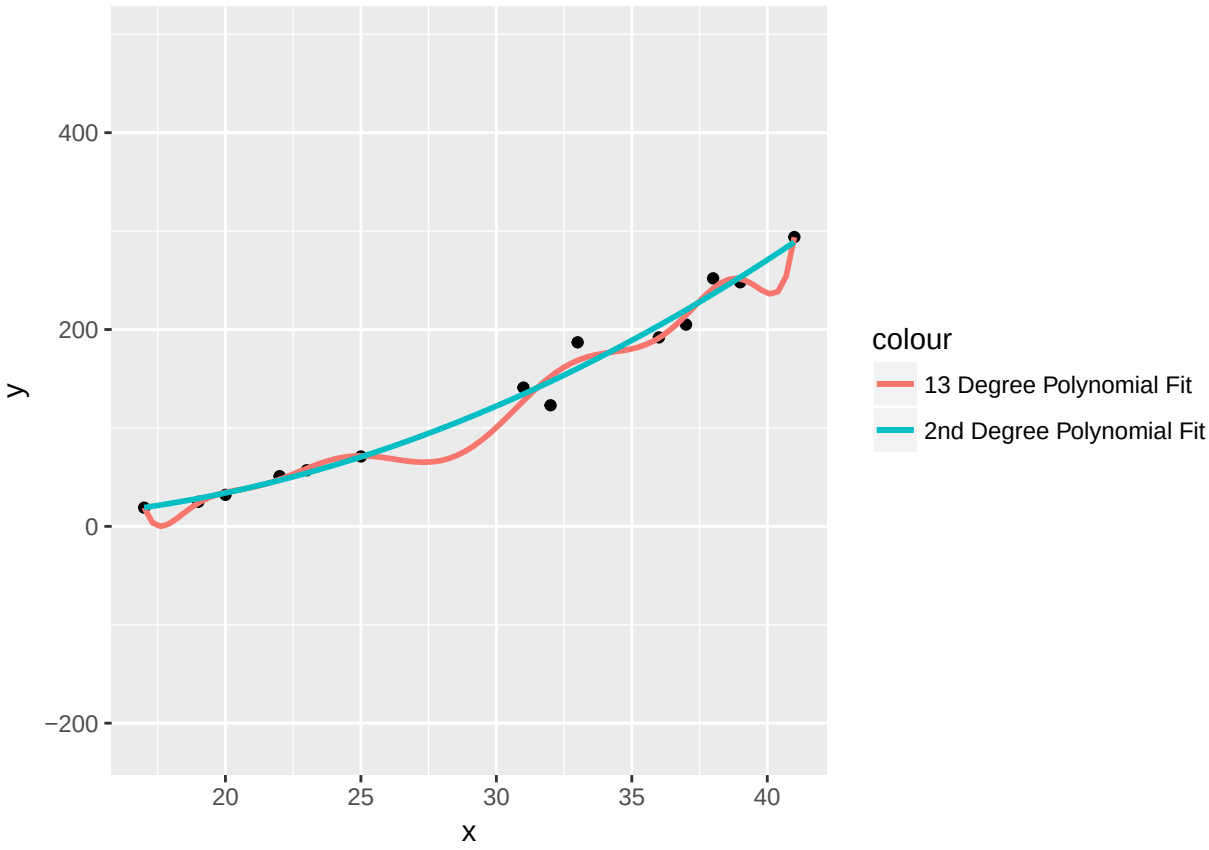
p 157 #4

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 board feet divided by 10. Make a scatterplot of the data. Discuss the appropriateness of using a 13th-degree polynomial that passes through the data points as an empirical model. If you have a computer available, t a polynomial to the data and graph the results.

```
problem_157_4 = 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)
  )

ggplot(problem_157_4, aes(x=x, y=y) ) + geom_point() +
  stat_smooth(method="lm", se=TRUE, fill=NA, formula=y ~ poly(x, 13, raw=TRUE), aes(colour = '13 Degree
  stat_smooth(method="lm", se=TRUE, fill=NA, formula=y ~ poly(x, 2, raw=TRUE), aes(colour = '2nd Degree
```

```
## Warning in predict.lm(model, newdata = data.frame(x = xseq), se.fit = se, :
## prediction from a rank-deficient fit may be misleading
```



As you can see the 13th degree and 2rd degree polynomial fit the data pretty good. The 13th degree goes through more points and has less "error" based on the data provided. It however probably overfits the data. The 2nd degree also fits the data pretty good. The 2nd degree poly has a lot less chance of overfitting the model since it does not turn on a dime to adjust to every point on the graph. It may though have more overall "error"