Intro To Regression

James Woods

Assignment and Access

1. Lets start with a few R conventions. Lets start with scalars. '<-' is the assignment operator in R. '=' also works but '==' is used for comparisons.

```
1+1
```

```
## [1] 2
```

```
A <-  2+3
A + 2
```

```
## [1] 7
```

```
B <- A +4
A+B
```

```
## [1] 14
```

```
C = A+B
C
```

```
## [1] 14
```

```
A == B
```

```
## [1] FALSE
```

2. R has many ways of representing values

```
A <- "Shale"
A
```

```
## [1] "Shale"
```

```
C <- c("Tom", "Dick", "Harry")
C
```

```
## [1] "Tom"   "Dick"  "Harry"
```

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
D <- 1:10
D
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
as.character(D)
```

```
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
```

```r
as.factor(D)
```

```
##  [1] 1  2  3  4  5  6  7  8  9  10
## Levels: 1 2 3 4 5 6 7 8 9 10
```

```r
summary(D)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    3.25    5.50    5.50    7.75   10.00
```

```r
summary(as.character(D))
```

```
##    Length     Class      Mode
##        10 character character
```

```r
summary(as.factor(D))
```

```
##  1  2  3  4  5  6  7  8  9 10
##  1  1  1  1  1  1  1  1  1  1
```

3. There are also functions. Most, but not all, work on many things at once.

```r
sqrt(D)
```

```
##  [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751
##  [8] 2.828427 3.000000 3.162278
```

```r
#sqrt(A)

sqrt(D[6])
```

```
## [1] 2.44949
```

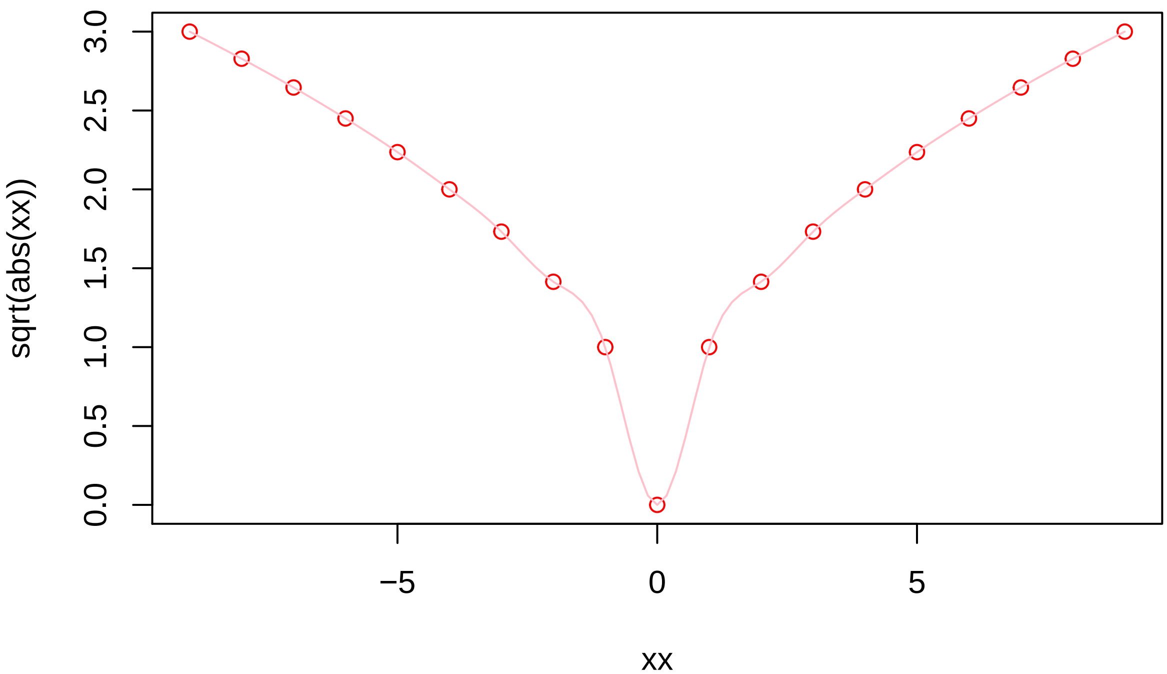
```r
sqrt(D[1:3])
```

```
## [1] 1.000000 1.414214 1.732051
```

4. If you need help, use the help pane or ask for help. Everything has a help file and many functions come with examples of how to use.

```r
help(sqrt)
example(sqrt)
```

```
##
## sqrt> require(stats) # for spline
##
## sqrt> require(graphics)
##
## sqrt> xx <- -9:9
##
## sqrt> plot(xx, sqrt(abs(xx)),  col = "red")
```




```
##
## sqrt> lines(spline(xx, sqrt(abs(xx)), n=101), col = "pink")
```

4. You can work with vectors and matrix but you will most frequently deal with dataframes, which is a matrix with extra attributes. Dataframes are objects that have variables inside them. You can access those variables with specific functions or with a '$'.

```r
X <- runif(20, 2, 40)
MyData <- as.data.frame(X)
MyData$Y <-  100 - 2*MyData$X +rnorm(20)

names(MyData)
```

```
## [1] "X" "Y"
```

```r
names(MyData) <- c("Price","Quantity")
names(MyData)
```

```
## [1] "Price"    "Quantity"
```

```r
summary(MyData)
```

```
##      Price           Quantity
##  Min.   : 2.274   Min.   :20.89
##  1st Qu.: 8.929   1st Qu.:49.35
##  Median :16.043   Median :67.54
##  Mean   :17.239   Mean   :65.60
##  3rd Qu.:25.550   3rd Qu.:82.02
##  Max.   :38.959   Max.   :96.01
```

5. You can get at columns and rows in other ways.

```r
MyData[1:2]
```

```
##        Price Quantity
## 1  15.831475 67.91359
## 2   7.370678 85.94109
## 3   4.101431 92.33850
## 4  25.459489 49.59623
## 5   9.448518 80.71947
## 6   4.750363 91.83278
## 7  10.227161 79.05250
## 8  10.738834 78.07294
## 9  38.958831 20.88632
## 10  2.273739 95.21228
## 11 16.254610 67.15910
## 12 32.392242 35.33110
## 13  2.913173 96.00633
## 14 13.128225 73.29739
## 15 23.250306 53.81789
## 16 21.966850 56.26392
## 17 26.120150 46.91280
## 18 22.696905 54.77946
## 19 31.078532 38.16117
## 20 25.819634 48.61589
```

```r
MyData[1]
```

```
##        Price
## 1  15.831475
## 2   7.370678
## 3   4.101431
## 4  25.459489
## 5   9.448518
## 6   4.750363
```

```
## 7  10.227161
## 8  10.738834
## 9  38.958831
## 10  2.273739
## 11 16.254610
## 12 32.392242
## 13  2.913173
## 14 13.128225
## 15 23.250306
## 16 21.966850
## 17 26.120150
## 18 22.696905
## 19 31.078532
## 20 25.819634
```

```
MyData[1:3,]
```

```
##       Price Quantity
## 1 15.831475 67.91359
## 2  7.370678 85.94109
## 3  4.101431 92.33850
```

```
MyData[1:3,"Price"]
```

```
## [1] 15.831475  7.370678  4.101431
```

```
MyData["Price"]
```

```
##        Price
## 1  15.831475
## 2   7.370678
## 3   4.101431
## 4  25.459489
## 5   9.448518
## 6   4.750363
## 7  10.227161
## 8  10.738834
## 9  38.958831
## 10  2.273739
## 11 16.254610
## 12 32.392242
## 13  2.913173
## 14 13.128225
## 15 23.250306
## 16 21.966850
## 17 26.120150
## 18 22.696905
## 19 31.078532
## 20 25.819634
```

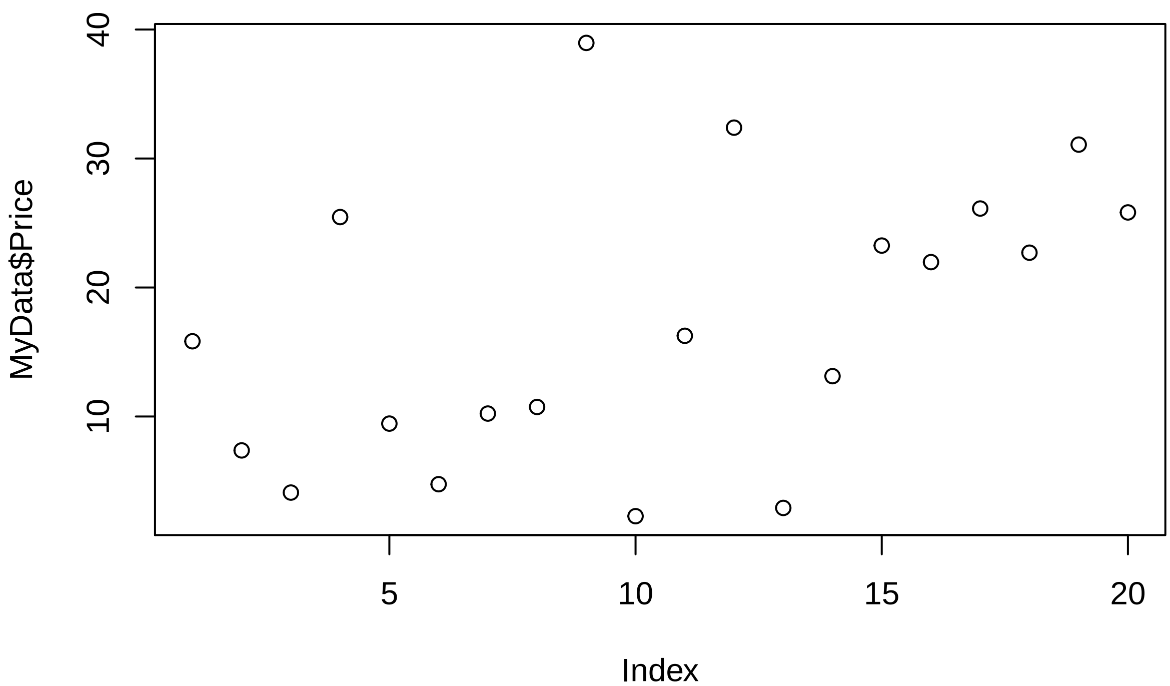
```r
MyData[c(1,5,8),]
```

```
##       Price Quantity
## 1 15.831475 67.91359
## 5  9.448518 80.71947
## 8 10.738834 78.07294
```

Pictures

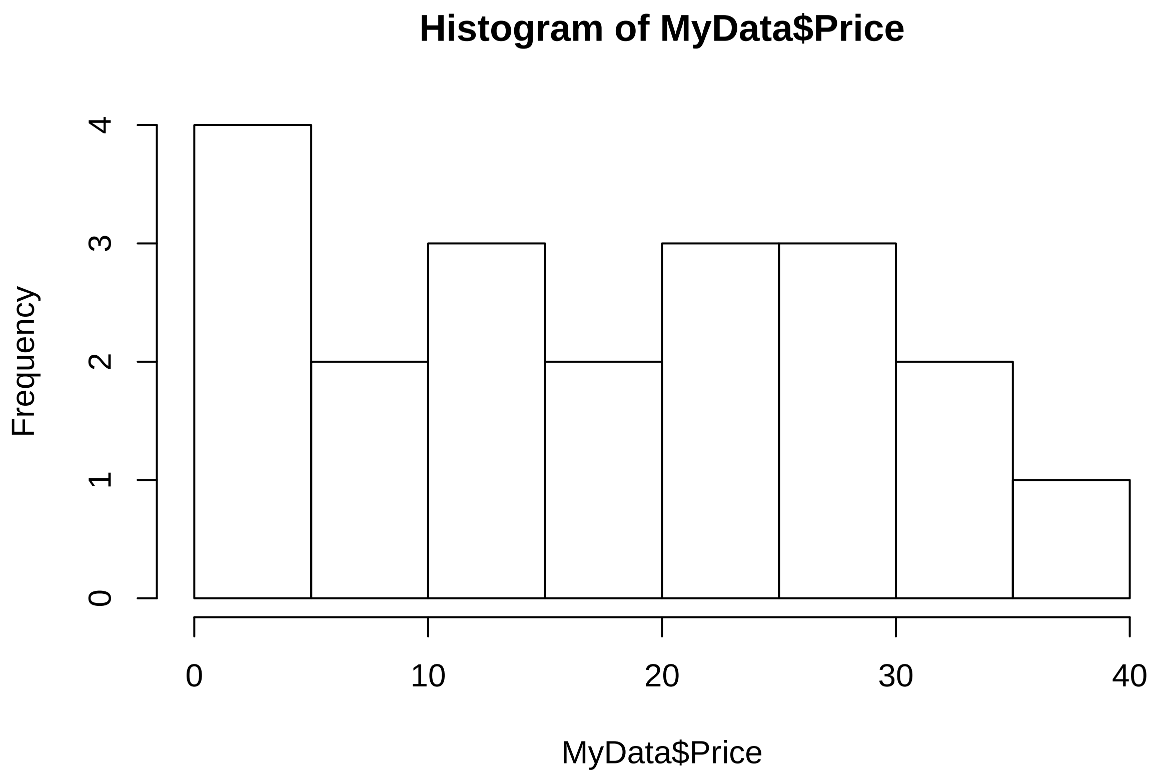
Now lets make pictures.

```r
plot(MyData$Price)
```



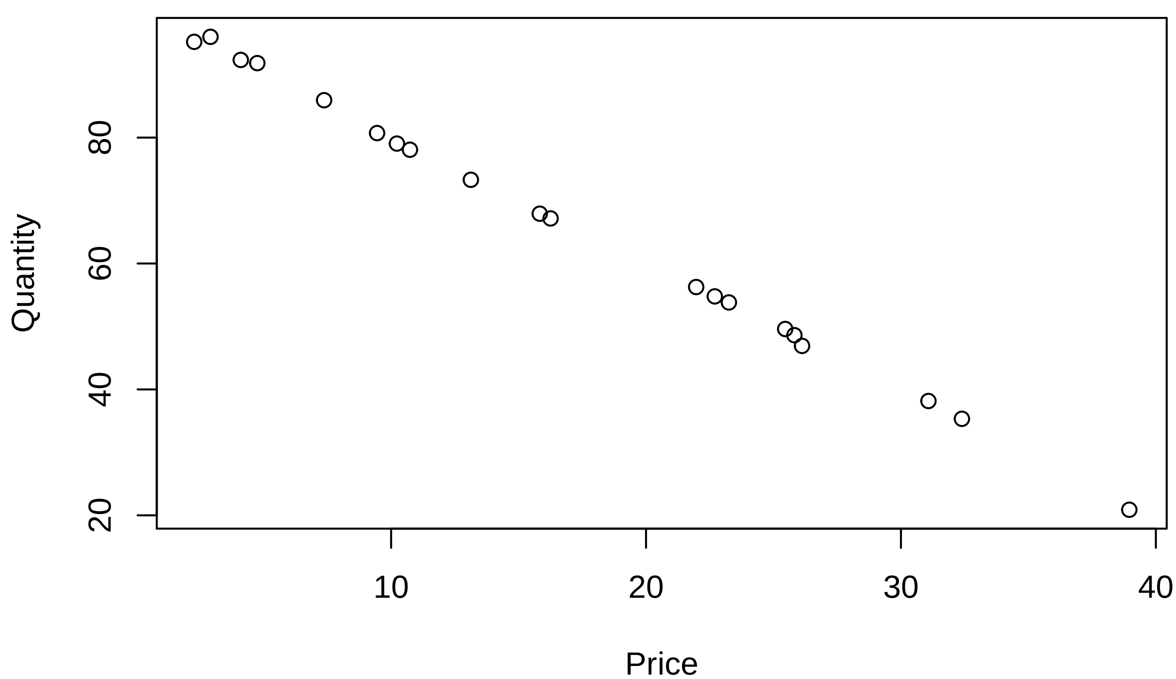
Historgram

```r
hist(MyData$Price)
```



Show the relationship

```
plot(Quantity~Price, data= MyData)
```



Regression with Pictures

5. A regression line can be thought of as just putting a line through a cloud of data in a well defined way. Remember the data wer created.

```
FirstRegression<- lm(Quantity ~ Price, data=MyData)

summary(FirstRegression)
```

```
##
## Call:
## lm(formula = Quantity ~ Price, data = MyData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.7498 -0.6675  0.1834  0.4134  1.3851
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 100.52359    0.28851   348.4   <2e-16 ***
## Price        -2.02610    0.01427  -142.0   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6738 on 18 degrees of freedom
## Multiple R-squared:  0.9991, Adjusted R-squared:  0.9991
## F-statistic: 2.015e+04 on 1 and 18 DF,  p-value: < 2.2e-16
```

Note that the estimates are the parameter, or very close to the parameters we used to create the data. Note also that the residual standard errors are the same as what we used to create the data. Nice when you know what is true.

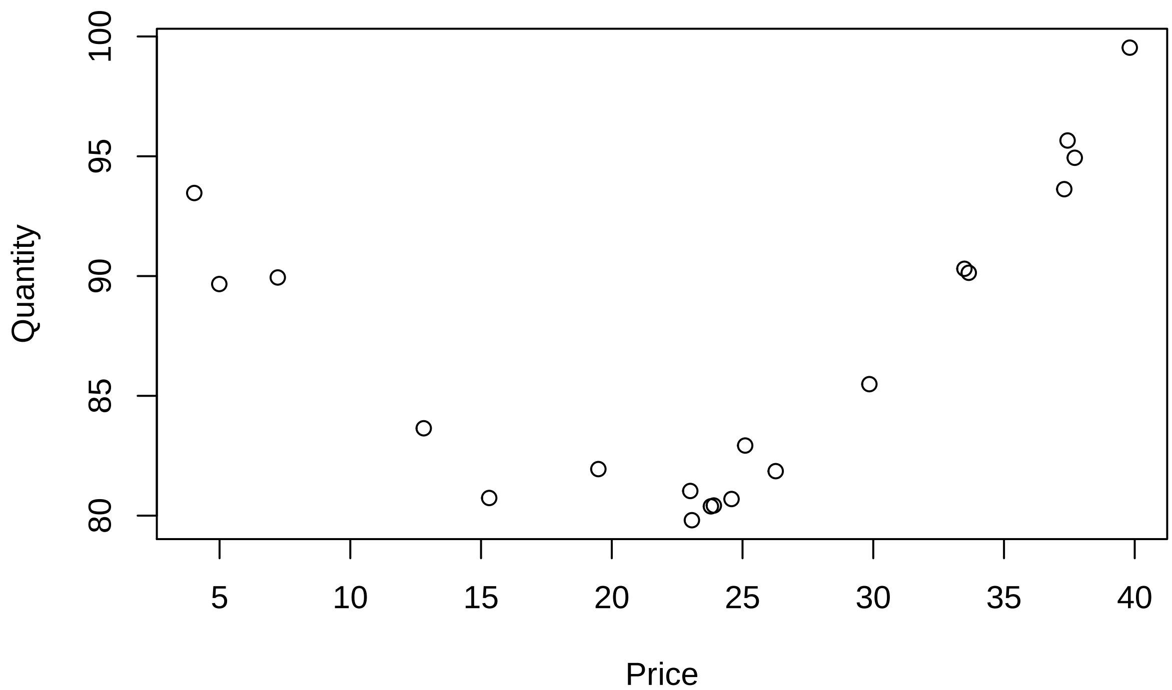
Regression can also show you things that are not true. You have to assume a functional form. Make new data.

```
MyData2<- data.frame(Price = runif(20, 2, 40))
MyData2$Quantity <-  100 - 2*MyData2$Price + .05*MyData2$Price^2 +rnorm(20)

summary(MyData2)
```

```
##      Price           Quantity
##  Min.   : 4.032   Min.   :79.81
##  1st Qu.:18.443   1st Qu.:80.96
##  Median :24.242   Median :84.57
##  Mean   :24.140   Mean   :86.81
##  3rd Qu.:33.528   3rd Qu.:91.09
##  Max.   :39.810   Max.   :99.53
```

Not a line

```
plot(Quantity~Price, data= MyData2)
```



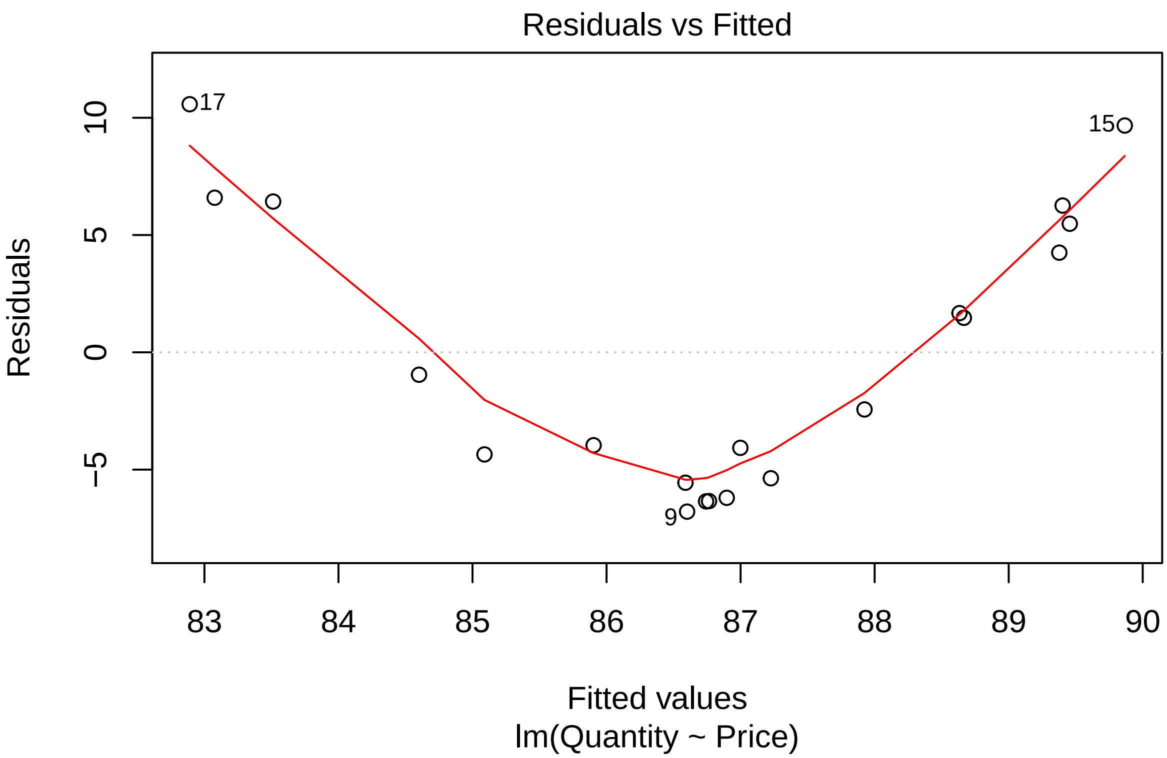
Give me a line anyway.

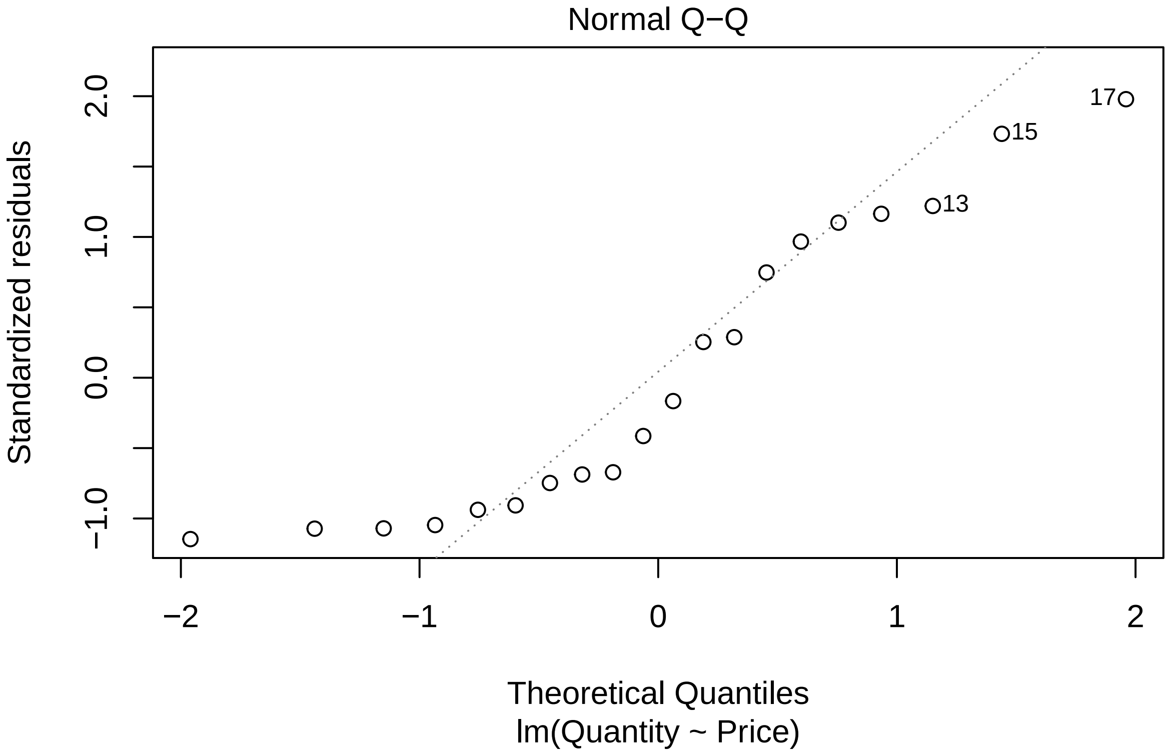
```
NotALine<- lm(Quantity ~ Price, data=MyData2)

summary(NotALine)
```

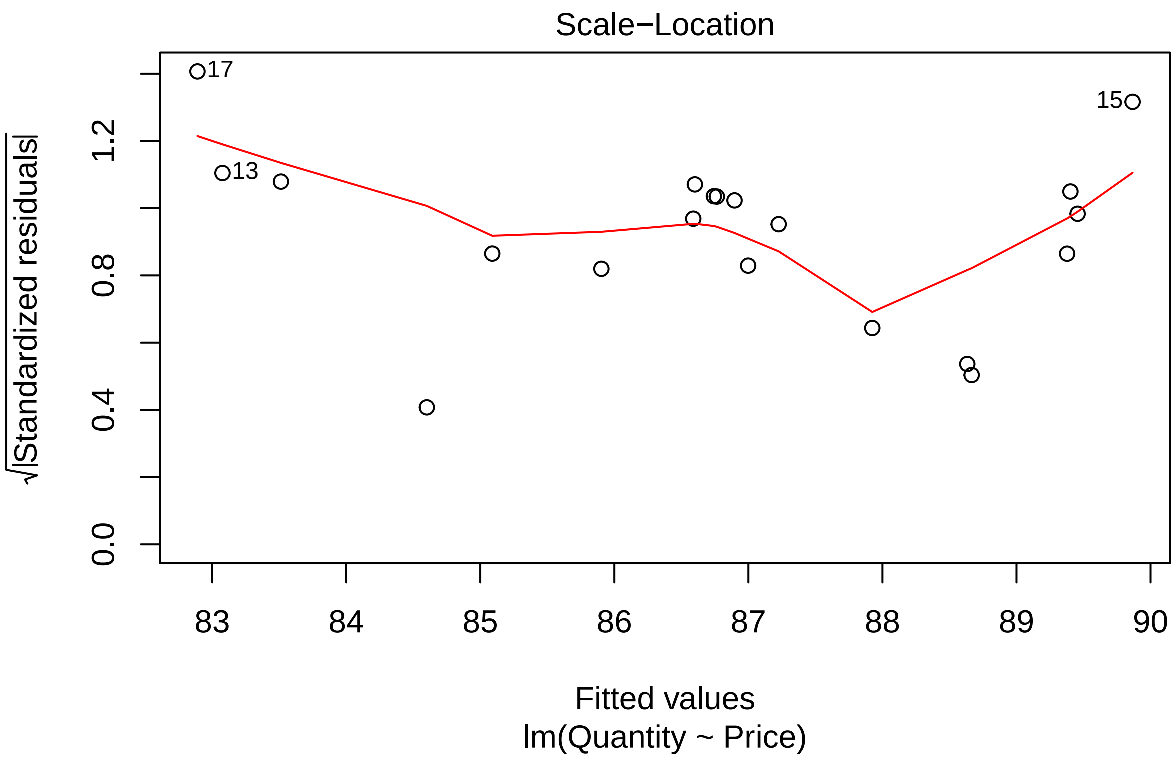
```
##
## Call:
## lm(formula = Quantity ~ Price, data = MyData2)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.791 -5.416 -1.695  5.676 10.576
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  82.1038     3.3585  24.446 2.94e-15 ***
## Price         0.1950     0.1272   1.533    0.143
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.079 on 18 degrees of freedom
## Multiple R-squared:  0.1154, Adjusted R-squared:  0.06629
## F-statistic: 2.349 on 1 and 18 DF,  p-value: 0.1428
```

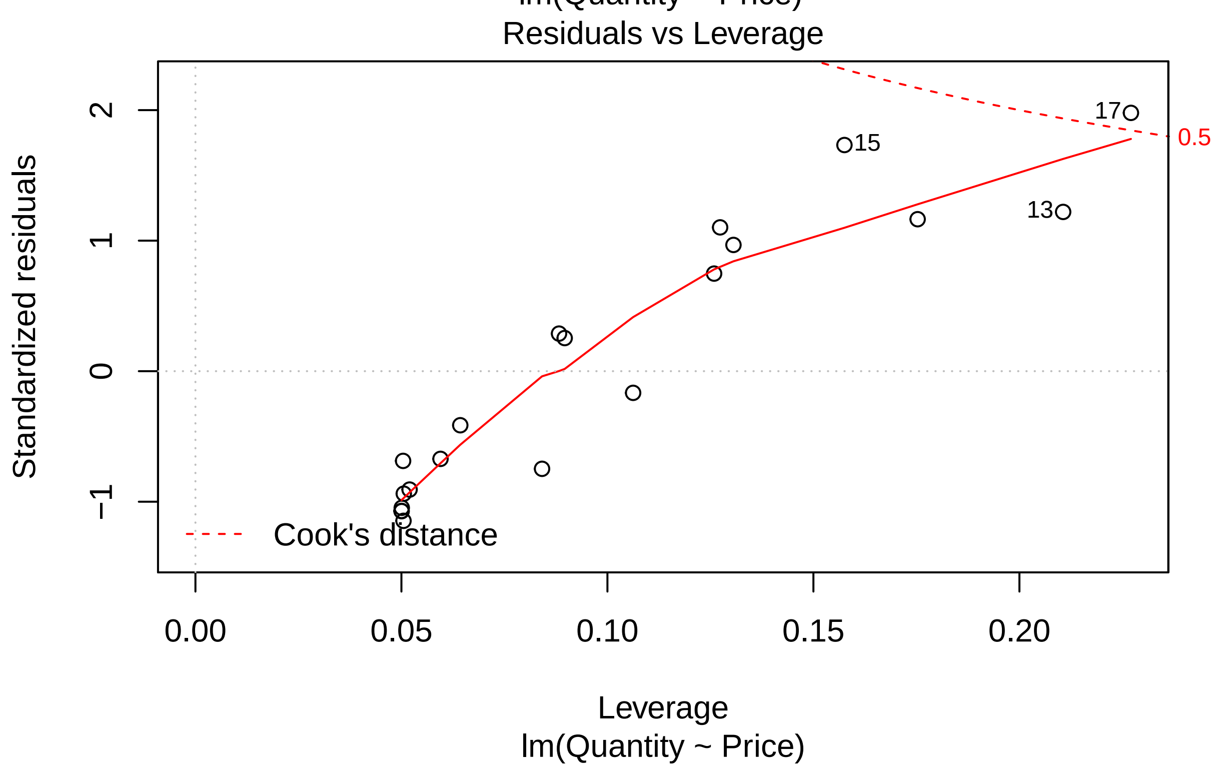
How to tell you were wrong thinking it was a line. Make a pictures

```
plot(NotALine)
```


Residuals vs Fitted
Residuals
10
5
0
−5
17
15
9
83
84
85
86
87
88
89
90
Fitted values
lm(Quantity ~ Price)

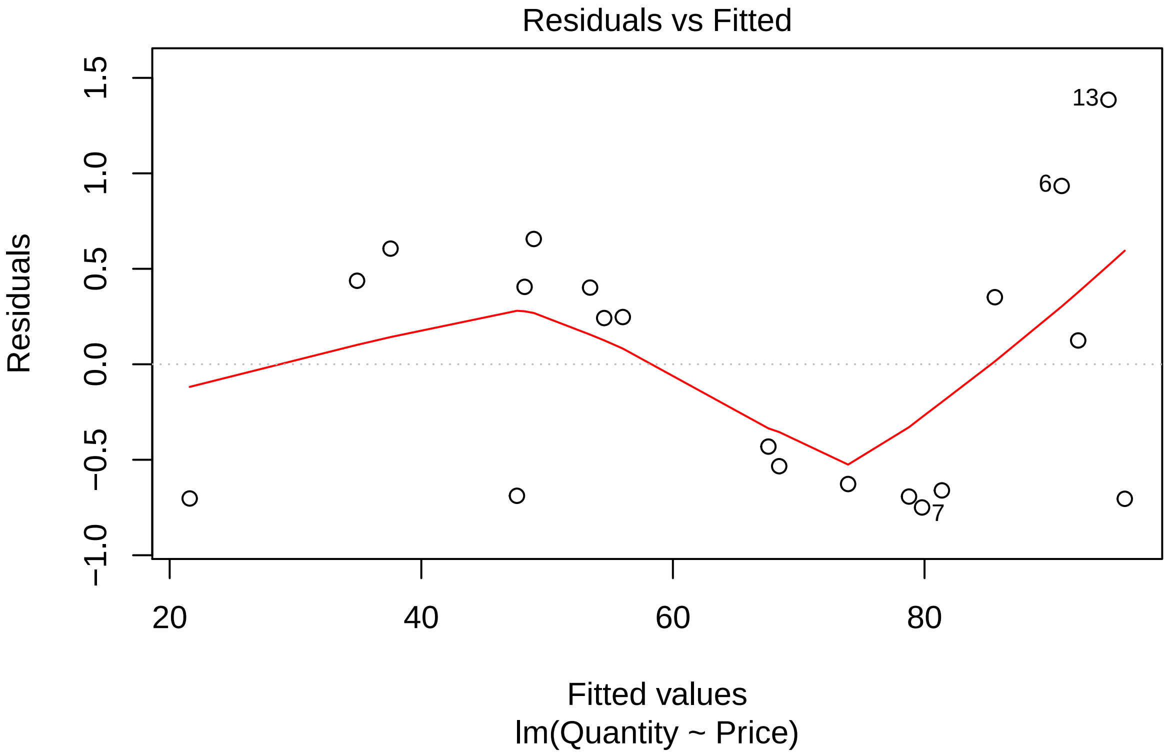

Normal Q−Q
Standardized residuals
2.0
1.0
0.0
−1.0
17
15
13
−2
−1
0
1
2
Theoretical Quantiles
lm(Quantity ~ Price)



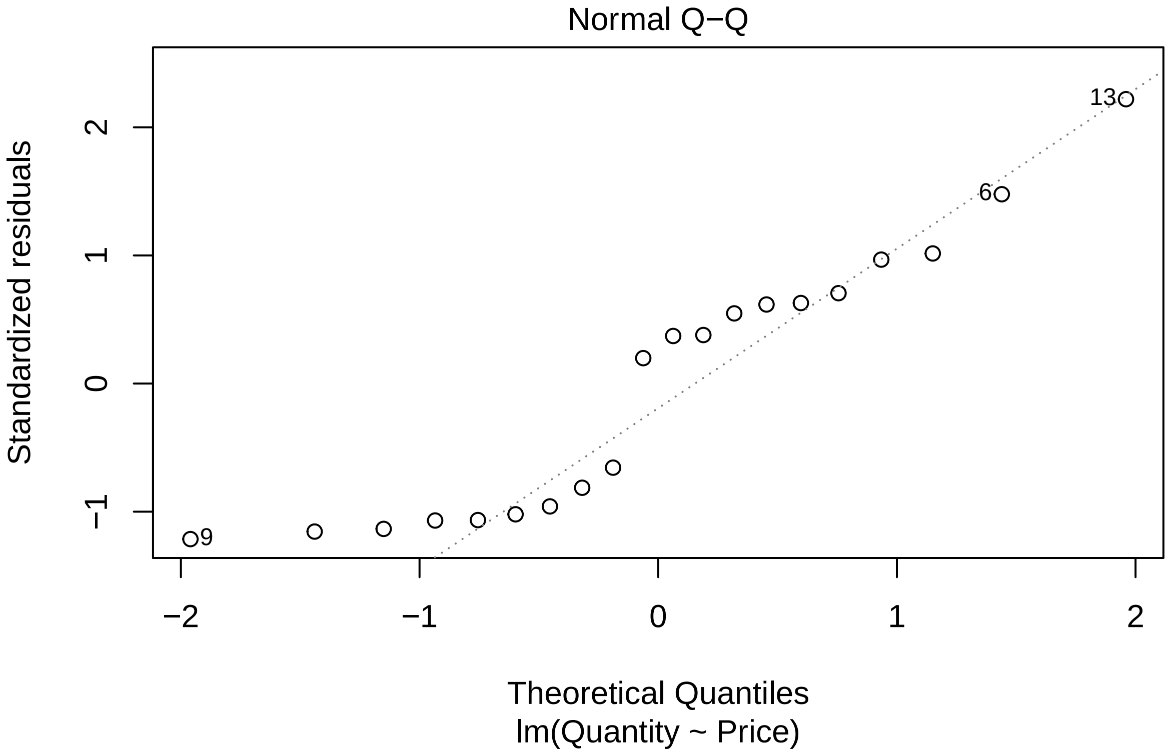


Compare with when it was a line and you estimated a line

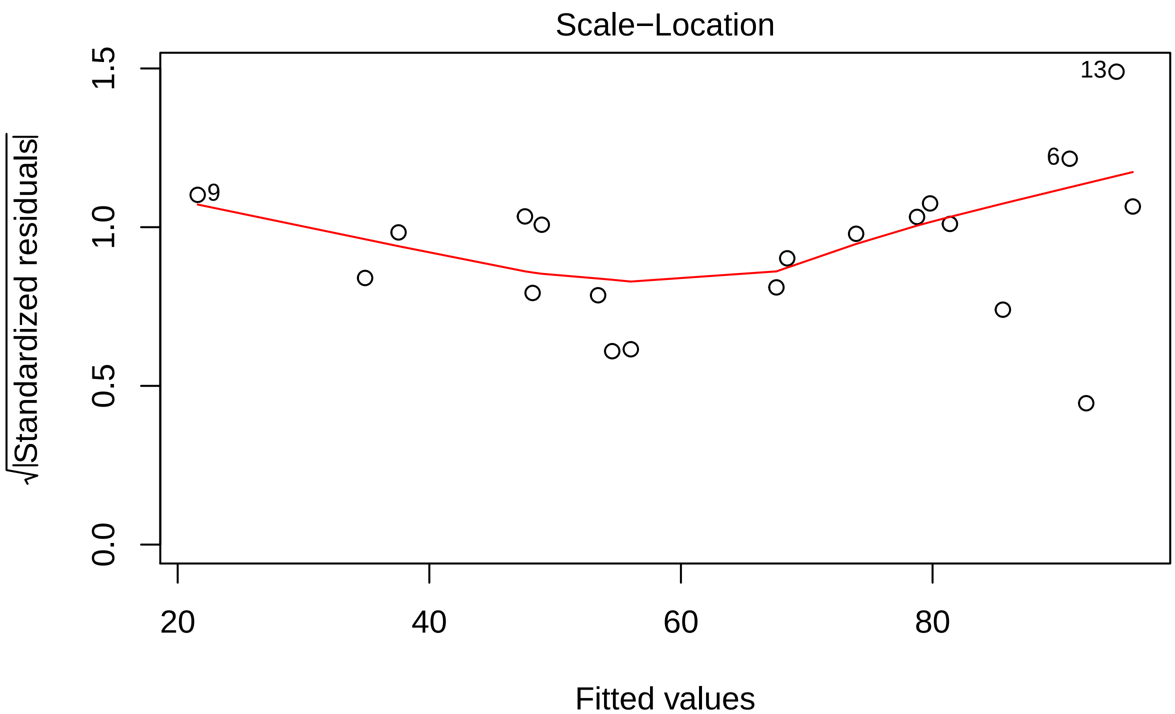
```
plot(FirstRegression)
```

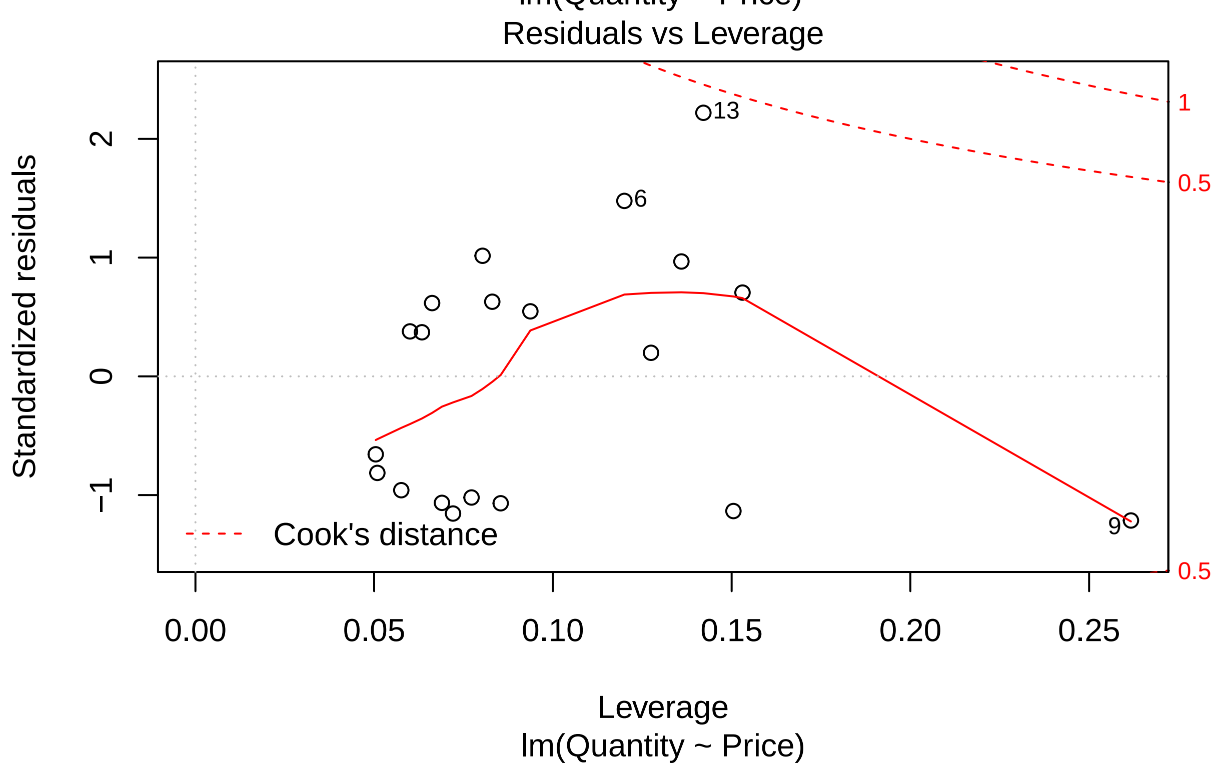


Residuals vs Fitted
Residuals
1.5
1.0
0.5
0.0
−0.5
−1.0
13
6
7
20
40
60
80
Fitted values
lm(Quantity ~ Price)



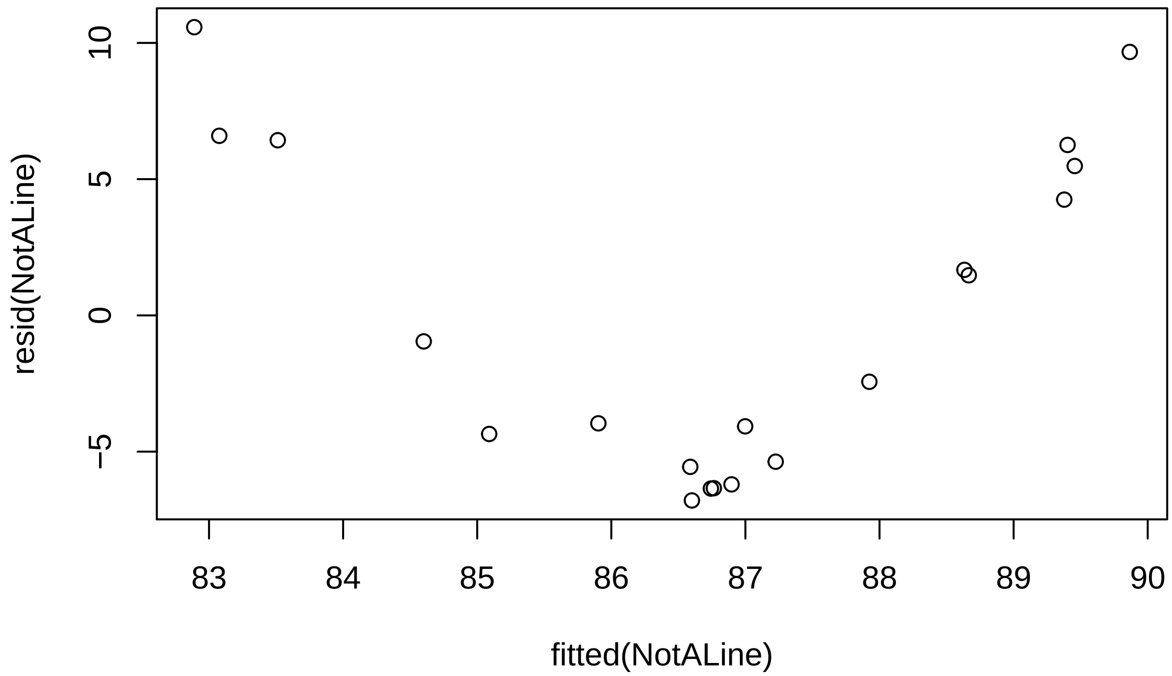
Normal Q−Q
Standardized residuals
2
1
0
−1
13
6
9
−2
−1
0
1
2
Theoretical Quantiles
lm(Quantity ~ Price)





You can make these by hand too. Here are two accessor functions.

```
plot(resid(NotALine)~fitted(NotALine))
```



Dummy Variables

Looking at Residuals for problems

2. I will walk you through a few steps on reading in the data. The biggest hurdle to doing stats on the data is reading it in. There is actually an R library for working with EIA data directly, EIAdata, but we will use the more general tools to read files.

3. Download into R data on coal prices and quantities. Again, the assignment operator in R is the "<-" symbol.

```
Coal <- read.csv("https://www.eia.gov/totalenergy/data/browser/csv.cfm?tbl=T06.01")
```

There are many ways of loading data into R (http://www.r-tutor.com/r-introduction/data-frame/data-import). Some work some of the time. In most cases Comma Separated Values (CSV) is the safest format to work with.

4. Take a look at the summary of the data

```
summary(Coal)
```

```
##       MSN          YYYYMM                   Value      Column_Order
##  CLEXPUS: 591   Min.   :194913   Not Available: 244   Min.   :1.00
##  CLIMPUS: 591   1st Qu.:198207   816.667      :   6   1st Qu.:2.75
##  CLLUPUS: 591   Median :199312   2            :   3   Median :4.50
##  CLNIPUS: 591   Mean   :199301   3            :   3   Mean   :4.50
##  CLPRPUS: 591   3rd Qu.:200504   114          :   2   3rd Qu.:6.25
##  CLSCPUS: 591   Max.   :201608   129          :   2   Max.   :8.00
```

```
##  (Other):1182                    (Other)      :4468
##                          Description                    Unit
##  Coal Consumption              : 591   Thousand Short Tons:4728
##  Coal Exports                  : 591
##  Coal Imports                  : 591
##  Coal Losses and Unaccounted for: 591
##  Coal Net Imports              : 591
##  Coal Production               : 591
##  (Other)                       :1182
```

You will notice that for some variables they give counts, e.g., MSN, and for others you get numerical summaries, e.g., Column_Order. The difference has to do with the data types (http://www.statmethods.net/input/datatypes.html).

5. Since we are trying to make a simple supply model, i.e., trying to explain coal production, lets select just the production part of the data set and also get only the annual production values. This is a little primer on changing data types and taking part of data.

- Load the dplyr package. This is the normal way to load libraries of functions that you need.

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

- Select only the Coal Production Figures and save it as CoalProduction. There is a cheat sheet for dplyr built into R. Look under the help menu.

```r
CoalProduction <- Coal %>% filter(MSN == "CLPRPUS")
```

- Note that you have monthly data but that the annual data is shown as month 13. Grab all of the observations with month equal to 13.

```r
library(stringr)

CoalProduction <- CoalProduction %>% filter(str_sub(as.character(YYYYMM),5 ) == "13")
```

- At this point you will have noticed that we don't need all the columns so lets keep just the ones we need and then give the two columns we saved new names.

```r
CoalProduction <- CoalProduction %>% select(YYYYMM, Value)

names(CoalProduction) <- c("RawYear", "ProductionKShortTon")
summary(CoalProduction)
```

```
##     RawYear        ProductionKShortTon
##  Min.   :194913   1000048.758: 1
##  1st Qu.:196563   1016458.418: 1
##  Median :198213   1029075.527: 1
##  Mean   :198213   1032973.77 : 1
##  3rd Qu.:199863   1033504.288: 1
##  Max.   :201513   1063855.51 : 1
##                   (Other)    :61
```

- Notice that the ProductionKShortTon variable shows a count rather than a numerical summary. This means that R thinks it is a factor rather than a number. Lets fix that. It requires to first convert the factor, which is an integer, to the real value as a character and then convert that to numeric.

```r
CoalProduction$ProductionKShortTon <- as.numeric(as.character(CoalProduction$ProductionKShortTon))
```

Play around with this doing one function at a time to see what each does and what each does alone.

- Next lets create a column for the year and make it a numeric value.

```r
CoalProduction <- CoalProduction  %>% mutate(Year = as.numeric(str_sub(as.character(RawYear),0,4 )))


summary(CoalProduction)
```

```
##     RawYear       ProductionKShortTon      Year
##  Min.   :194913   Min.   : 420423     Min.   :1949
##  1st Qu.:196563   1st Qu.: 558547     1st Qu.:1966
##  Median :198213   Median : 829700     Median :1982
##  Mean   :198213   Mean   : 796953     Mean   :1982
##  3rd Qu.:199863   3rd Qu.:1033239     3rd Qu.:1998
##  Max.   :201513   Max.   :1171809     Max.   :2015
```

5. Now grab some price data. We are going to use Quandl (https://www.quandl.com/), which has a bunch of data on energy and a lot of other things (https://www.quandl.com/collections/markets/coal). Make sure you have the Quandl library installed. If you don't install.packages("Quandl") should do it.

```r
library(Quandl)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
##
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```r
Prices <- Quandl("EPI/152")

summary(Prices)
```

```
##       Year             Price (U.S. Dollars)
##  Min.   :1949-01-01   Min.   :16.78
##  1st Qu.:1963-01-01   1st Qu.:20.19
##  Median :1977-01-01   Median :25.02
##  Mean   :1976-12-31   Mean   :27.85
##  3rd Qu.:1991-01-01   3rd Qu.:31.52
##  Max.   :2005-01-01   Max.   :50.92
```

- As before we will convert the year to a numeric value

```r
Prices$Year <- as.numeric(str_sub(Prices$Year,0,4))
```

- And simplify the names

```r
names(Prices) <- c("Year", "PriceMBTU")
```

Please note that we don't have the price per short ton of coal. What we have is the price per million BTUs, which is a measure of energy content. The BTUs per short ton of coal (2000 lbs) is about 20 MBTUs but varies from place-to-place and year-to-year.

- Merge the two data frames

```r
summary(CoalProduction)
```

```
##     RawYear       ProductionKShortTon      Year
##  Min.   :194913   Min.   : 420423     Min.   :1949
##  1st Qu.:196563   1st Qu.: 558547     1st Qu.:1966
##  Median :198213   Median : 829700     Median :1982
##  Mean   :198213   Mean   : 796953     Mean   :1982
##  3rd Qu.:199863   3rd Qu.:1033239     3rd Qu.:1998
##  Max.   :201513   Max.   :1171809     Max.   :2015
```

```r
summary(Prices)
```

```
##       Year        PriceMBTU
##  Min.   :1949   Min.   :16.78
##  1st Qu.:1963   1st Qu.:20.19
##  Median :1977   Median :25.02
##  Mean   :1977   Mean   :27.85
##  3rd Qu.:1991   3rd Qu.:31.52
##  Max.   :2005   Max.   :50.92
```

```
CoalMarket<-inner_join(Prices, CoalProduction, by ="Year")

summary(CoalMarket)
```

```
##       Year        PriceMBTU        RawYear        ProductionKShortTon
##  Min.   :1949   Min.   :16.78   Min.   :194913   Min.   : 420423
##  1st Qu.:1963   1st Qu.:20.19   1st Qu.:196313   1st Qu.: 529774
##  Median :1977   Median :25.02   Median :197713   Median : 684913
##  Mean   :1977   Mean   :27.85   Mean   :197713   Mean   : 750200
##  3rd Qu.:1991   3rd Qu.:31.52   3rd Qu.:199113   3rd Qu.: 995984
##  Max.   :2005   Max.   :50.92   Max.   :200513   Max.   :1131498
```