Intro To Regression

James Woods

This is something like a notebook. You can work with data and text at the same time. Things chunks are for code

```
print("I am in a chunk.")
```

```
## [1] "I am in a chunk."
```

By default they echo the code and then produce the result.

You can turn off the echo and just see the result.

```
## [1] "No Echo"
```

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
```

Note that when you save values to variables you can see them in the Environment tab.

Play with this in the chunk below and run to see what happens.

2. R has many ways of representing values

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

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

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

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

```r
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
```

There are many others, “list” being the most important not shown.

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
```

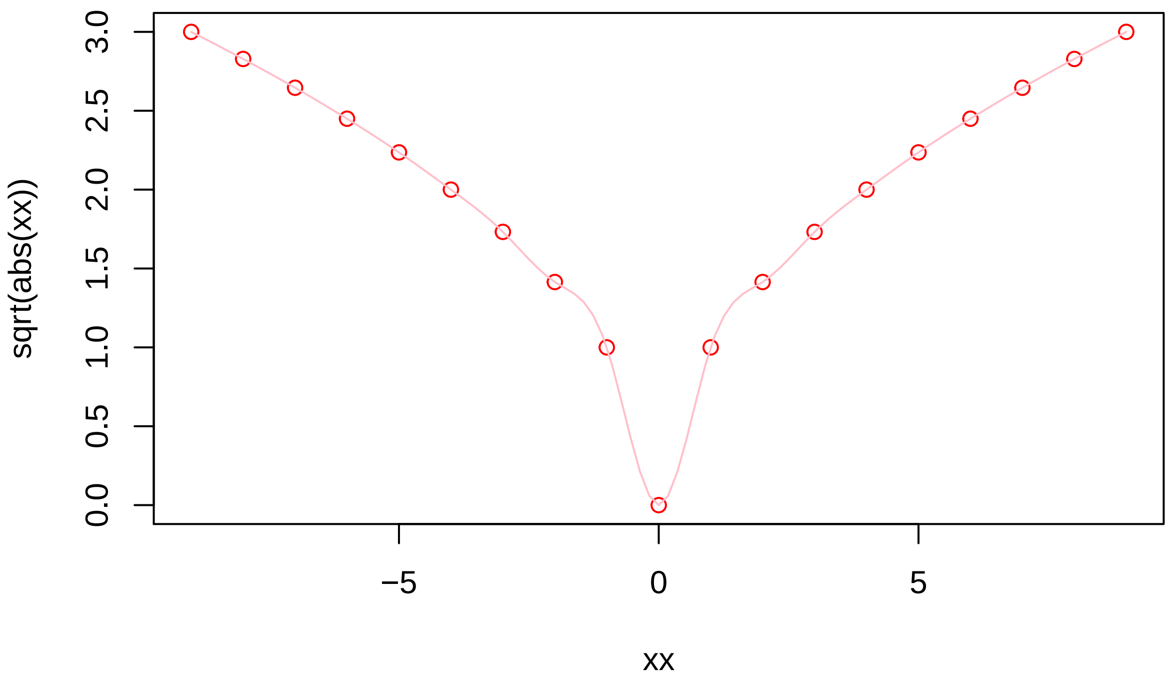
```r
sqrt(D[1:3])
```

```
## [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 '$'.

```
X <- runif(20, 2, 40)
MyData <- as.data.frame(X)
MyData$Y <-  100 - 2 * MyData$X + rnorm(20, mean = 0, sd = 1)

names(MyData)
```

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

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

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

```
summary(MyData)
```

```
##      Price          Quantity    
##  Min.   : 2.99   Min.   :22.08  
##  1st Qu.:15.05   1st Qu.:51.31  
##  Median :19.93   Median :59.73  
##  Mean   :20.51   Mean   :58.99  
##  3rd Qu.:24.72   3rd Qu.:69.83  
##  Max.   :38.64   Max.   :94.95
```

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

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

```
##        Price Quantity
## 1  20.466132 59.08826
## 2  15.306832 66.78173
## 3  23.618657 53.59829
## 4   5.097639 89.62193
## 5   2.989633 94.95143
## 6  36.177622 26.33105
## 7  15.444948 67.81517
## 8  15.273082 69.19427
## 9  20.151092 60.27521
## 10 31.124408 38.69608
## 11 22.138930 55.57921
## 12 14.379346 71.75647
## 13 28.043522 44.43357
## 14 19.711879 59.18522
## 15 12.067485 77.32345
## 16 20.959311 59.07557
## 17 38.636934 22.08061
## 18 12.723748 74.34274
## 19 38.410083 23.63324
## 20 17.551746 65.94663
```

```
MyData[1]
```

```
##        Price
## 1  20.466132
## 2  15.306832
## 3  23.618657
## 4   5.097639
## 5   2.989633
## 6  36.177622
## 7  15.444948
## 8  15.273082
## 9  20.151092
## 10 31.124408
## 11 22.138930
## 12 14.379346
## 13 28.043522
## 14 19.711879
## 15 12.067485
## 16 20.959311
## 17 38.636934
## 18 12.723748
## 19 38.410083
## 20 17.551746
```

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

```
##      Price Quantity
## 1 20.46613 59.08826
## 2 15.30683 66.78173
## 3 23.61866 53.59829
```

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

```
## [1] 20.46613 15.30683 23.61866
```

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

```
##        Price
## 1  20.466132
## 2  15.306832
## 3  23.618657
## 4   5.097639
## 5   2.989633
## 6  36.177622
## 7  15.444948
## 8  15.273082
## 9  20.151092
## 10 31.124408
## 11 22.138930
## 12 14.379346
## 13 28.043522
## 14 19.711879
## 15 12.067485
## 16 20.959311
## 17 38.636934
## 18 12.723748
## 19 38.410083
## 20 17.551746
```

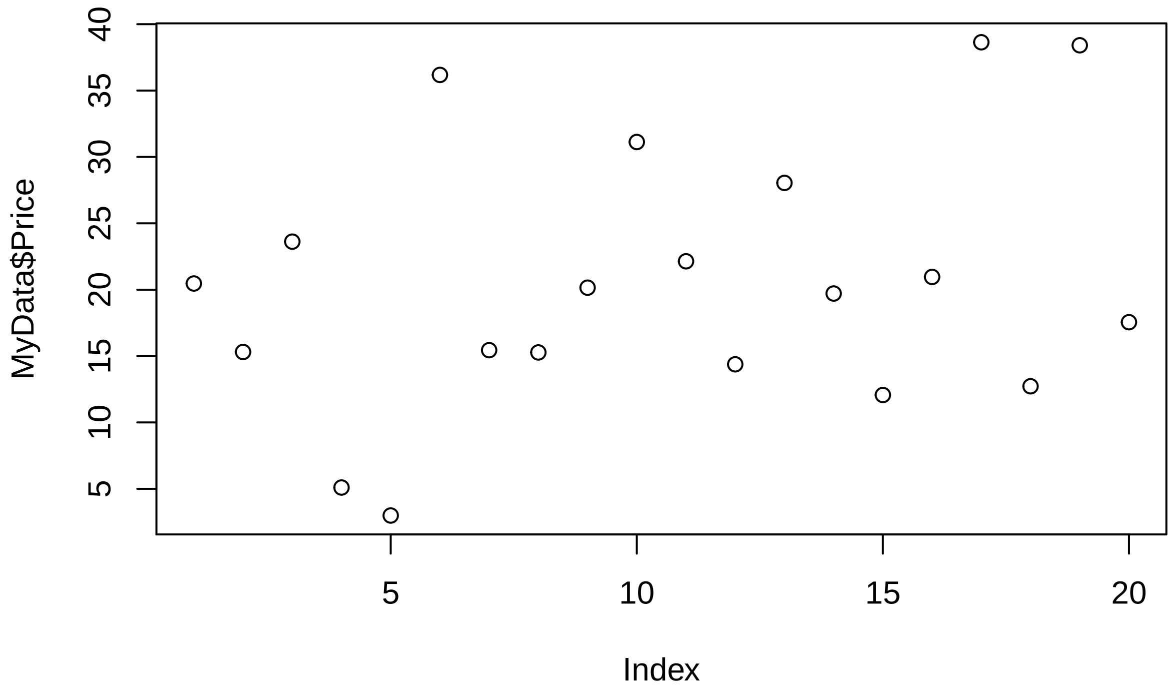
```
MyData[c(1,5,8),]
```

```
##       Price Quantity
## 1 20.466132 59.08826
## 5  2.989633 94.95143
## 8 15.273082 69.19427
```

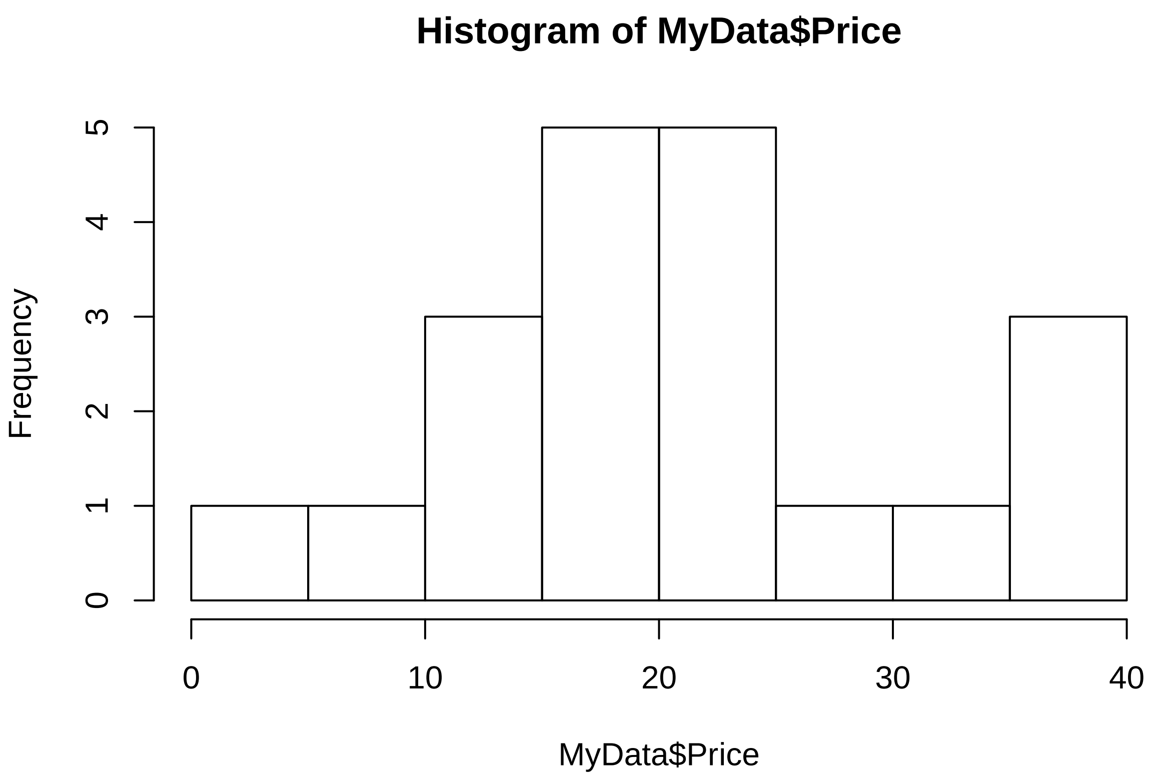
Pictures

Now lets make pictures.

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

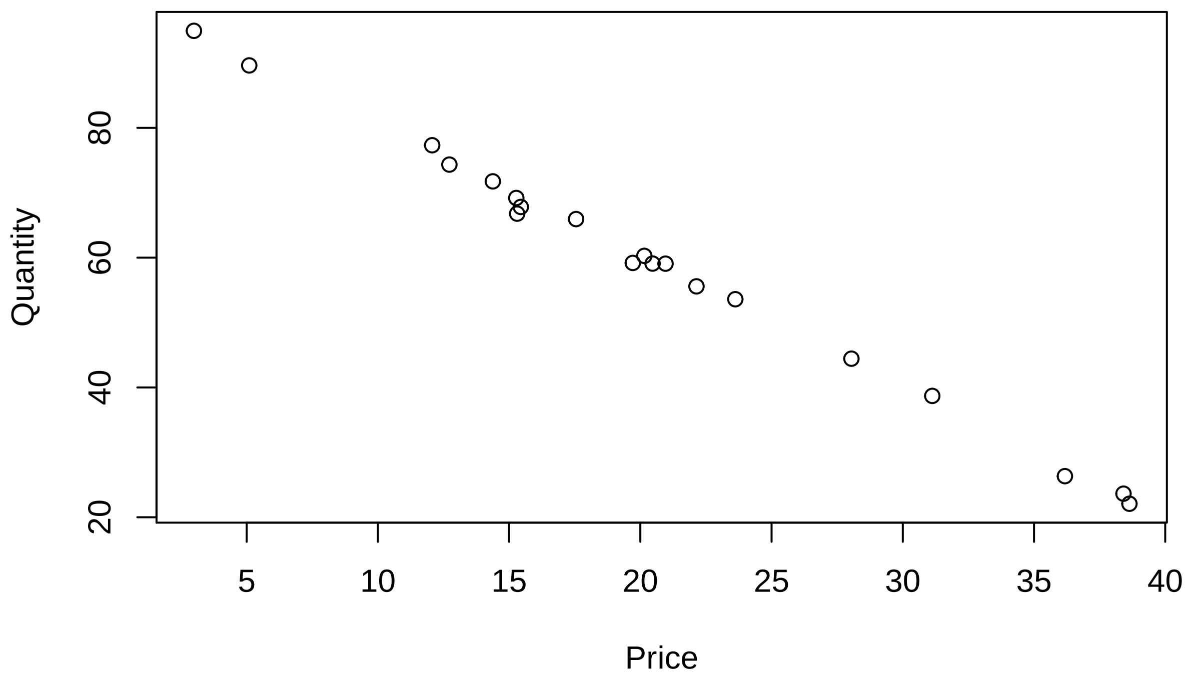


If you don't want to see the code and just the result, make sure to add echo=false to the chunk. Here is a Histogram without the code.



Show the relationship between two variables with "plot" or more than two with "pairs".

```r
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 we created? Here is what the regression gives you. You may want to make some notes on how to write formulas for regression in this file.

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

summary(FirstRegression)
```

```
##
## Call:
## lm(formula = Quantity ~ Price, data = MyData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6603 -0.3691  0.2296  0.7924  1.3761
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 100.18157    0.55604  180.17   <2e-16 ***
## Price        -2.00823    0.02451  -81.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.063 on 18 degrees of freedom
## Multiple R-squared:  0.9973, Adjusted R-squared:  0.9972
## F-statistic:  6715 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.

Some other things you should note:

- ADD *NOTES* TO THIS LIST
- MORE NOTES FOR YOU TO ADD

You should probably make a lot of notes in this document so you can remember what we are talking about. Just add things. By the way, you can do equations too, $\frac{1}{2}x$ or even

$$\lim_{x\to+\infty} \frac{3x^2+7x^3}{x^2+5x^4} = 3$$

If you take 400 level math classes, your homework will be in latex like this.

Regression can also show you things that are not true. You always have to assume a functional form and if you get that functional form wrong you get garbage. Sometimes the garbage makes some kind of sense and sometimes it leads you to believe that you have to make some kind of statistical fix. Students that have taken econometrics will be aware of autocorrelation, heterskedasticity and endogenaity as common problems. Having the wrong functional form will often present as one of these problems.

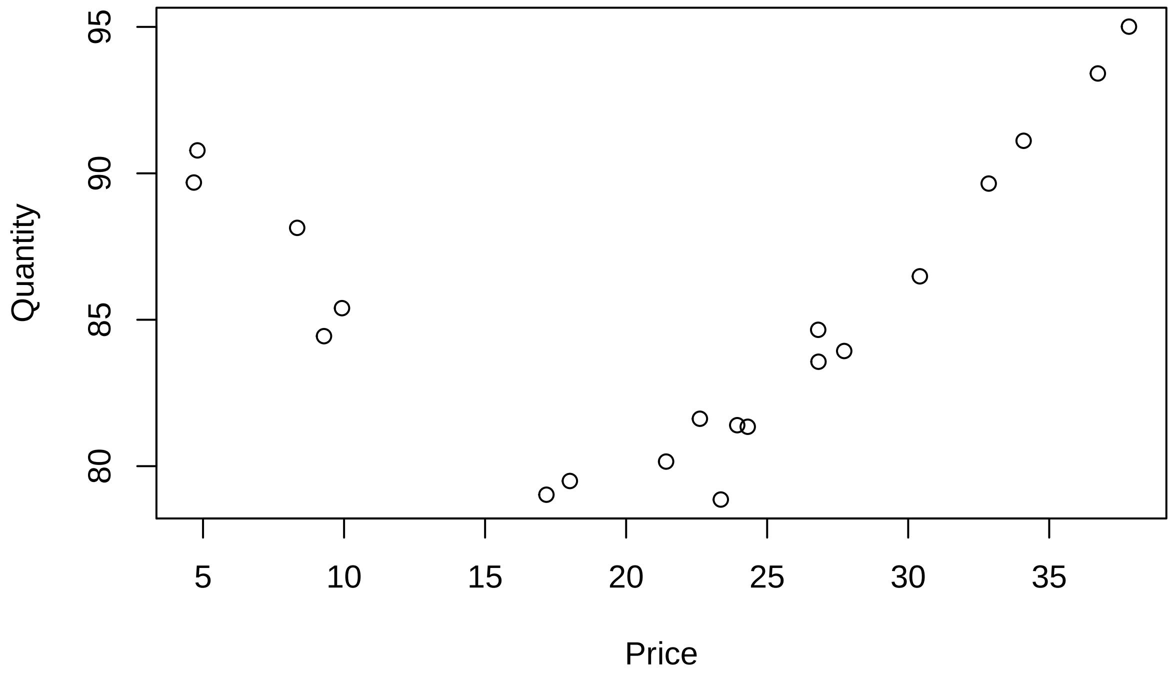
Lets wake new data and show how getting the regression model wrong can cause problems.

```
MyData2 <- data.frame(Price = runif(20, 2, 40))
MyData2$Quantity <-  100 - 2 * MyData2$Price + .05 * MyData2$Price^2 + rnorm(20, mean =0, sd = 1)

summary(MyData2)
```

```
##      Price           Quantity
##  Min.   : 4.674   Min.   :78.86
##  1st Qu.:15.362   1st Qu.:81.39
##  Median :23.648   Median :84.55
##  Mean   :22.056   Mean   :85.41
##  3rd Qu.:28.403   3rd Qu.:89.66
##  Max.   :37.828   Max.   :95.01
```

Not a line

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



Give me a line anyway.

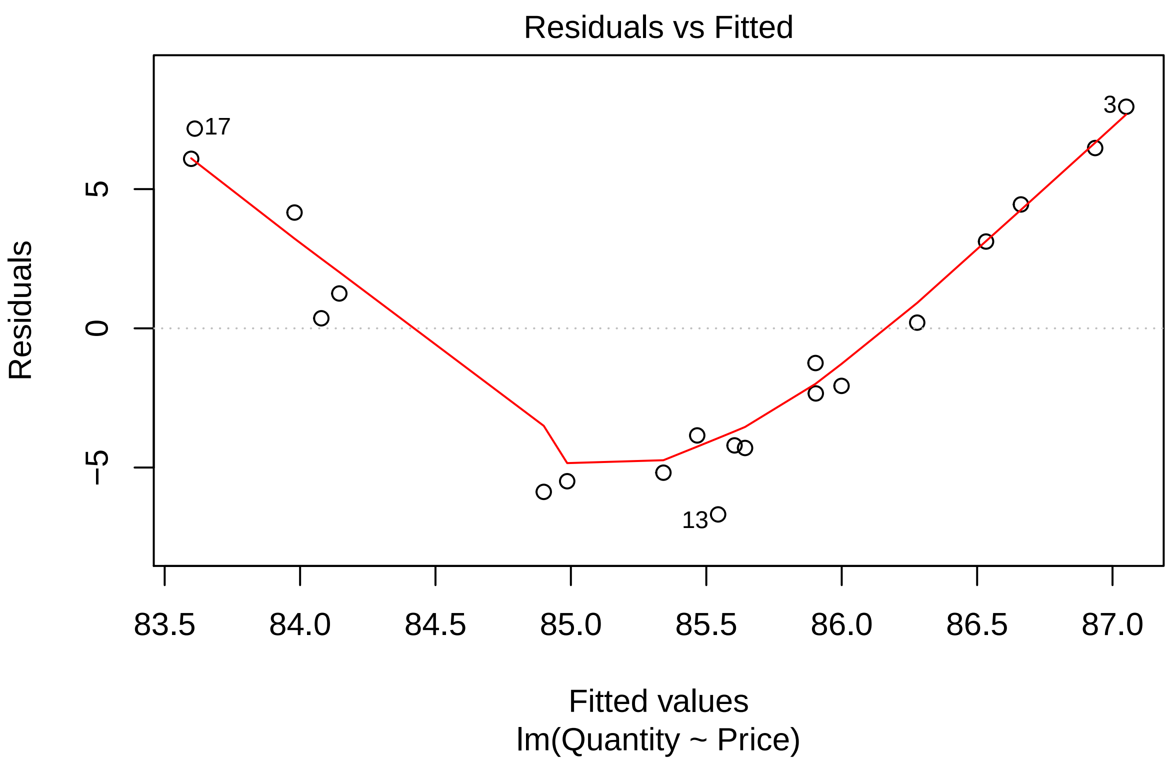
```
NotALine<- lm(Quantity ~ Price, data=MyData2)

summary(NotALine)
```

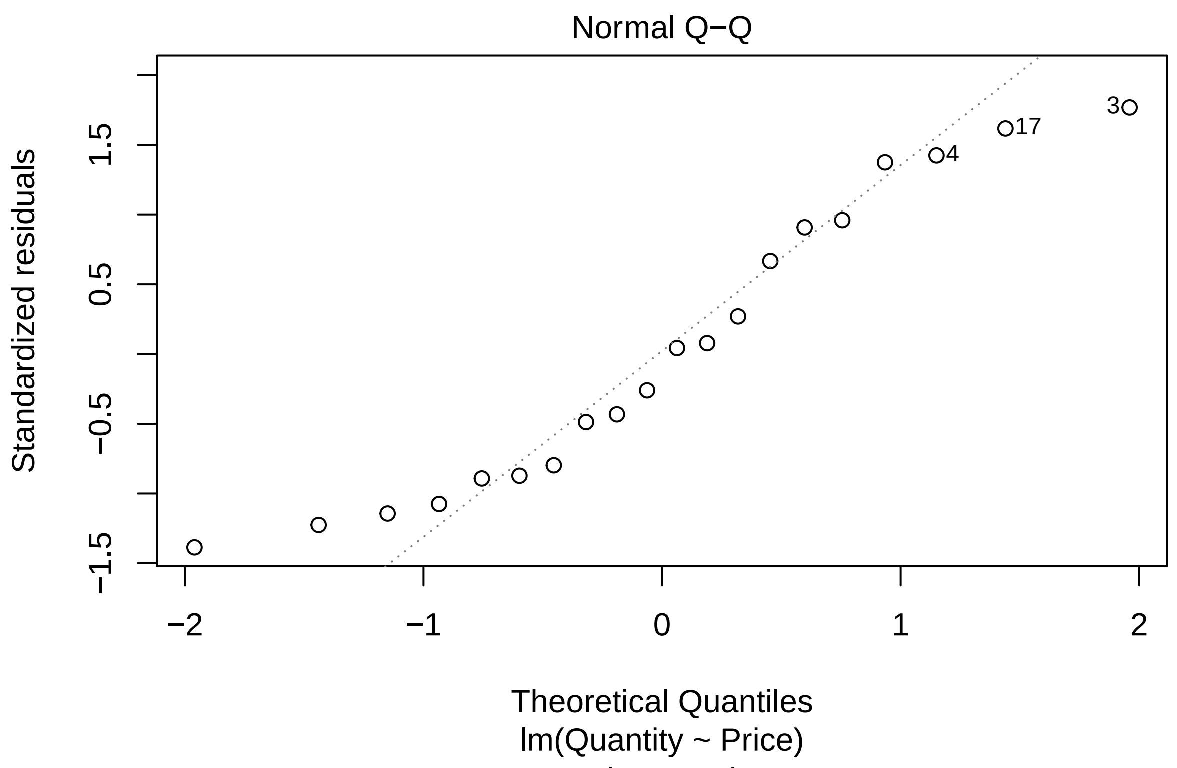
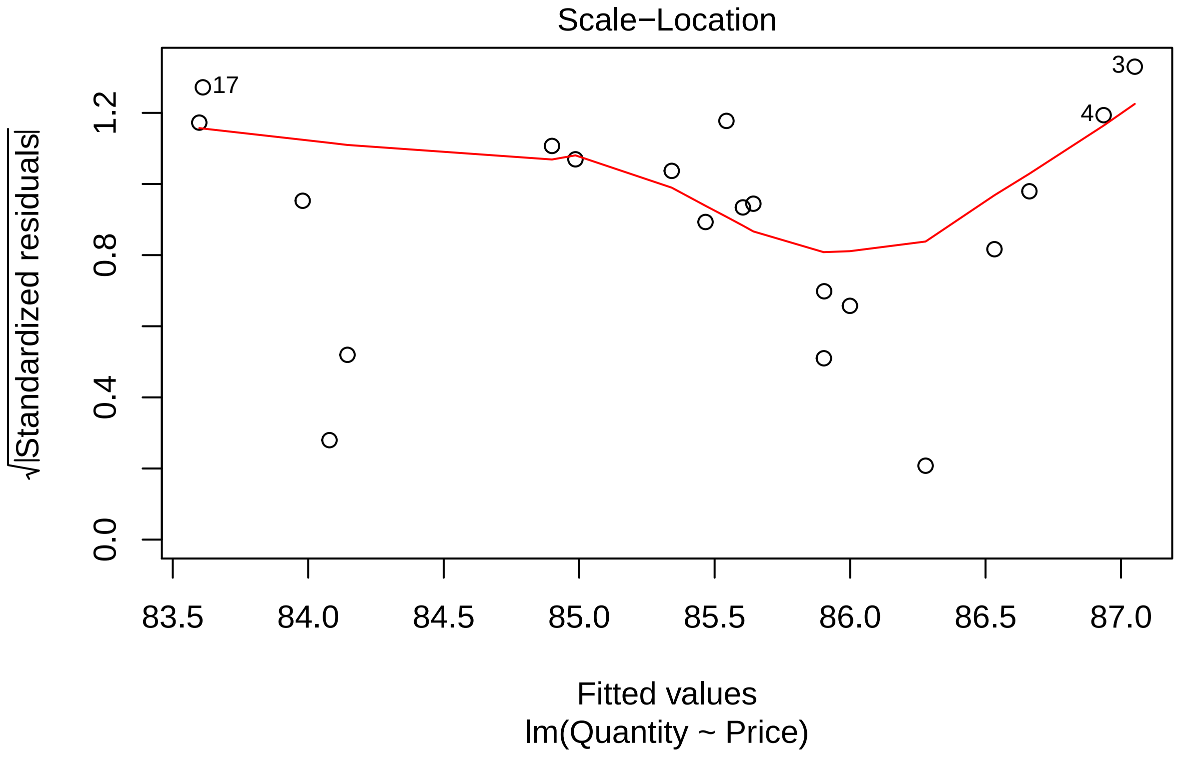
```
##
## Call:
## lm(formula = Quantity ~ Price, data = MyData2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.6832 -4.2283 -0.5211  4.2314  7.9584
```

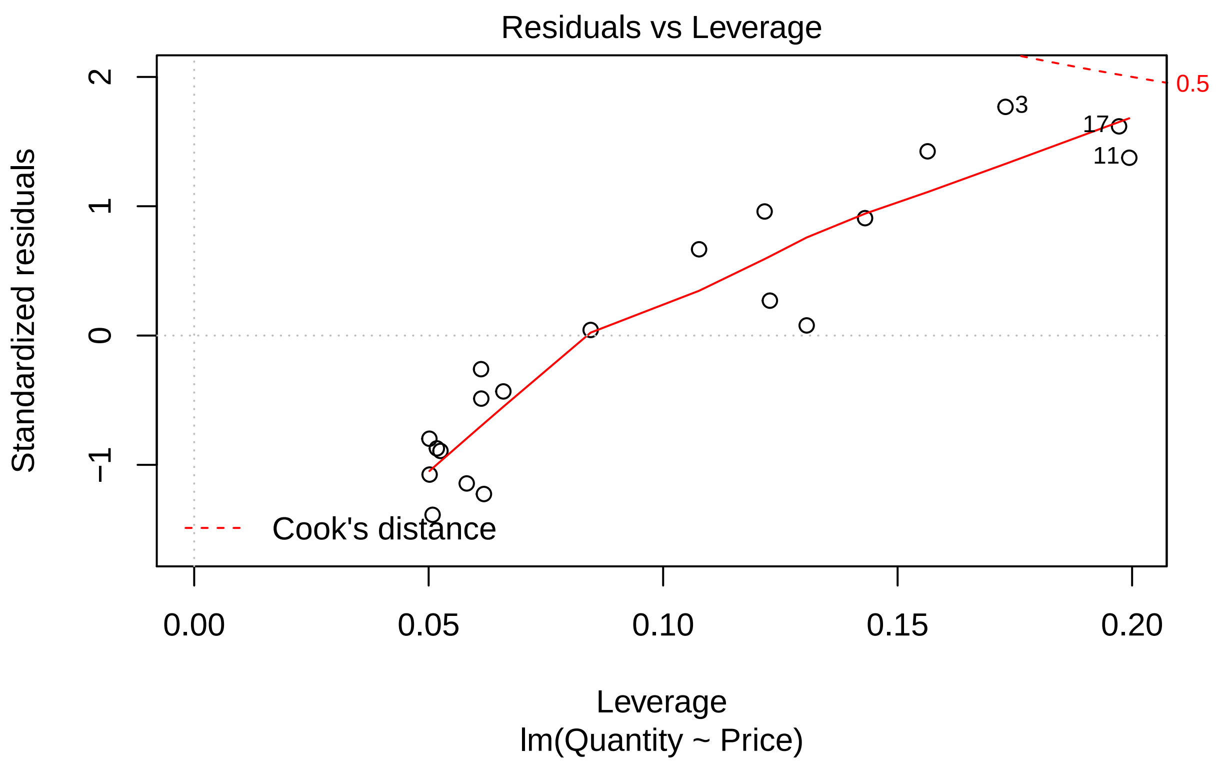
```
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  83.1110     2.6674  31.158   <2e-16 ***
## Price         0.1041     0.1100   0.946    0.356
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.948 on 18 degrees of freedom
## Multiple R-squared:  0.04741,    Adjusted R-squared:  -0.005514
## F-statistic: 0.8958 on 1 and 18 DF,  p-value: 0.3564
```

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

```
plot(NotALine)
```

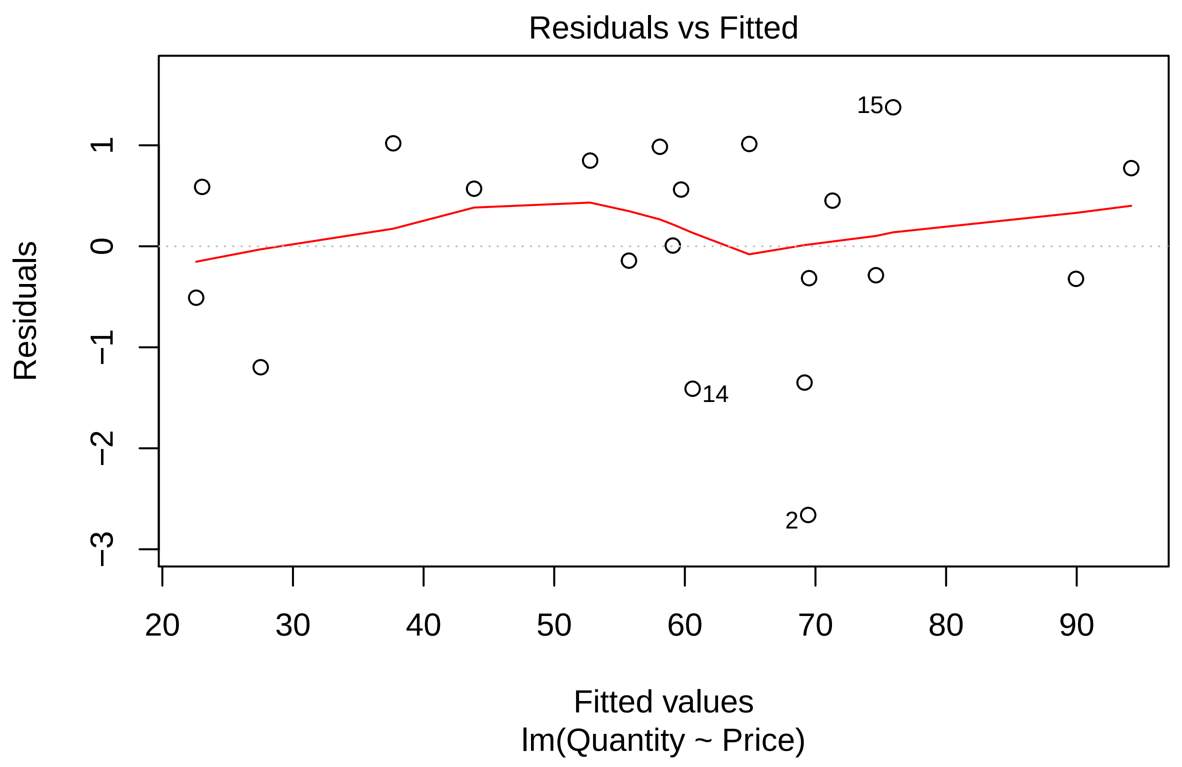


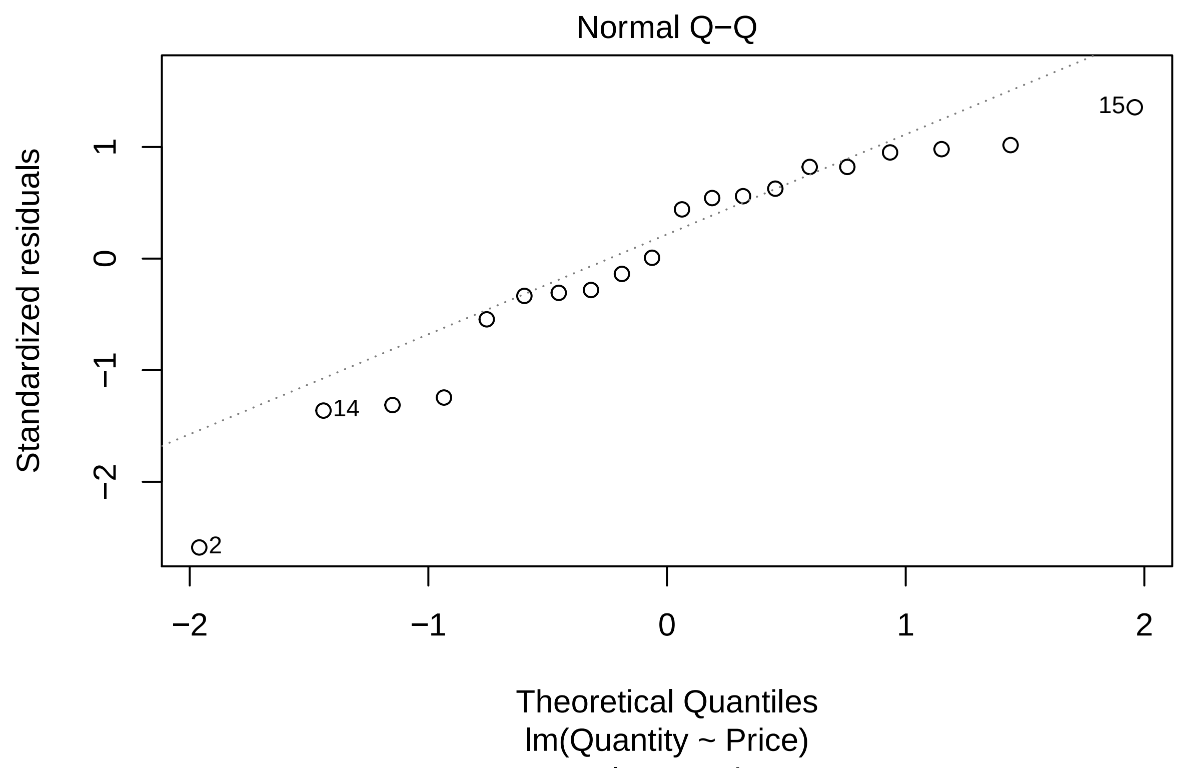

Normal Q−Q
Standardized residuals
−1.5
−0.5
0.5
1.5
−2
−1
0
1
2
Theoretical Quantiles
lm(Quantity ~ Price)
3
17
4

Scale−Location
√|Standardized residuals|
0.0
0.4
0.8
1.2
83.5
84.0
84.5
85.0
85.5
86.0
86.5
87.0
Fitted values
lm(Quantity ~ Price)
17
3
4

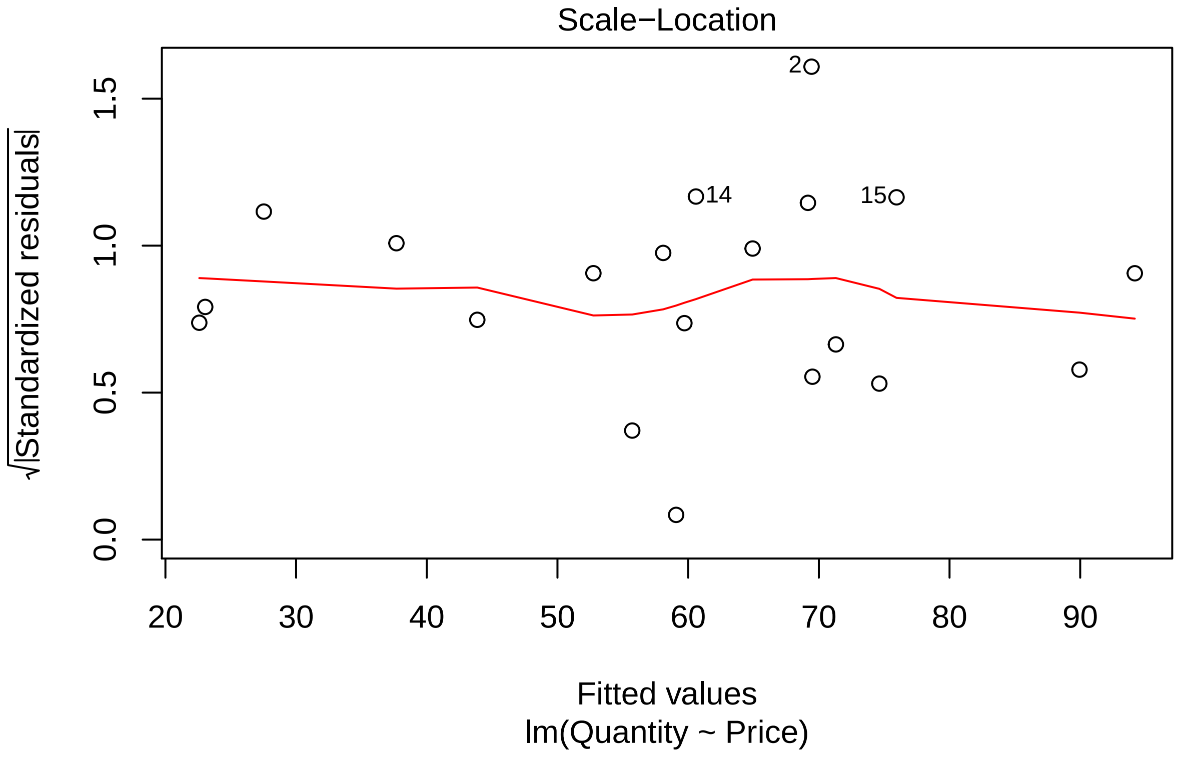


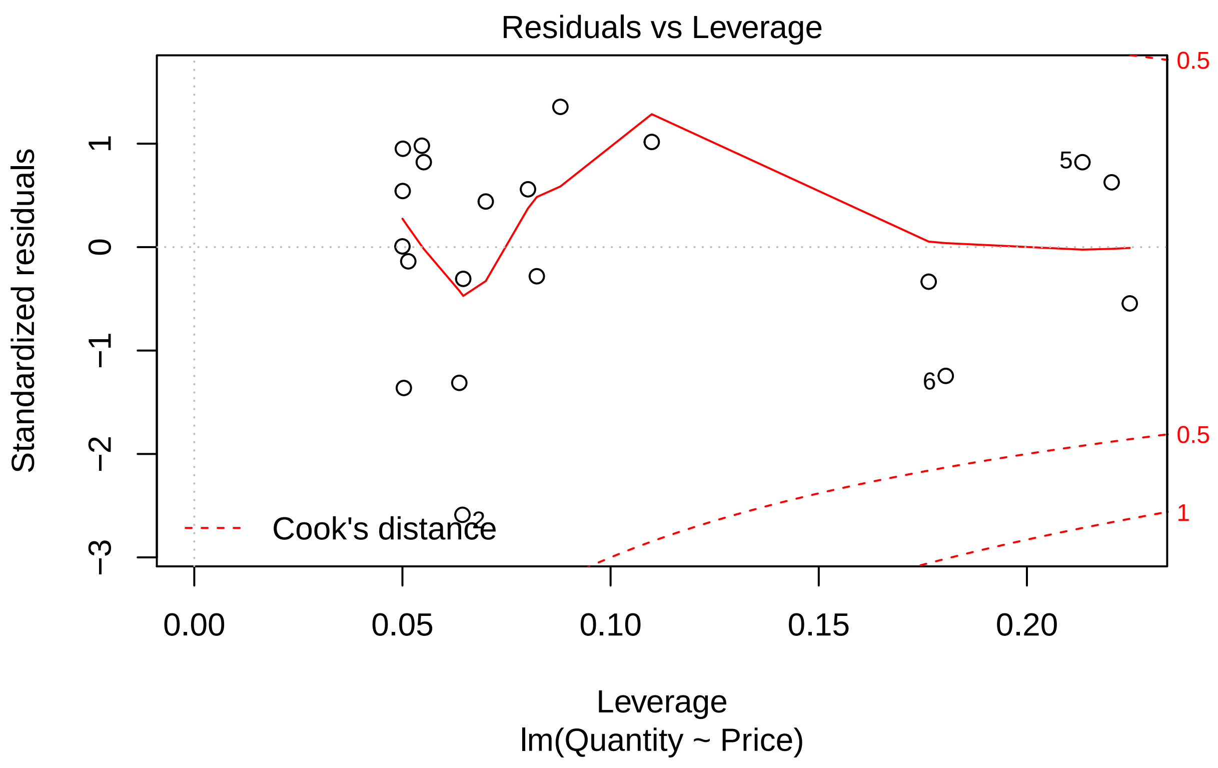
Compare with when it was a line and you estimated a line

```r
plot(FirstRegression)
```



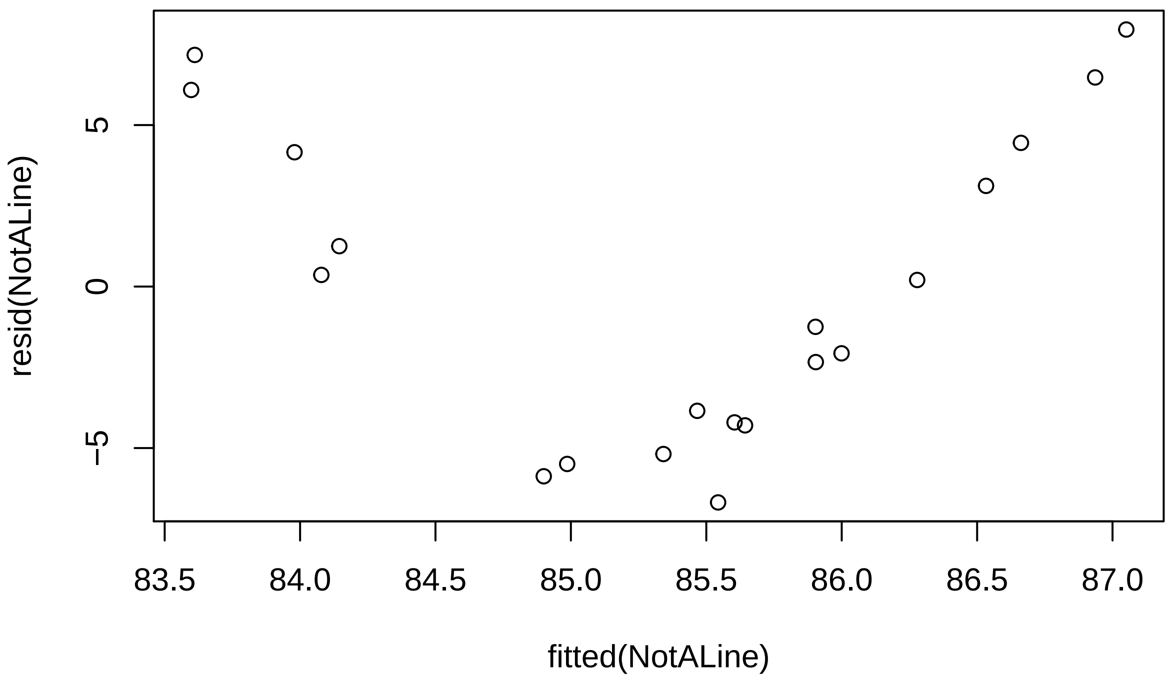

Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(Quantity ~ Price)
−2
−1
0
1
2
15
14
2


Scale−Location
√|Standardized residuals|
Fitted values
lm(Quantity ~ Price)
0.0
0.5
1.0
1.5
20
30
40
50
60
70
80
90
2
14
15



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

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



Dummy Variables

Not every variable is real valued, some are true false or categorical like night and day. R deals with categorical variables, called factors, very well and has fewer of the problems you see in other languages.

Lets make new fake data with dummy variables. I will reuse the first artificial dataset that was truly linear.

```r
MyData$Time <- as.factor(ifelse(runif(20) < .5, "Day", "Night"))

MyData$Quantity[MyData$Time == "Day"] <- MyData$Quantity[MyData$Time == "Day"] + 12
```

I just made it so that if it is Day the quantity is 12 higher.

```r
WTime <- lm(Quantity ~ Price + Time, data=MyData)

summary(WTime)
```

```
##
## Call:
## lm(formula = Quantity ~ Price + Time, data = MyData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.4040 -0.5153  0.3303  0.7278  1.6170
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 111.99808    0.59071  189.60  < 2e-16 ***
## Price        -2.01298    0.02509  -80.22  < 2e-16 ***
## TimeNight   -11.53157    0.49664  -23.22 2.58e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.066 on 17 degrees of freedom
## Multiple R-squared:  0.9979, Adjusted R-squared:  0.9976
## F-statistic:  4024 on 2 and 17 DF,  p-value: < 2.2e-16
```

See something odd. Interpreting this can be tricky. Best take some notes.

You can also make it so that the reaction to price is different depending on if it is night or day.

```r
WTimeInteract <- lm(Quantity ~ Price*Time, data=MyData)

summary(WTimeInteract)
```

```
##
## Call:
## lm(formula = Quantity ~ Price * Time, data = MyData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3925 -0.5374  0.3231  0.7385  1.6417
##
```

```
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)    
## (Intercept)     111.924370   0.850538 131.592  < 2e-16 ***
## Price            -2.008917   0.041747 -48.122  < 2e-16 ***
## TimeNight       -11.401928   1.163491  -9.800 3.64e-08 ***
## Price:TimeNight  -0.006596   0.053170  -0.124    0.903    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 1.098 on 16 degrees of freedom
## Multiple R-squared:  0.9979, Adjusted R-squared:  0.9975 
## F-statistic:  2527 on 3 and 16 DF,  p-value: < 2.2e-16
```

You will need a lot of notes on this. Interpreting interactive effects is tricky.

Useful facts

You can get elasticites directly out of a regression model if both quantity and price are logged. In every other case you have to calculate elasticities by hand and they are different at every point.

```
ElastForm <- lm(log(Quantity) ~ log(Price) + Time, data=MyData)

summary(ElastForm)
```

```
## 
## Call:
## lm(formula = log(Quantity) ~ log(Price) + Time, data = MyData)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -0.44104 -0.09647  0.09126  0.14624  0.19410 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  5.66148    0.23679  23.910 1.59e-14 ***
## log(Price)  -0.49932    0.08101  -6.163 1.04e-05 ***
## TimeNight   -0.23611    0.10150  -2.326   0.0326 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.2199 on 17 degrees of freedom
## Multiple R-squared:  0.7414, Adjusted R-squared:  0.711 
## F-statistic: 24.37 on 2 and 17 DF,  p-value: 1.018e-05
```

I did it, but is it right?

If only the right hand side variable is logged, you interpret the parameter as the effect of a percent change.

```
PercentForm <- lm(Quantity ~ log(Price) + Time, data=MyData)

summary(PercentForm)
```

```
##
## Call:
## lm(formula = Quantity ~ log(Price) + Time, data = MyData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.503  -3.951   3.347   5.896   7.604
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  155.609      8.765  17.753 2.08e-12 ***
## log(Price)   -29.015      2.999  -9.675 2.51e-08 ***
## TimeNight    -14.136      3.757  -3.762  0.00155 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.141 on 17 degrees of freedom
## Multiple R-squared:  0.877,  Adjusted R-squared:  0.8626
## F-statistic: 60.63 on 2 and 17 DF,  p-value: 1.833e-08
```

Common statistical problems

This is the basics. You now know enough to be insanely overconfident. Don't destroy the world.

Beyond getting the specification wrong, which you usually support with several similar models that show similar results, there are common statistical problems that you learn about in econometrics

- Autocorrelation, which makes your regression look better than it actually is.
- Heterskadasticity, which makes it worse.
- Endogenaity, which biases your parameters in unknown directions.
- Errors in variables (RHS), which bias parameters towards zero.
- …

Don't get cocky.

What we are going to do next is read in some real data and you will for homework estimate something that looks like either supply or demand but will be neither. Why, because supply and demand determine price and quantity simultaneously, that means that something either price or quantity is endogenous. At this point, you are at about 1910 as far as standard econometric practice.

Real Data

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

```r
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.

- 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).

- 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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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
```

- 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.

```
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
```

```
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.

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

- And simplify the names.

```
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 (MMBTU), which is a measure of energy content. The BTUs per short ton of coal (2000 lbs) is about 20 MMBTUs but varies from place-to-place, year-to-year and type of coal (See table 3-4).

- 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)
```

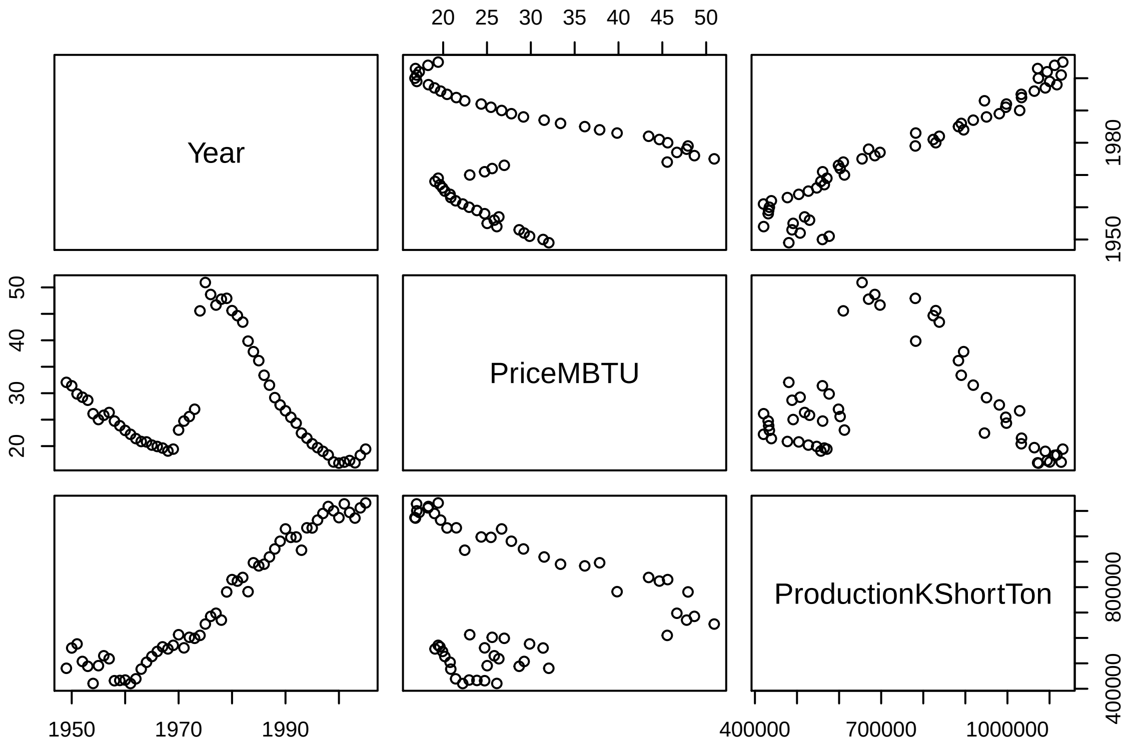
```
##       Year         PriceMBTU    
##  Min.   :1949   Min.   :16.78  
##  1st Qu.:1963   1st Qu.:20.19  
##  Median :1977   Median :25.02  
##  Mean   :1977   Mean   :27.85  
##  3rd Qu.:1991   3rd Qu.:31.52  
##  Max.   :2005   Max.   :50.92  
```

```r
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    
```

Lets look at the data.

```r
pairs(~ Year+ PriceMBTU + ProductionKShortTon, data = CoalMarket)
```


20
25
30
35
40
45
50
Year
1950
1980
PriceMBTU
20
30
40
50
ProductionKShortTon
400000
800000
1950
1970
1990
400000
700000
1000000