

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
```

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

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

```
A + B
```

```
## [1] 18
```

```
C = A + B
```

```
C
```

```
## [1] 18
```

```
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.

```
ThisIsAVariable <-10
```

```
this_is_a_variable <-23
```

2. R has many ways of representing values

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

```
A
```

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

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

```
C
```

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

```
C[[1]]
```

```
## [1] "Tom"
```

```
C[1]
```

```
## [1] "Tom"
```

```
1:1000
```

##	[1]	1	2	3	4	5	6	7	8	9	10	11	12	13
##	[14]	14	15	16	17	18	19	20	21	22	23	24	25	26
##	[27]	27	28	29	30	31	32	33	34	35	36	37	38	39
##	[40]	40	41	42	43	44	45	46	47	48	49	50	51	52
##	[53]	53	54	55	56	57	58	59	60	61	62	63	64	65
##	[66]	66	67	68	69	70	71	72	73	74	75	76	77	78
##	[79]	79	80	81	82	83	84	85	86	87	88	89	90	91
##	[92]	92	93	94	95	96	97	98	99	100	101	102	103	104
##	[105]	105	106	107	108	109	110	111	112	113	114	115	116	117
##	[118]	118	119	120	121	122	123	124	125	126	127	128	129	130
##	[131]	131	132	133	134	135	136	137	138	139	140	141	142	143
##	[144]	144	145	146	147	148	149	150	151	152	153	154	155	156
##	[157]	157	158	159	160	161	162	163	164	165	166	167	168	169
##	[170]	170	171	172	173	174	175	176	177	178	179	180	181	182
##	[183]	183	184	185	186	187	188	189	190	191	192	193	194	195
##	[196]	196	197	198	199	200	201	202	203	204	205	206	207	208
##	[209]	209	210	211	212	213	214	215	216	217	218	219	220	221
##	[222]	222	223	224	225	226	227	228	229	230	231	232	233	234
##	[235]	235	236	237	238	239	240	241	242	243	244	245	246	247
##	[248]	248	249	250	251	252	253	254	255	256	257	258	259	260
##	[261]	261	262	263	264	265	266	267	268	269	270	271	272	273
##	[274]	274	275	276	277	278	279	280	281	282	283	284	285	286
##	[287]	287	288	289	290	291	292	293	294	295	296	297	298	299
##	[300]	300	301	302	303	304	305	306	307	308	309	310	311	312
##	[313]	313	314	315	316	317	318	319	320	321	322	323	324	325
##	[326]	326	327	328	329	330	331	332	333	334	335	336	337	338
##	[339]	339	340	341	342	343	344	345	346	347	348	349	350	351
##	[352]	352	353	354	355	356	357	358	359	360	361	362	363	364
##	[365]	365	366	367	368	369	370	371	372	373	374	375	376	377
##	[378]	378	379	380	381	382	383	384	385	386	387	388	389	390
##	[391]	391	392	393	394	395	396	397	398	399	400	401	402	403
##	[404]	404	405	406	407	408	409	410	411	412	413	414	415	416
##	[417]	417	418	419	420	421	422	423	424	425	426	427	428	429
##	[430]	430	431	432	433	434	435	436	437	438	439	440	441	442
##	[443]	443	444	445	446	447	448	449	450	451	452	453	454	455
##	[456]	456	457	458	459	460	461	462	463	464	465	466	467	468
##	[469]	469	470	471	472	473	474	475	476	477	478	479	480	481
##	[482]	482	483	484	485	486	487	488	489	490	491	492	493	494
##	[495]	495	496	497	498	499	500	501	502	503	504	505	506	507
##	[508]	508	509	510	511	512	513	514	515	516	517	518	519	520
##	[521]	521	522	523	524	525	526	527	528	529	530	531	532	533
##	[534]	534	535	536	537	538	539	540	541	542	543	544	545	546
##	[547]	547	548	549	550	551	552	553	554	555	556	557	558	559
##	[560]	560	561	562	563	564	565	566	567	568	569	570	571	572
##	[573]	573	574	575	576	577	578	579	580	581	582	583	584	585

```
## [586] 586 587 588 589 590 591 592 593 594 595 596 597 598
## [599] 599 600 601 602 603 604 605 606 607 608 609 610 611
## [612] 612 613 614 615 616 617 618 619 620 621 622 623 624
## [625] 625 626 627 628 629 630 631 632 633 634 635 636 637
## [638] 638 639 640 641 642 643 644 645 646 647 648 649 650
## [651] 651 652 653 654 655 656 657 658 659 660 661 662 663
## [664] 664 665 666 667 668 669 670 671 672 673 674 675 676
## [677] 677 678 679 680 681 682 683 684 685 686 687 688 689
## [690] 690 691 692 693 694 695 696 697 698 699 700 701 702
## [703] 703 704 705 706 707 708 709 710 711 712 713 714 715
## [716] 716 717 718 719 720 721 722 723 724 725 726 727 728
## [729] 729 730 731 732 733 734 735 736 737 738 739 740 741
## [742] 742 743 744 745 746 747 748 749 750 751 752 753 754
## [755] 755 756 757 758 759 760 761 762 763 764 765 766 767
## [768] 768 769 770 771 772 773 774 775 776 777 778 779 780
## [781] 781 782 783 784 785 786 787 788 789 790 791 792 793
## [794] 794 795 796 797 798 799 800 801 802 803 804 805 806
## [807] 807 808 809 810 811 812 813 814 815 816 817 818 819
## [820] 820 821 822 823 824 825 826 827 828 829 830 831 832
## [833] 833 834 835 836 837 838 839 840 841 842 843 844 845
## [846] 846 847 848 849 850 851 852 853 854 855 856 857 858
## [859] 859 860 861 862 863 864 865 866 867 868 869 870 871
## [872] 872 873 874 875 876 877 878 879 880 881 882 883 884
## [885] 885 886 887 888 889 890 891 892 893 894 895 896 897
## [898] 898 899 900 901 902 903 904 905 906 907 908 909 910
## [911] 911 912 913 914 915 916 917 918 919 920 921 922 923
## [924] 924 925 926 927 928 929 930 931 932 933 934 935 936
## [937] 937 938 939 940 941 942 943 944 945 946 947 948 949
## [950] 950 951 952 953 954 955 956 957 958 959 960 961 962
## [963] 963 964 965 966 967 968 969 970 971 972 973 974 975
## [976] 976 977 978 979 980 981 982 983 984 985 986 987 988
## [989] 989 990 991 992 993 994 995 996 997 998 999 1000
```

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

```
D
```

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

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

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

```
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
```

```
summary(D)
```

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

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

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

```
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.

```
sqrt(D)
```

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

```
#sqrt(A)
```

```
sqrt(D[6])
```

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

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

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

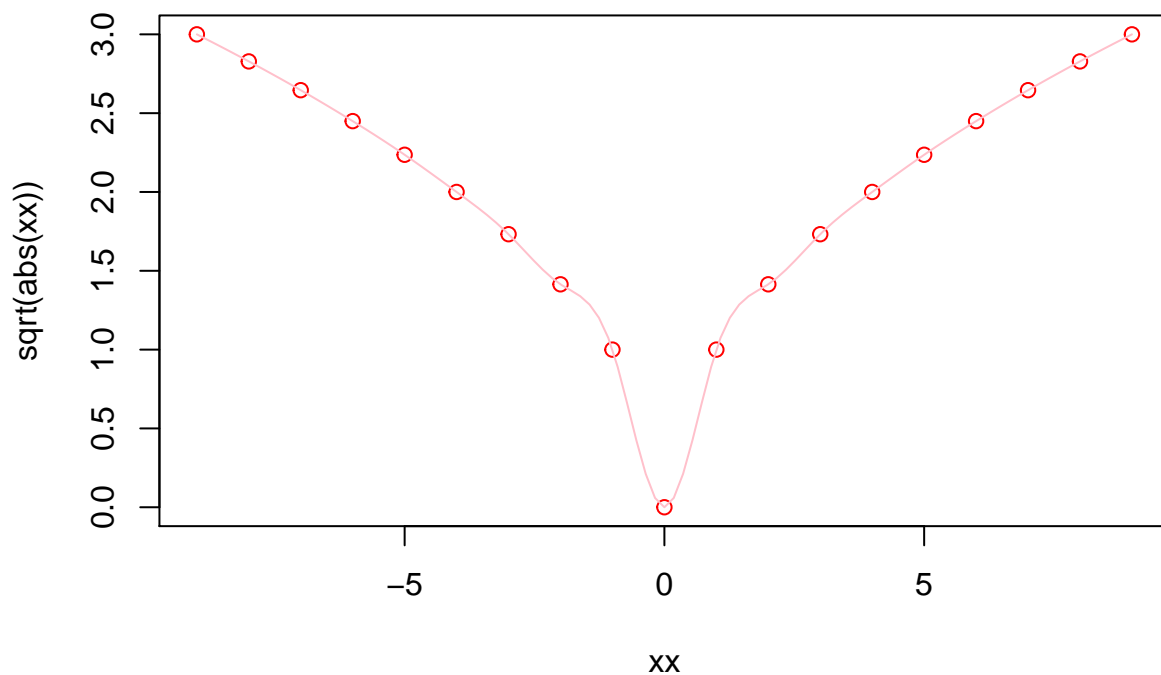
4. If you need help, use the help pane or ask for help. Everything has a help file and many functions come with examples of how to use.

```
help(sqrt)
```

```
## starting httpd help server ... done
```

```
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.   : 5.001   Min.    :28.49
##  1st Qu.:12.673   1st Qu.:56.89
##  Median :18.166   Median :63.48
##  Mean   :17.838   Mean    :64.70
##  3rd Qu.:21.569   3rd Qu.:74.22
##  Max.   :35.864   Max.    :89.66
```

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

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

```
##      Price Quantity
## 1  12.398847 76.16359
## 2  26.563382 47.88302
## 3  35.863552 28.48715
## 4  14.144763 71.81721
## 5  17.721183 64.54552
## 6  13.138583 73.57231
## 7  26.887124 46.59248
## 8  12.764642 73.31685
## 9   8.801452 83.61818
## 10 18.611558 62.13668
## 11 10.937182 80.20880
## 12 26.261519 47.97472
## 13 25.230284 50.63122
## 14 18.714767 62.38143
## 15 20.348490 58.97369
## 16 15.220098 69.60273
## 17 19.543000 60.02417
## 18 19.856604 62.40889
## 19  5.001052 89.66214
## 20  8.750835 84.02796
```

```
MyData[1]
```

```
##      Price
## 1  12.398847
## 2  26.563382
## 3  35.863552
## 4  14.144763
## 5  17.721183
## 6  13.138583
## 7  26.887124
## 8  12.764642
## 9   8.801452
## 10 18.611558
## 11 10.937182
## 12 26.261519
## 13 25.230284
## 14 18.714767
## 15 20.348490
## 16 15.220098
## 17 19.543000
## 18 19.856604
## 19  5.001052
## 20  8.750835
```

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

```
##      Price Quantity
## 1 12.39885 76.16359
## 2 26.56338 47.88302
## 3 35.86355 28.48715
```

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

```
## [1] 12.39885 26.56338 35.86355
```

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

```
##      Price
## 1 12.398847
## 2 26.563382
## 3 35.863552
## 4 14.144763
## 5 17.721183
## 6 13.138583
## 7 26.887124
## 8 12.764642
## 9  8.801452
## 10 18.611558
## 11 10.937182
## 12 26.261519
## 13 25.230284
## 14 18.714767
## 15 20.348490
## 16 15.220098
## 17 19.543000
## 18 19.856604
## 19  5.001052
## 20  8.750835
```

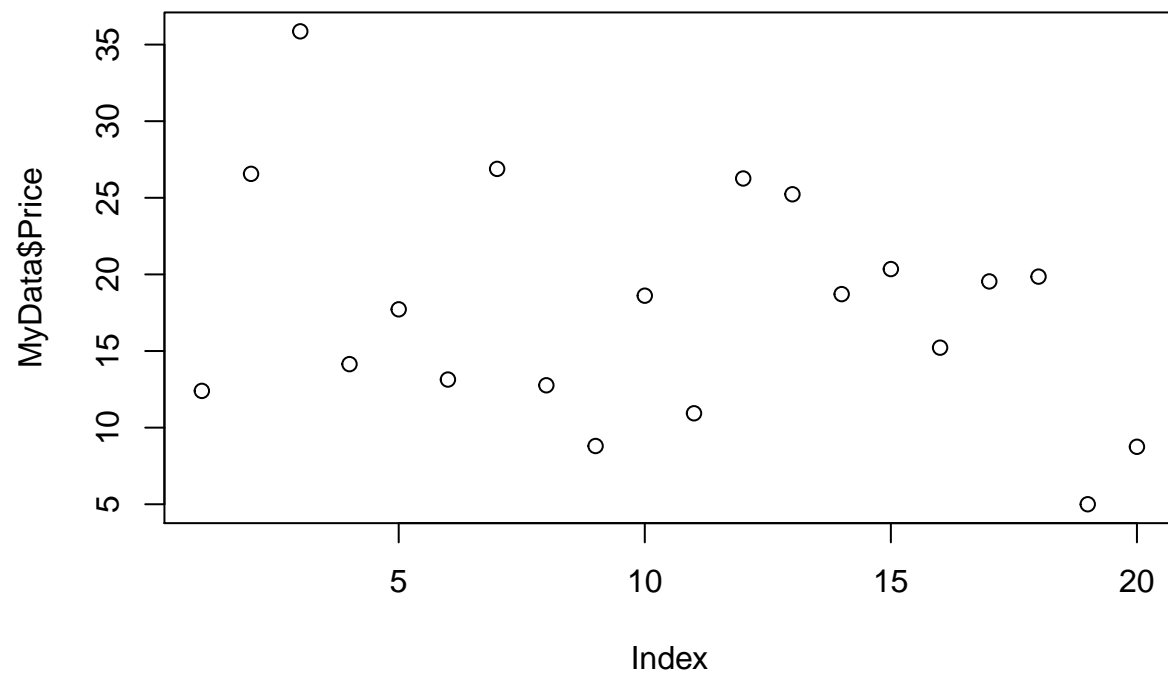
```
MyData[c(1,5,8),]
```

```
##      Price Quantity
## 1 12.39885 76.16359
## 5 17.72118 64.54552
## 8 12.76464 73.31685
```

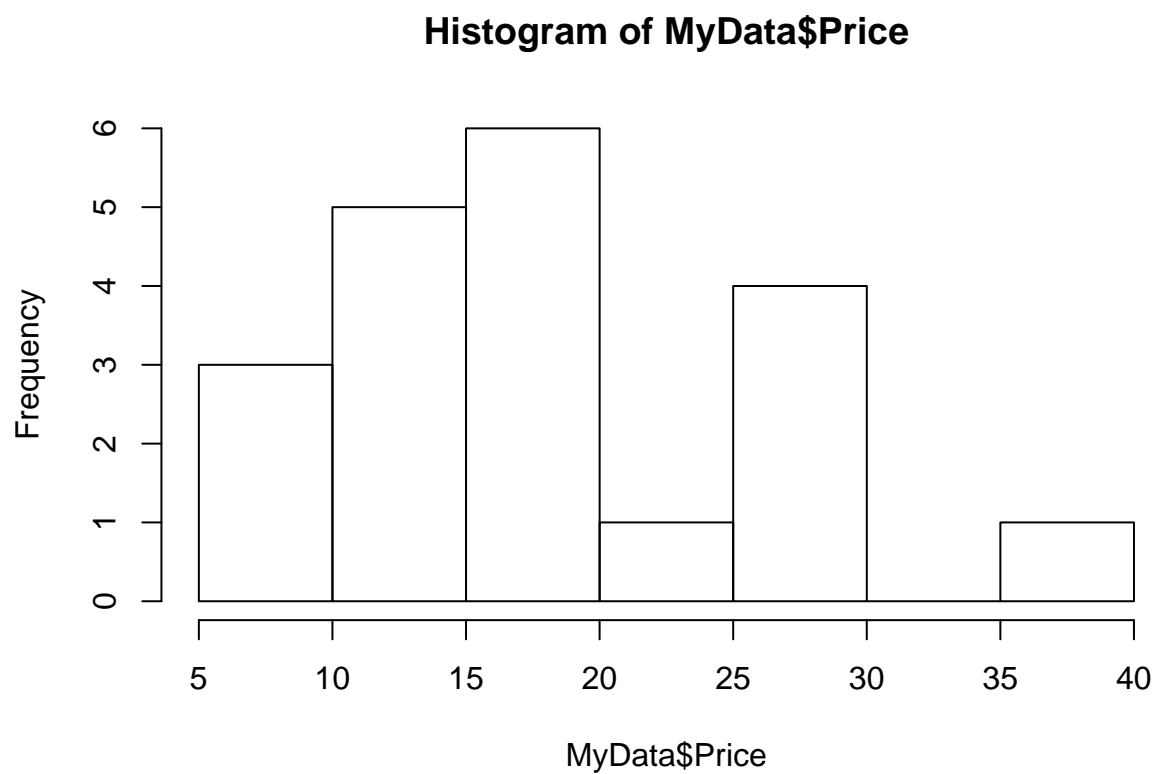
Pictures

Now lets make pictures.

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

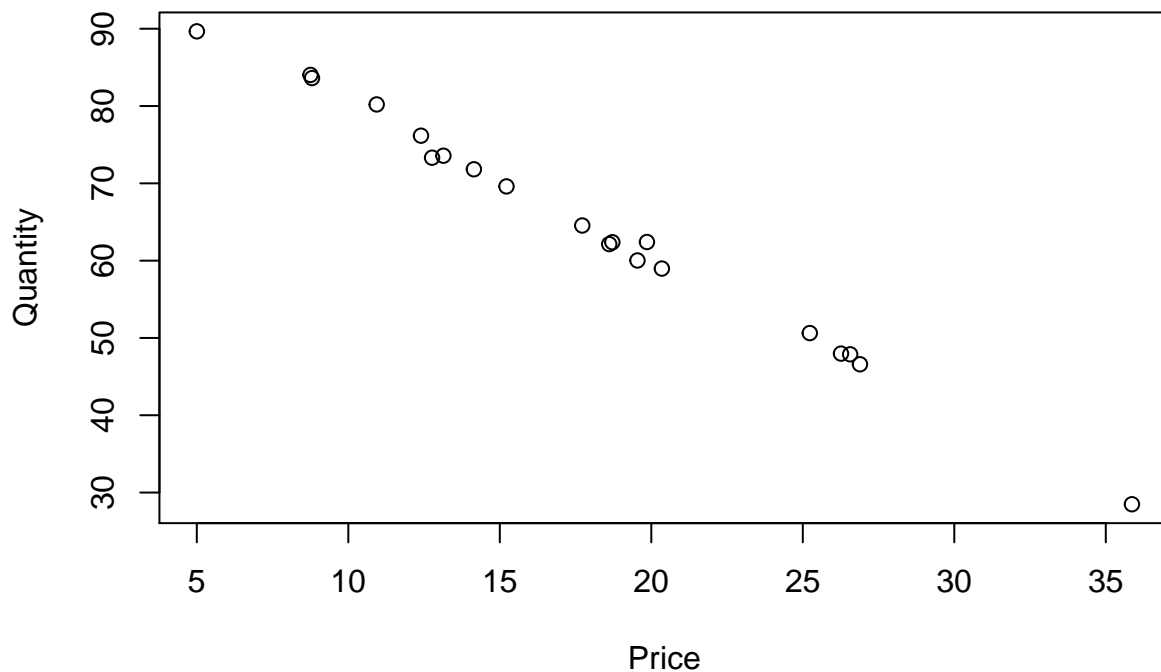


If you don't want to see the code and just the result, make sure to add `echo=false` to the chunk. Here is a Histogram without the code.



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

```
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
	-1.5526	-0.5960	-0.1865	0.6884	1.7533

```
##
## Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	100.45269	0.55300	181.7	<2e-16 ***
Price	-2.00422	0.02863	-70.0	<2e-16 ***

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.9486 on 18 degrees of freedom
## Multiple R-squared:  0.9963, Adjusted R-squared:  0.9961
## F-statistic: 4901 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

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

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, heteroskedasticity and endogeneity 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.

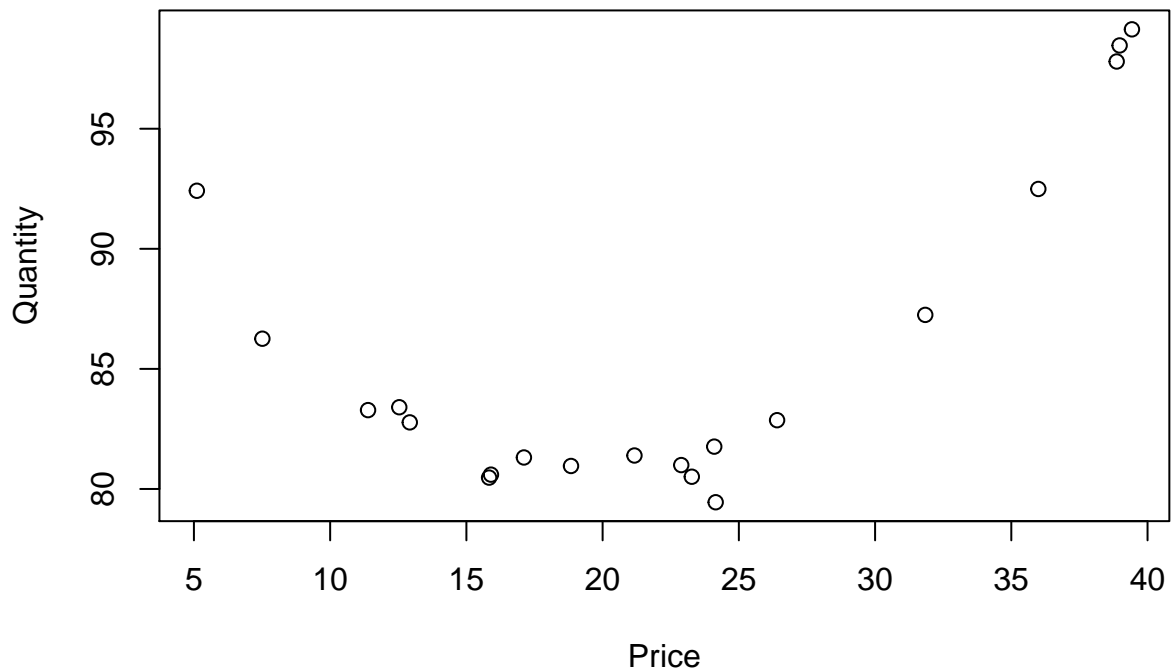
```
MyData2 <- data.frame(Price = runif(20, 2, 40))
MyData2$Quantity <- 100 - 2 * MyData2$Price + .05 * MyData2$Price^2 + rnorm(20, mean = 0, sd = 1)

summary(MyData2)
```

```
##      Price      Quantity
## Min.   : 5.107   Min.   :79.45
## 1st Qu.:15.106   1st Qu.:80.99
## Median :22.027   Median :82.82
## Mean   :22.212   Mean    :85.68
## 3rd Qu.:27.763   3rd Qu.:88.54
## Max.   :39.428   Max.    :99.14
```

Not a line

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



Give me a line anyway.

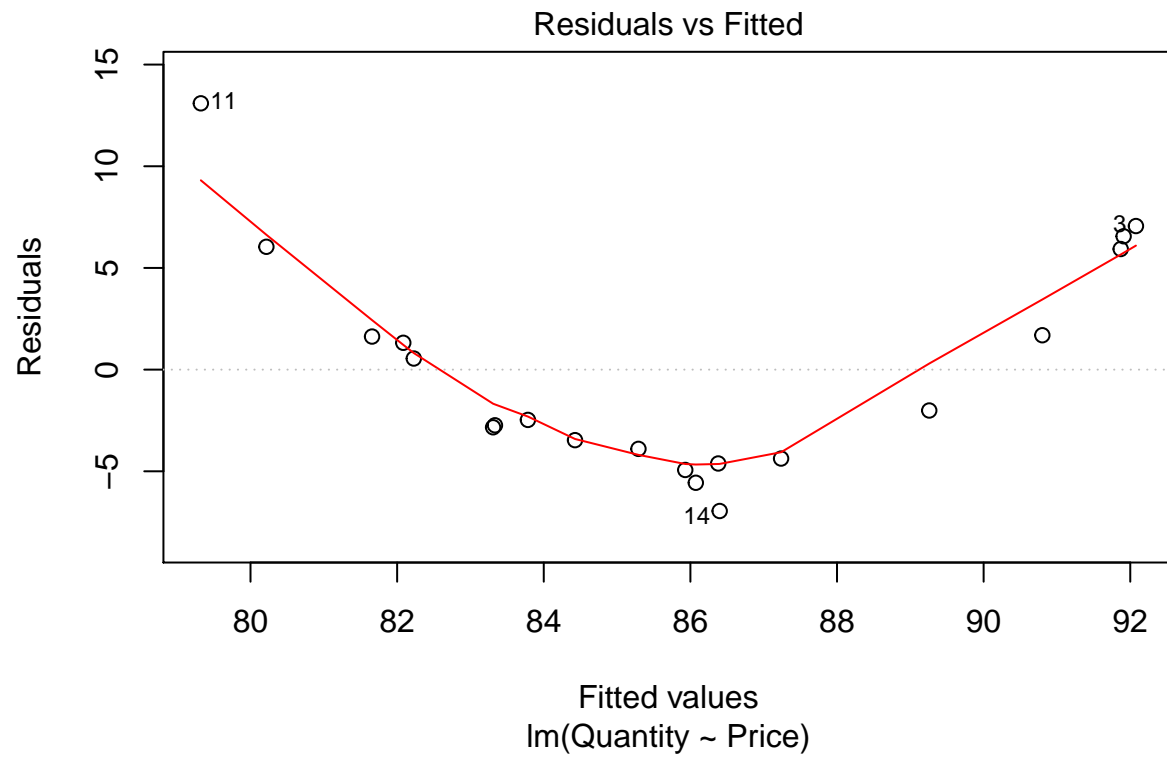
```
NotALine<- lm(Quantity ~ Price, data=MyData2)
```

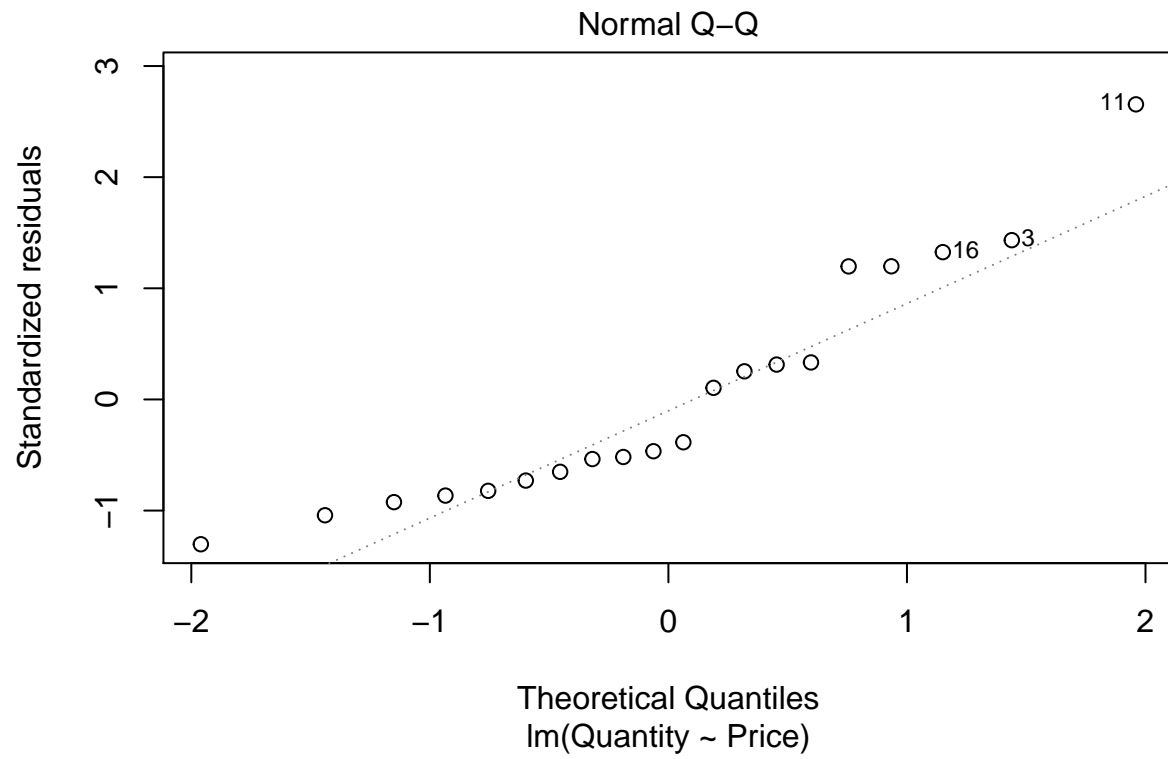
```
summary(NotALine)
```

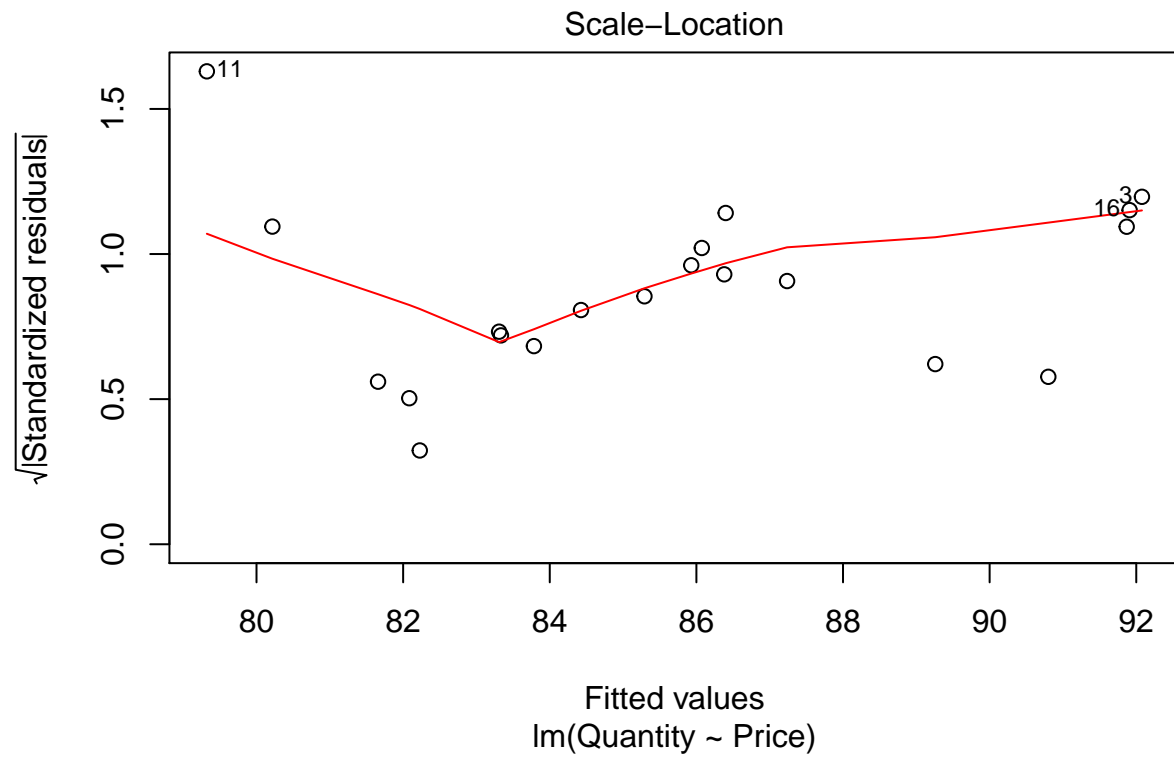
```
##
## Call:
## lm(formula = Quantity ~ Price, data = MyData2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -6.952 -4.016 -2.241  2.750 13.094
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  77.4245     2.9316   26.41 7.57e-16 ***
## Price         0.3716     0.1199    3.10 0.00619 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.48 on 18 degrees of freedom
## Multiple R-squared:  0.348, Adjusted R-squared:  0.3118
## F-statistic: 9.607 on 1 and 18 DF, p-value: 0.006185
```

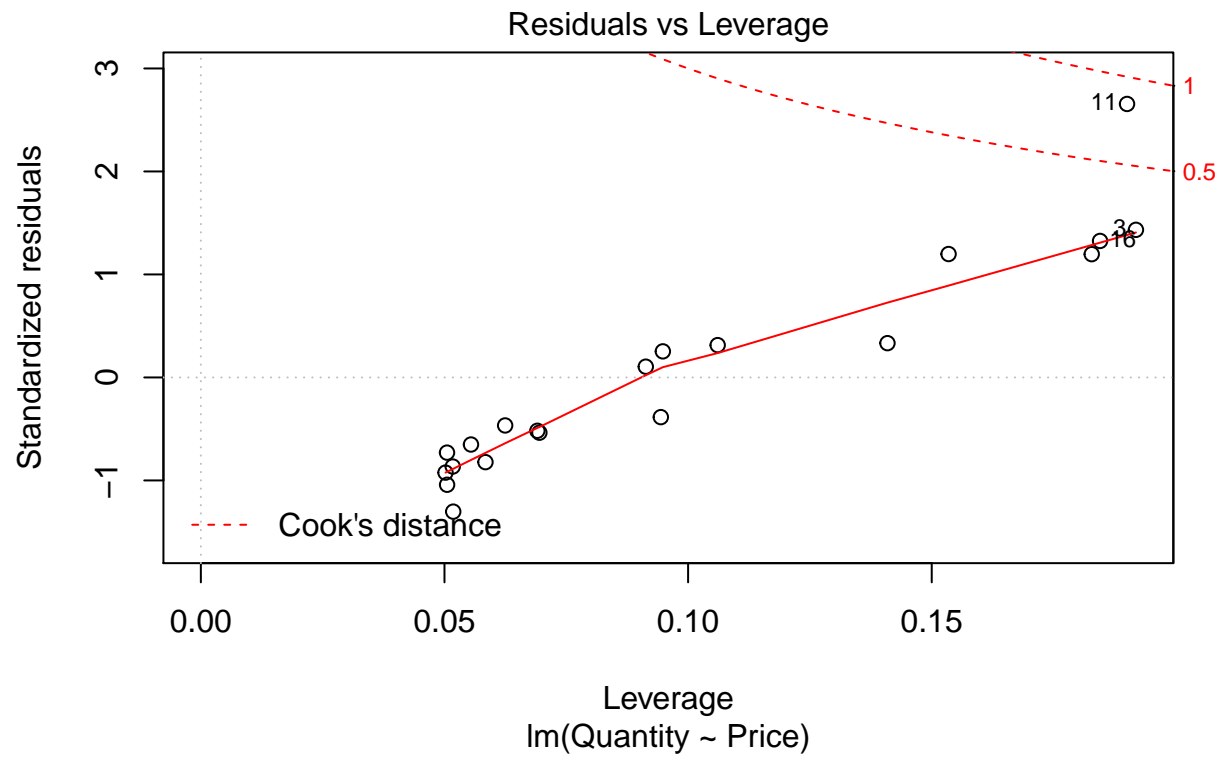
How to tell you were wrong thinking it was a line. Make a pictures

```
plot(NotALine)
```



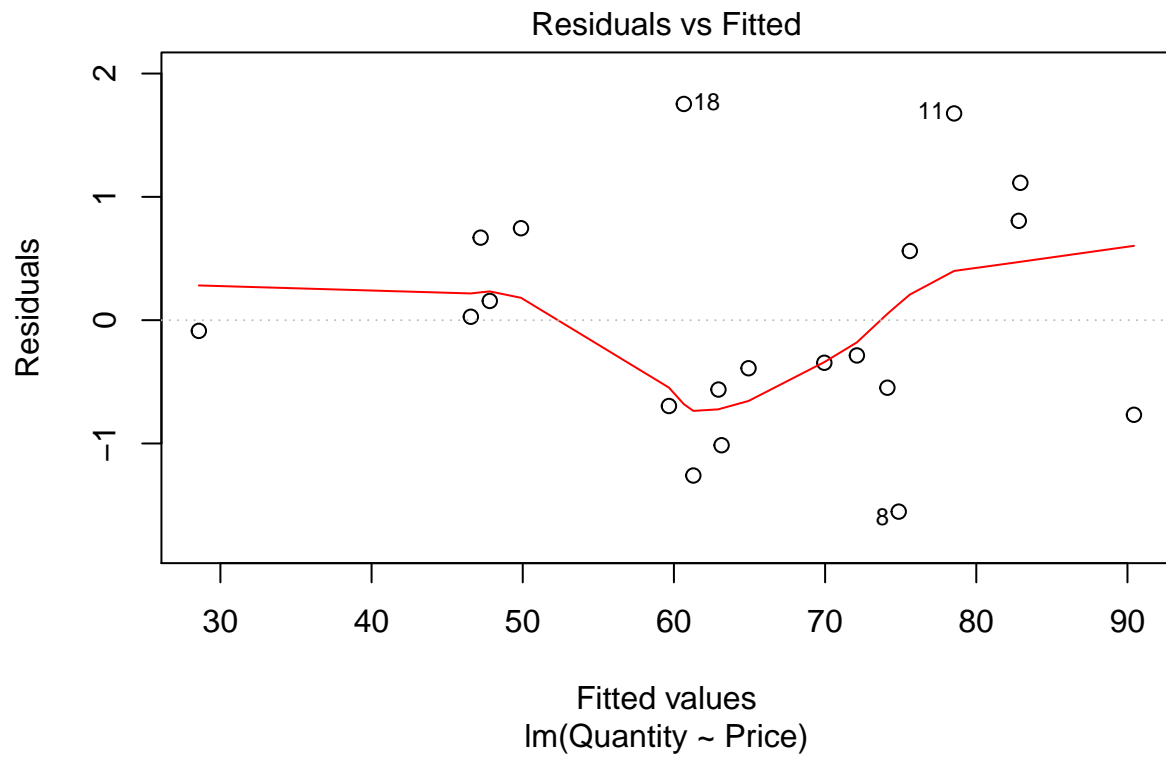


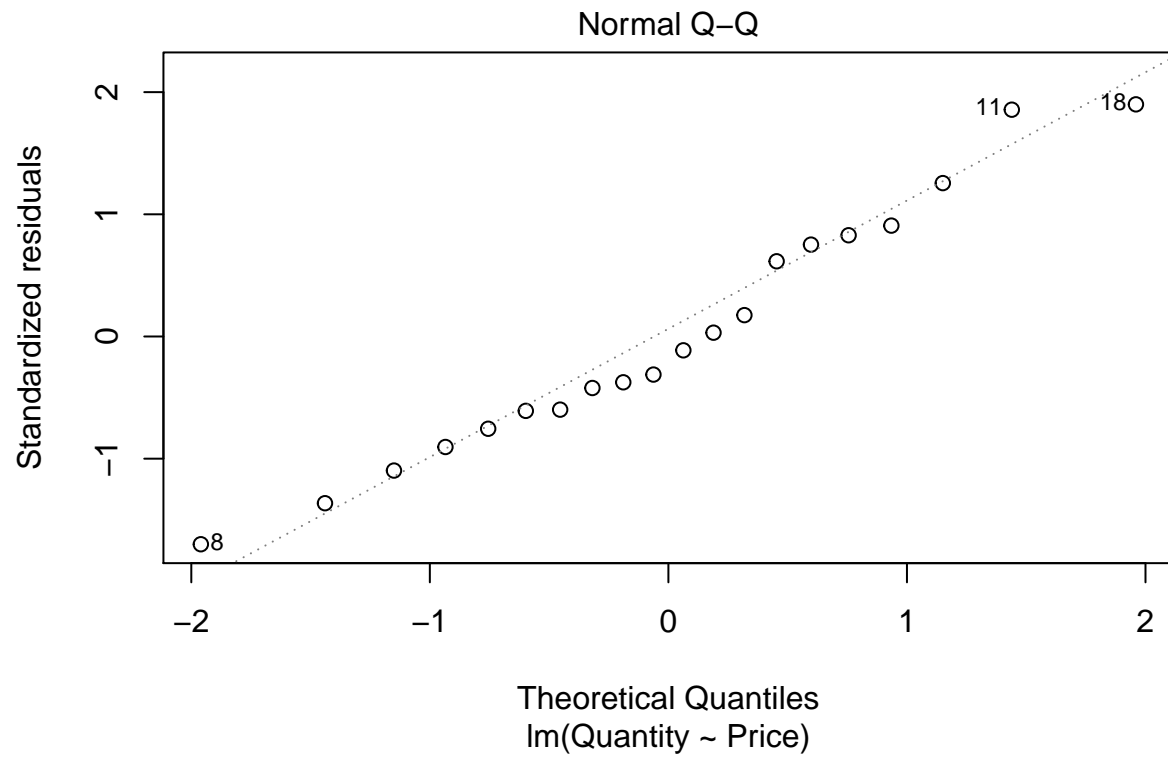


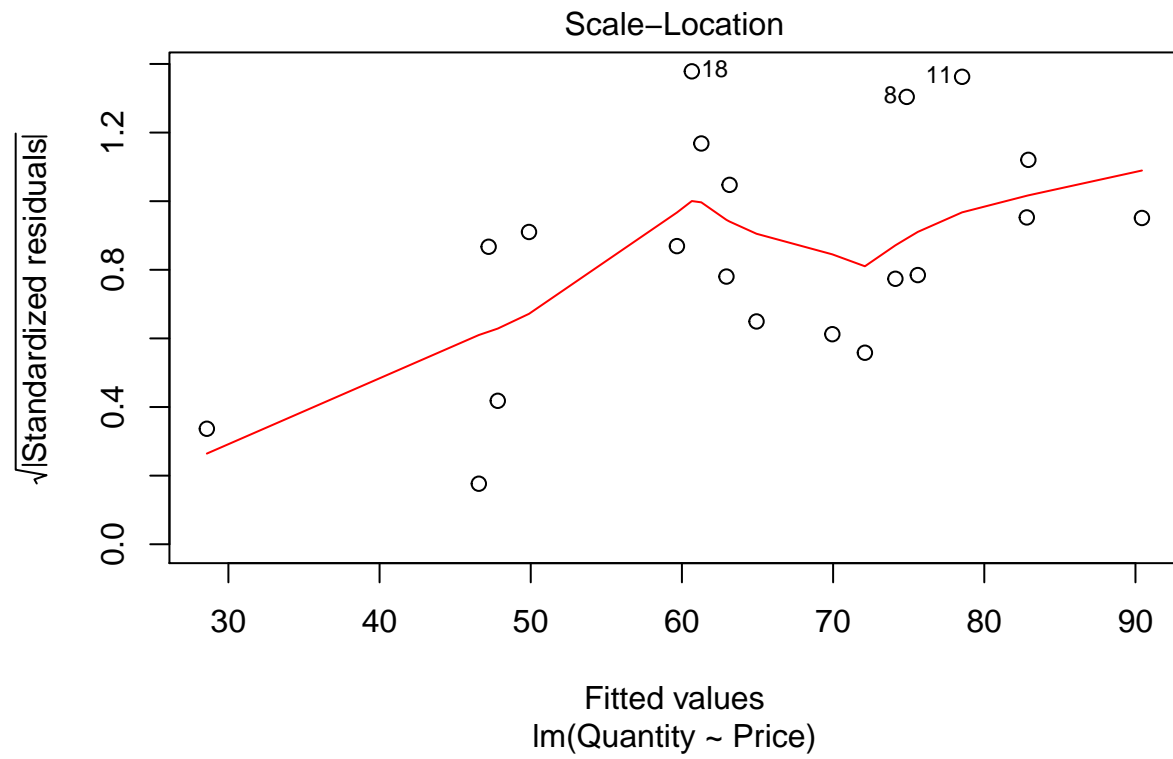


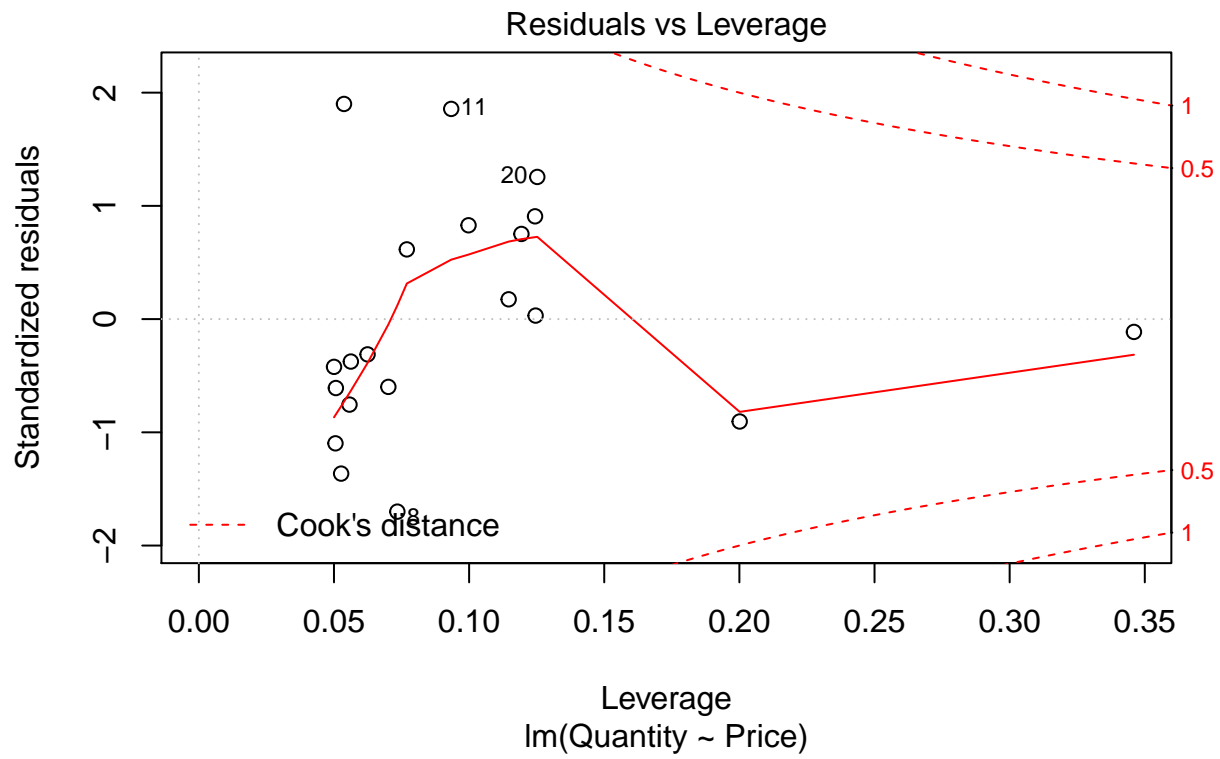
Compare with when it was a line and you estimated a line

```
plot(FirstRegression)
```

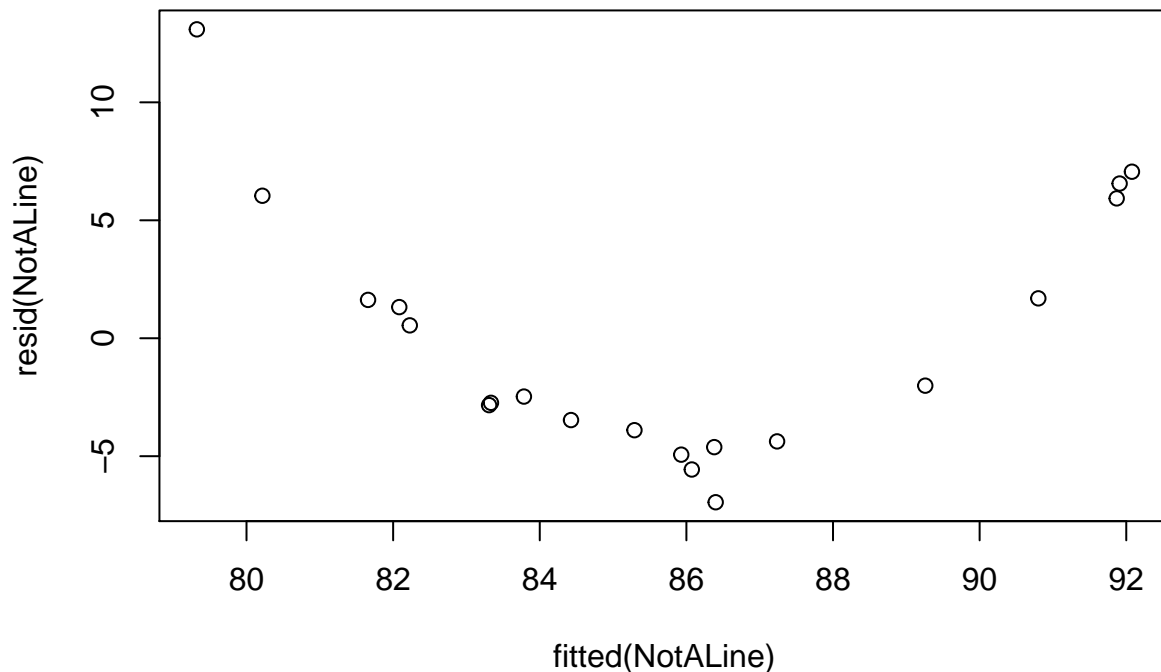






You can make these by hand too. Here are two accessor functions.

```
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.

```
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.

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

```
summary(WTime)
```

```
##
## Call:
## lm(formula = Quantity ~ Price + Time, data = MyData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.5244 -0.6323 -0.1873  0.7215  1.7173
##
## Coefficients:
```

```
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept) 112.49584    0.63562  176.99 < 2e-16 ***
## Price      -2.00458    0.02953  -67.87 < 2e-16 ***
## TimeNight  -12.06681    0.43983  -27.43 1.62e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9754 on 17 degrees of freedom
## Multiple R-squared:  0.9967, Adjusted R-squared:  0.9963
## F-statistic: 2547 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.

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

summary(WTimeInteract)
```

```
##
## Call:
## lm(formula = Quantity ~ Price * Time, data = MyData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.4779 -0.6286 -0.1197  0.6299  1.7696
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept)   112.60528    0.78080  144.218 < 2e-16 ***
## Price         -2.01050    0.03815  -52.696 < 2e-16 ***
## TimeNight     -12.35372    1.20680  -10.237 1.98e-08 ***
## Price:TimeNight  0.01617    0.06307   0.256   0.801
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.003 on 16 degrees of freedom
## Multiple R-squared:  0.9967, Adjusted R-squared:  0.9961
## F-statistic: 1605 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 elasticities 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
## -0.22717 -0.02676  0.02481  0.05043  0.11269
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.55542    0.13179  42.153 < 2e-16 ***
## log(Price)   -0.45458    0.04603  -9.876 1.86e-08 ***
## TimeNight    -0.11947    0.04250  -2.811  0.012 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09456 on 17 degrees of freedom
## Multiple R-squared:  0.8618, Adjusted R-squared:  0.8455
## F-statistic:    53 on 2 and 17 DF,  p-value: 4.948e-08
```

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
## -10.145  -1.291   1.315   2.282   5.455
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  161.832      6.073  26.649 2.63e-15 ***
## log(Price)   -31.078      2.121 -14.654 4.48e-11 ***
## TimeNight     -9.395      1.958  -4.797 0.000168 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.357 on 17 degrees of freedom
## Multiple R-squared:  0.9336, Adjusted R-squared:  0.9258
## F-statistic: 119.6 on 2 and 17 DF,  p-value: 9.688e-11
```

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.
- Heteroskedasticity, which makes it worse.
- Endogeneity, 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 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.

```
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: 604  Min.    :194913  Not Available: 244  Min.    :1.00
## CLIMPUS: 604  1st Qu.:198211  2          :    3  1st Qu.:2.75
## CLLUPUS: 604  Median :199406  3          :    3  Median :4.50
## CLNIPUS: 604  Mean    :199352  834.633    :    3  Mean    :4.50
## CLPRPUS: 604  3rd Qu.:200535  -4657      :    2  3rd Qu.:6.25
## CLSCPUS: 604  Max.    :201708  114        :    2  Max.    :8.00
## (Other):1208      (Other)    :4575
##
##      Description      Unit
## Coal Consumption      : 604  Thousand Short Tons:4832
## Coal Exports          : 604
## Coal Imports          : 604
## Coal Losses and Unaccounted for: 604
## Coal Net Imports      : 604
## Coal Production       : 604
## (Other)               :1208
```

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, let's 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 not 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("RawMonth", "ProductionKShortTon")
summary(CoalProduction)
```

```
##      RawMonth      ProductionKShortTon
## Min.   :197301    100000.27 : 1
## 1st Qu.:198403    100427.638: 1
## Median :199505    100627.455: 1
## Mean   :199490    100653.897: 1
## 3rd Qu.:200606    101144.833: 1
## Max.   :201708    101490.13 : 1
##              (Other)   :530
```

- 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(RawMonth),0,4))) %>%
summary(CoalProduction)
```

```
##      RawMonth      ProductionKShortTon      Year      Month
## Min.   :197301    Min.   : 23664      Min.   :1973    Min.   : 1.00
## 1st Qu.:198403    1st Qu.: 69745      1st Qu.:1984    1st Qu.: 3.00
## Median :199505    Median : 82459      Median :1995    Median : 6.00
## Mean   :199490    Mean   : 79100      Mean   :1995    Mean   : 6.47
## 3rd Qu.:200606    3rd Qu.: 90530      3rd Qu.:2006    3rd Qu.: 9.00
## Max.   :201708    Max.   :104390      Max.   :2017    Max.   :12.00
```

- 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("EIA/COAL")
```

```
summary(Prices)
```

```
##      Week Ended      Central Appalachia 12,500 Btu, 1.2 S02
```

```
## Min.      :2008-05-06  Min.      : 39.55
```

```
## 1st Qu.:2010-09-20  1st Qu.: 52.75
```

```
## Median :2013-01-11  Median : 60.90
```

```
## Mean    :2013-01-19  Mean     : 63.96
```

```
## 3rd Qu.:2015-05-23  3rd Qu.: 68.99
```

```
## Max.    :2017-09-29  Max.     :137.50
```

```
## Northern Appalachia 13,000 Btu, <3.0 S02
```

```
## Min.      : 40.75
```

```
## 1st Qu.: 51.50
```

```
## Median : 63.70
```

```
## Mean     : 64.43
```

```
## 3rd Qu.: 68.65
```

```
## Max.     :145.50
```

```
## Illinois Basin 11,800 Btu, 5.0 S02 Powder River Basin 8,800 Btu, 0.8 S02
```

```
## Min.      :30.70      Min.      : 8.25
```

```
## 1st Qu.:40.32      1st Qu.:10.13
```

```
## Median :46.00      Median :11.55
```

```
## Mean     :45.72      Mean     :11.38
```

```
## 3rd Qu.:47.90      3rd Qu.:12.86
```

```
## Max.     :92.00      Max.     :14.90
```

```
## Uinta Basin 11,700 Btu, 0.8 S02
```

```
## Min.      :35.50
```

```
## 1st Qu.:36.73
```

```
## Median :39.45
```

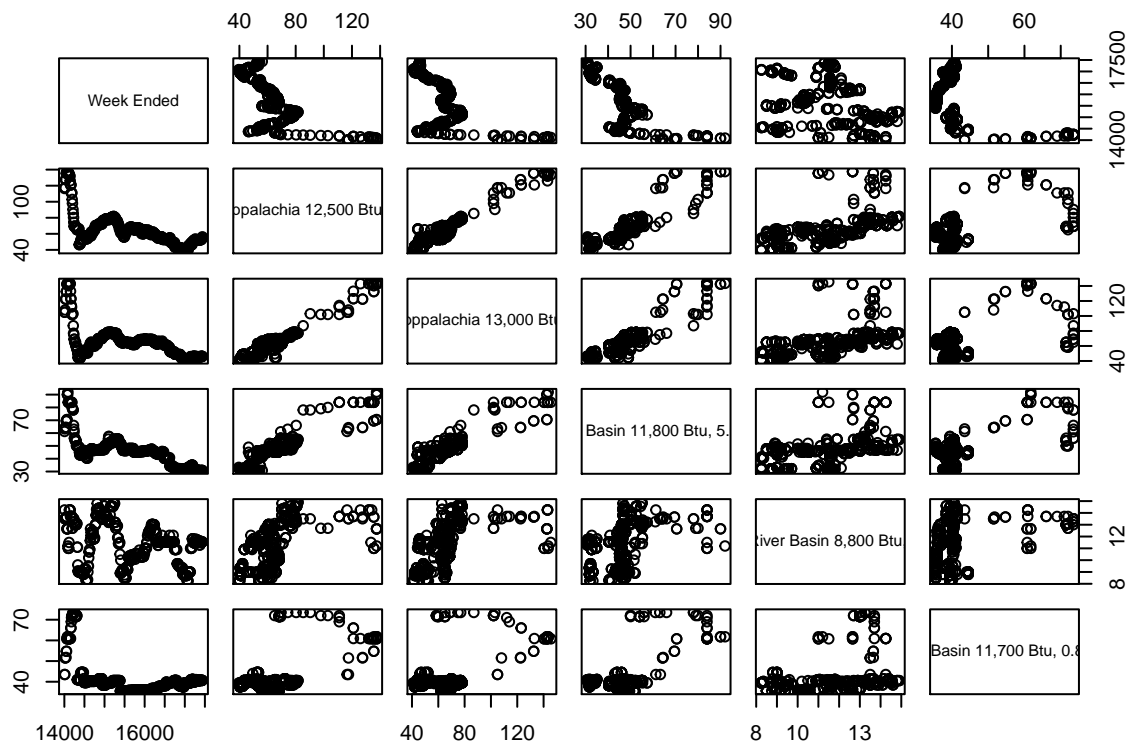
```
## Mean     :41.03
```

```
## 3rd Qu.:40.55
```

```
## Max.     :73.50
```

- Lets make a picture

```
library(ggplot2)
plot(Prices)
```



- Story time on the picture. What do you see? What stories come to mind?
- As before we will convert the year to a numeric value.

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
## The following object is masked from 'package:base':
##
##   date
# Prices$Year <- as.numeric(str_sub(Prices$Year,0,4))

Prices <- Prices %>% mutate(Month = month(`Week Ended`)) %>% mutate(Year = year(`Week Ended`))
```

- Now we are going to get monthly averages

```
Prices <- Prices %>% group_by(Month, Year) %>% summarise_all(mean)
```

For illustrative purposes let's stick with one price and then merge the price and volume data.

```
Prices <- Prices %>% select(Year, Month, Price = `Central Appalachia 12,500 Btu, 1.2 SO2` )
```

- Merge the two data frames

```
summary(CoalProduction)
```

```
##      RawMonth      ProductionKShortTon      Year      Month
## Min.   :197301   Min.   : 23664   Min.   :1973   Min.   : 1.00
## 1st Qu.:198403   1st Qu.: 69745   1st Qu.:1984   1st Qu.: 3.00
## Median :199505   Median : 82459   Median :1995   Median : 6.00
## Mean   :199490   Mean   : 79100   Mean   :1995   Mean   : 6.47
## 3rd Qu.:200606   3rd Qu.: 90530   3rd Qu.:2006   3rd Qu.: 9.00
## Max.   :201708   Max.   :104390   Max.   :2017   Max.   :12.00
```

```
summary(Prices)
```

```
##      Year      Month      Price
## Min.   :2008   Min.   : 1.000   Min.   : 40.50
## 1st Qu.:2010   1st Qu.: 4.000   1st Qu.: 52.83
## Median :2013   Median : 7.000   Median : 61.24
## Mean   :2013   Mean   : 6.522   Mean   : 64.08
## 3rd Qu.:2015   3rd Qu.: 9.000   3rd Qu.: 69.10
## Max.   :2017   Max.   :12.000   Max.   :137.50
```

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

```
summary(CoalMarket)
```

```
##      Year      Month      Price      RawMonth
## Min.   :2008   Min.   : 1.0   Min.   : 40.50   Min.   :200805
## 1st Qu.:2010   1st Qu.: 4.0   1st Qu.: 52.81   1st Qu.:201009
## Median :2012   Median : 6.5   Median : 61.37   Median :201257
## Mean   :2012   Mean   : 6.5   Mean   : 64.17   Mean   :201257
## 3rd Qu.:2015   3rd Qu.: 9.0   3rd Qu.: 69.33   3rd Qu.:201504
## Max.   :2017   Max.   :12.0   Max.   :137.50   Max.   :201708
## ProductionKShortTon
## Min.   : 48115
## 1st Qu.: 76273
## Median : 84543
## Mean   : 82097
## 3rd Qu.: 90310
## Max.   :104390
```

Lets look at the data.

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

