Modeling

David John Baker

12/23/2018

MINERVA
STATISTICAL CONSULTING

Lesson Goals

- Learn how to connect data indexing to ggplot2
- Understand layers and geoms in ggplot2
- Run basic linear regression in R
- Understand linear regression output

Plots and Modeling

Using our data from before, let's now try to make a plot where we predict X from Y. First let's make this with ggplot2.

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0

## -- Conflicts ------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

First let's review from this morning and look at the city of Paris (Texas).

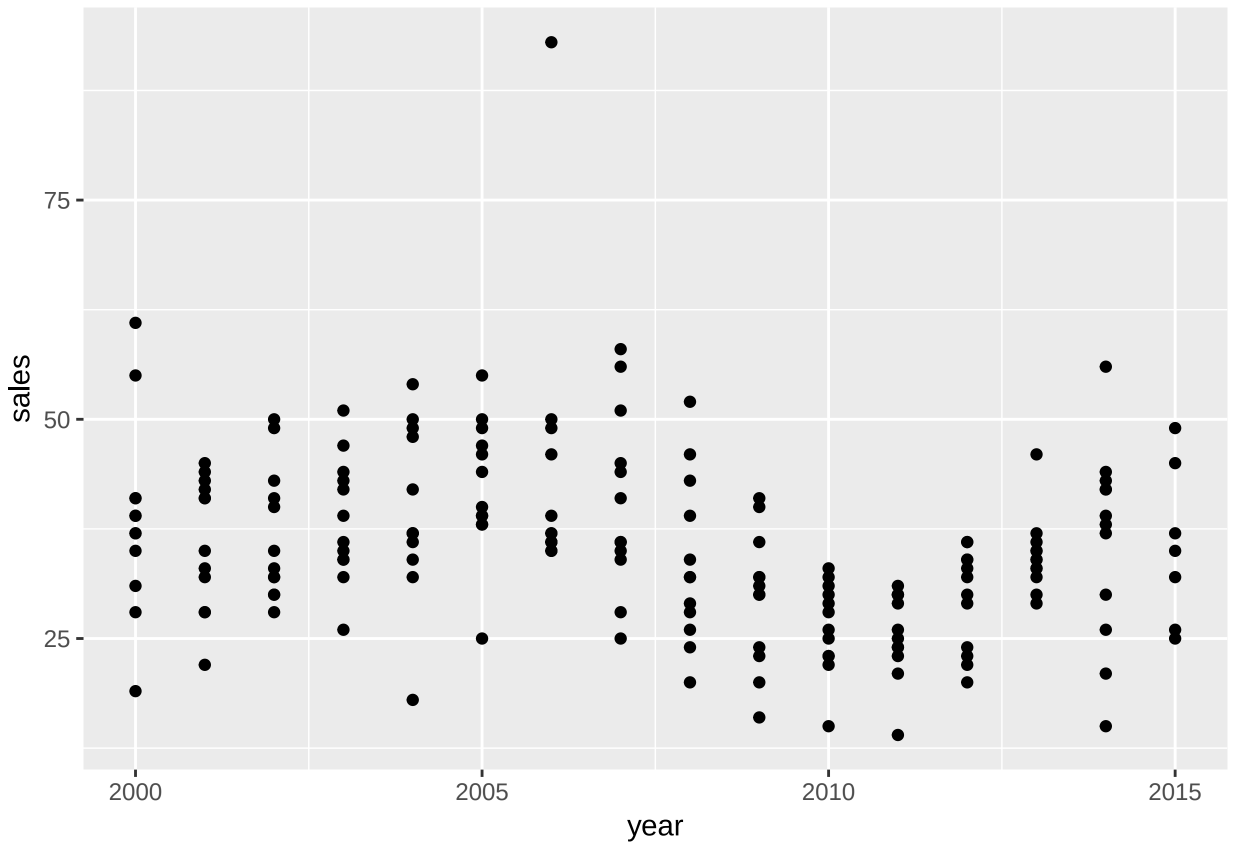
```
txhousing %>%
  filter(city == "Paris")
```

```
## # A tibble: 187 x 9
##    city   year month sales  volume median listings inventory  date
##    <chr> <int> <int> <dbl>   <dbl>  <dbl>    <dbl>     <dbl> <dbl>
##  1 Paris  2000     1    19 1440000  71700      286       7.5 2000
##  2 Paris  2000     2    31 2605000  69400      287       7.5 2000.
##   3 Paris   2000     3    37 2590000  66200      285       7.5 2000.
##   4 Paris   2000     4    41 3340000  66200      273       7   2000.
##   5 Paris   2000     5    55 3815000  58600      280       7   2000.
##   6 Paris   2000     6    39 3295000  79200      273       6.9 2000.
##   7 Paris   2000     7    39 3400000  71200      300       7.7 2000.
##   8 Paris   2000     8    61 4185200  70000      319       7.9 2001.
##   9 Paris   2000     9    41 3490000  75000      315       7.9 2001.
##  10 Paris   2000    10    35 2535000  72100      343       8.6 2001.
## # ... with 177 more rows
```

Now what are going to plot this. Note that we can fuse together what we did before in terms of our tidyverse indexing with that of ggplot2.

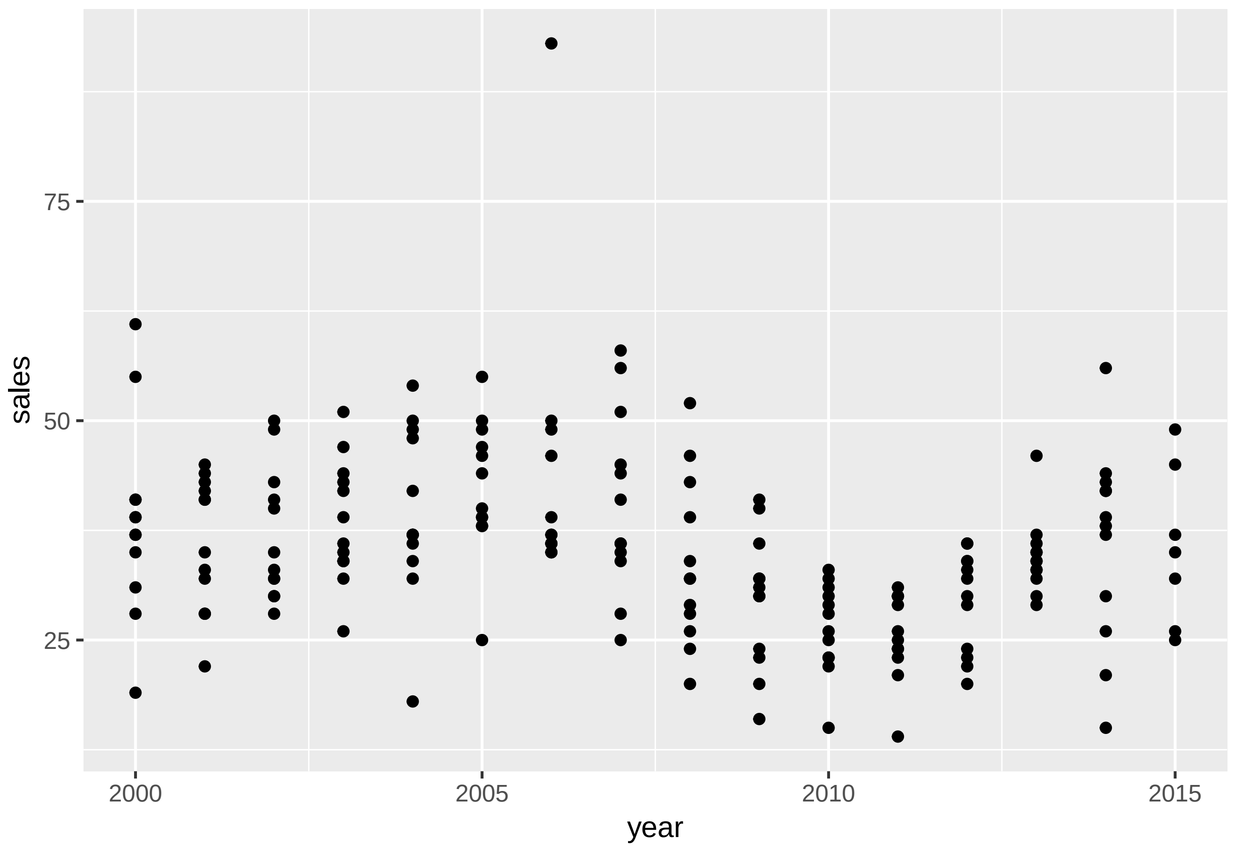
The way that this works it that we pipe our data to ggplot then pass what are called aesthetics to the function. These are very much like using transparencies with an overhead projector.

```
txhousing %>%
  filter(city == "Paris") %>%
  ggplot(aes(x = year , y = sales)) + geom_point()
```



Below is how this would look if we did it with its own data set

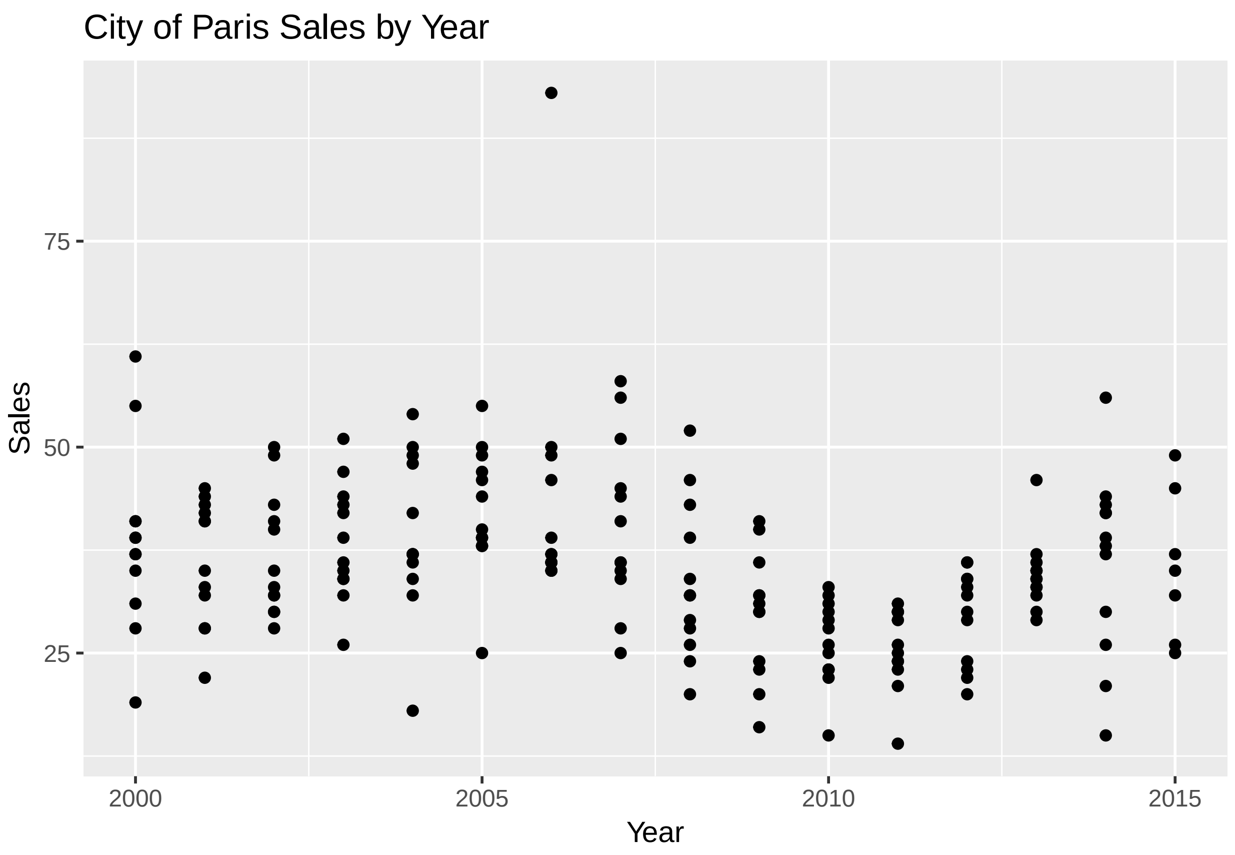
```
by_itself <- txhousing %>%
  filter(city == "Paris")

ggplot(by_itself, aes(x = year , y = sales)) + geom_point()
```



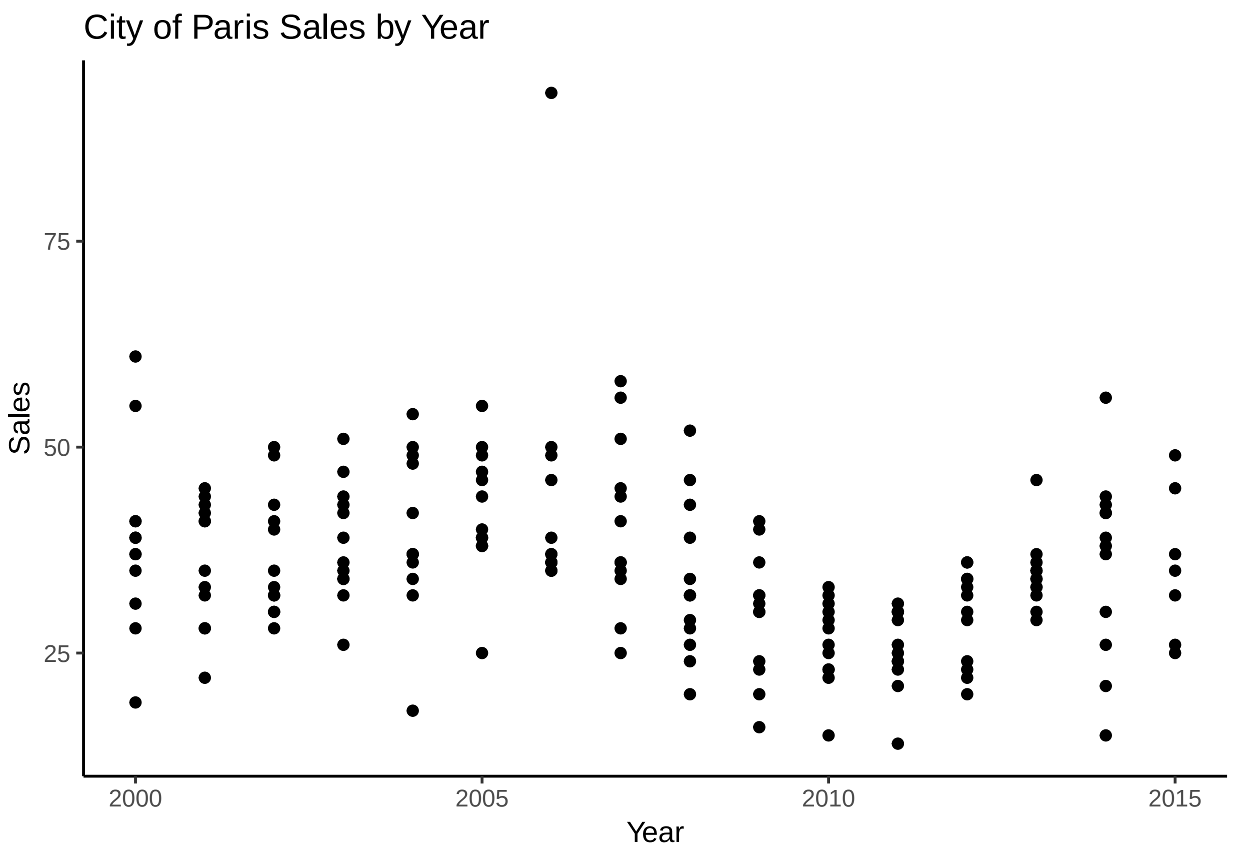
Then to use ggplot, we just add geoms and layers one at a time. We can look at this on many plots using this ggplot2 flipbook

Below together we will work through how to build this with our Texas housing dataset.

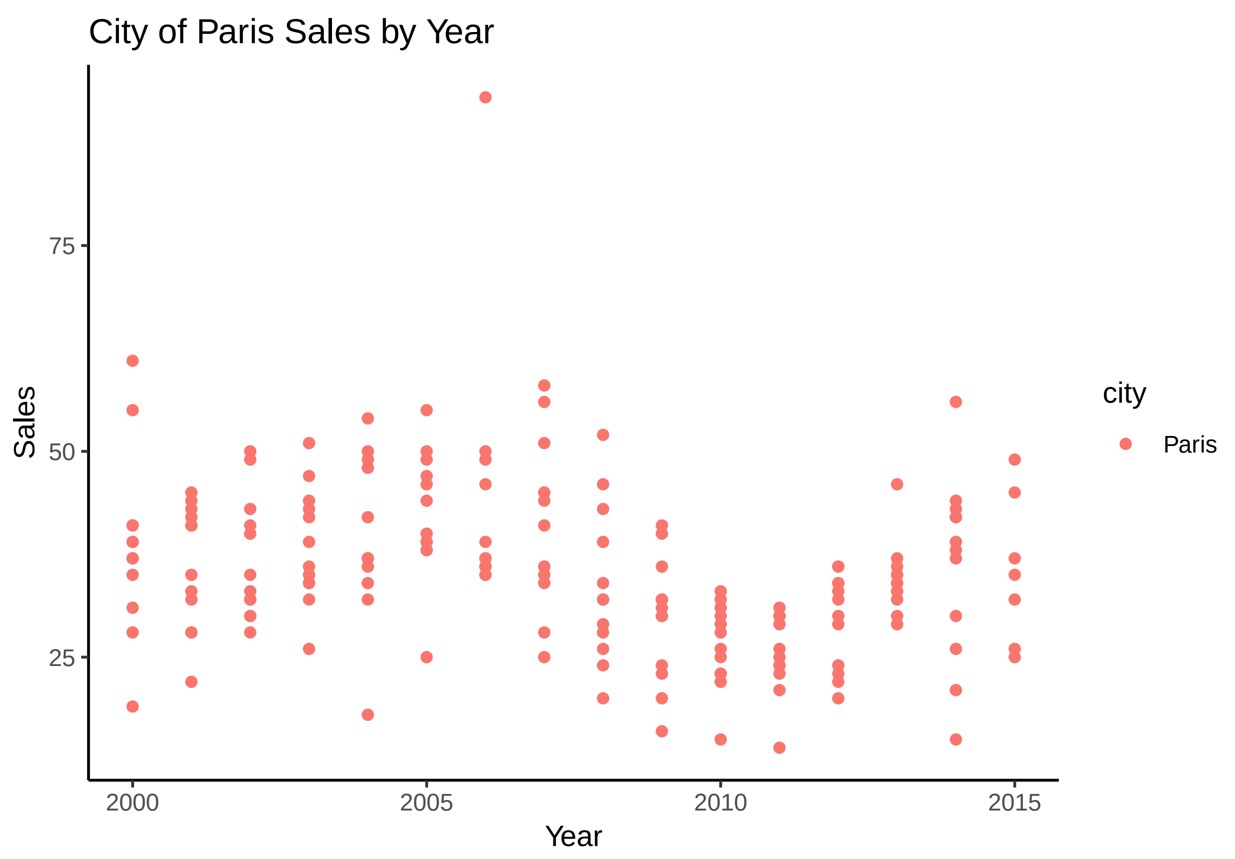
```
txhousing %>%
  filter(city == "Paris") %>%
  ggplot(aes(x = year , y = sales)) + geom_point() +
  labs(title = "City of Paris Sales by Year", x = "Year", y = "Sales")
```



```
txhousing %>%
  filter(city == "Paris") %>%
  ggplot(aes(x = year , y = sales)) + geom_point() +
  labs(title = "City of Paris Sales by Year", x = "Year", y = "Sales") +
  theme_classic()
```

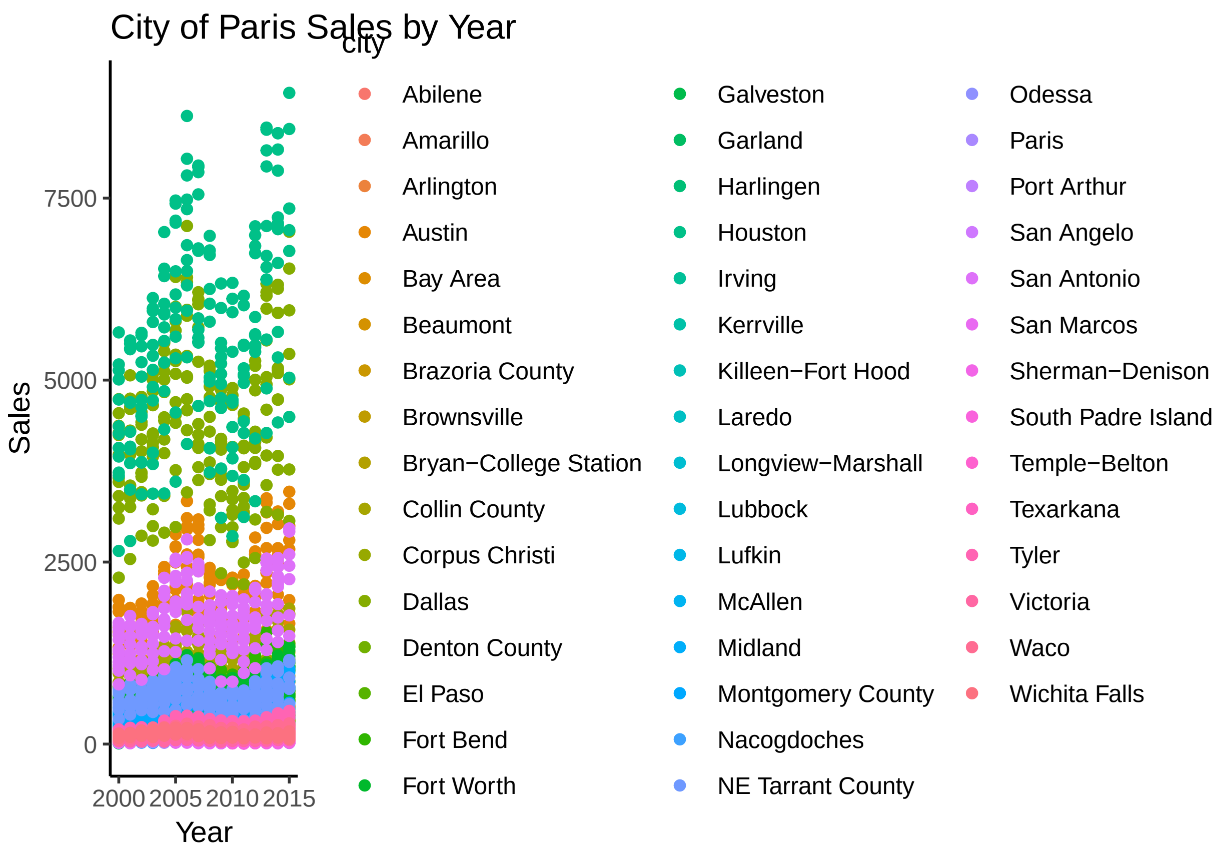


```
txhousing %>%
  filter(city == "Paris") %>%
  ggplot(aes(x = year , y = sales, color = city)) + geom_point() +
  labs(title = "City of Paris Sales by Year", x = "Year", y = "Sales") +
  theme_classic()
```



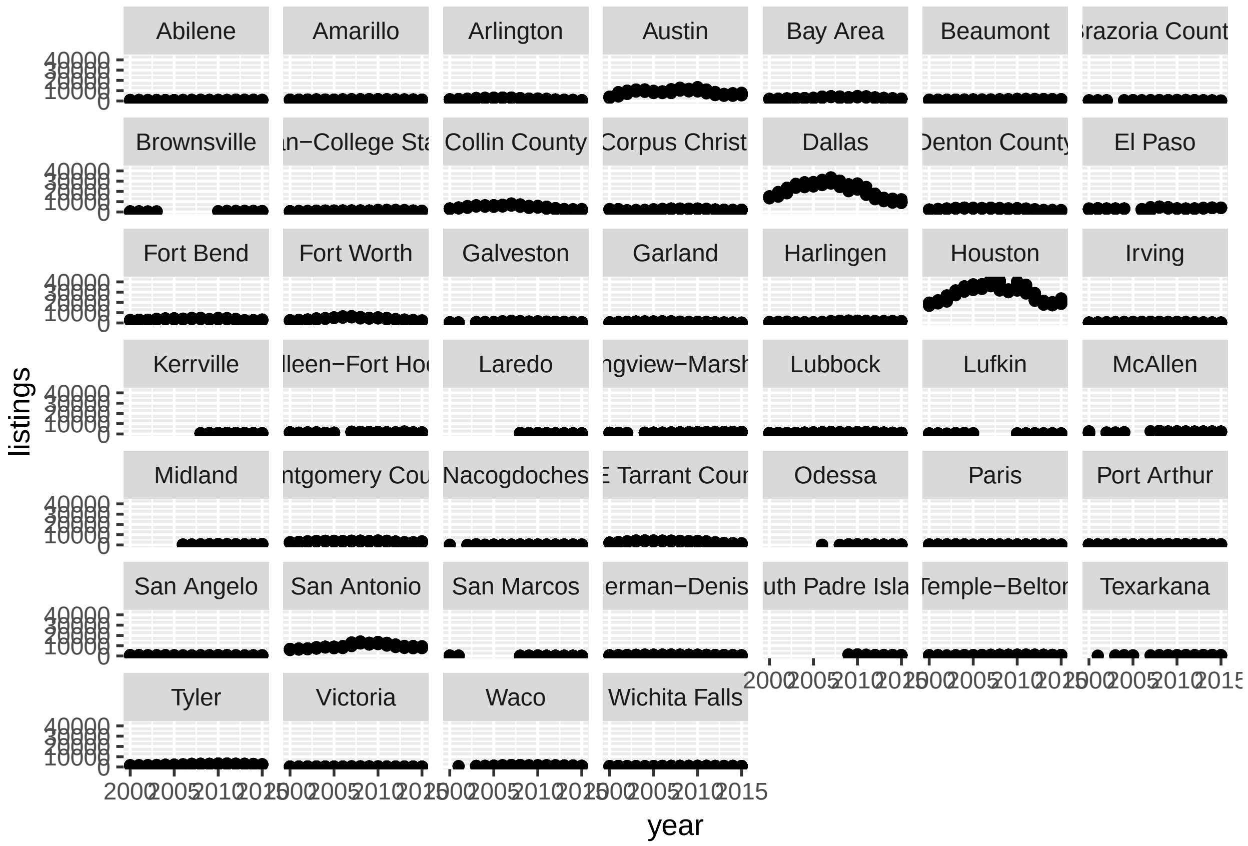
```
txhousing %>%
#  filter(city == "Paris") %>%
  ggplot(aes(x = year , y = sales, color = city)) + geom_point() +
  labs(title = "City of Paris Sales by Year", x = "Year", y = "Sales") +
  theme_classic()
```

```
## Warning: Removed 568 rows containing missing values (geom_point).
```



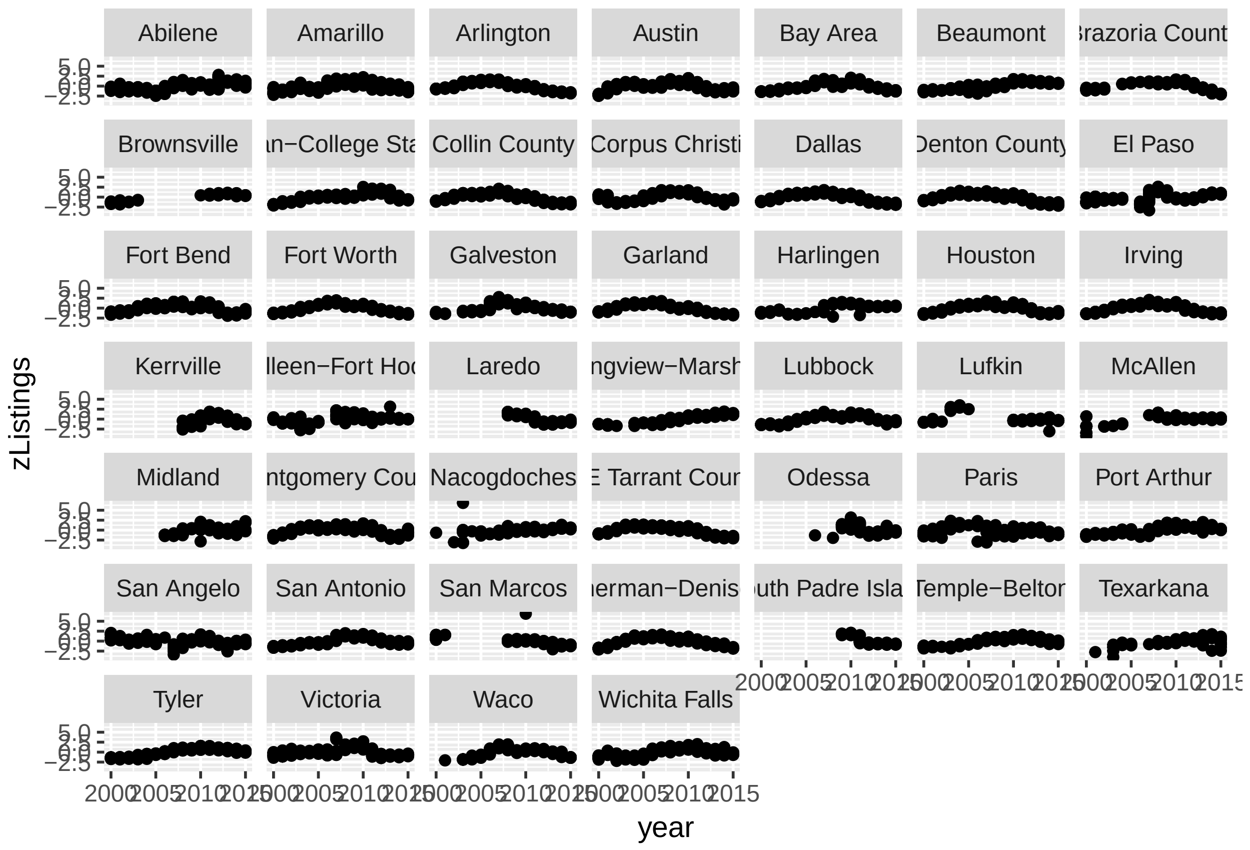
```
txhousing %>%
  ggplot(aes(x = year, y = listings)) + geom_point() +
  facet_wrap(~city)
```

```
## Warning: Removed 1424 rows containing missing values (geom_point).
```



```
txhousing %>%
  group_by(city) %>%
  mutate(zListings = scale(listings)) %>%
  ungroup(city) %>%
  ggplot(aes(x = year, y = zListings)) + geom_point() +
  facet_wrap(~city)
```

```
## Warning: Removed 1424 rows containing missing values (geom_point).
```



Now let's say that we want to run a t-test to see if there are any differences between sales in counties. There are (annoyingly) a few ways to do this with R. Generally we still need our data to be in some sort of tidy format.

```
elpasoVaustin <- txhousing %>%
  select(city, sales) %>%
  filter(city == "El Paso" | city == "Austin")
```

This has our dataset broken into two columns. One is our grouping variable, the other is the values were interested in. We can then pass it to the `t.test` function with various arguments and get our results.

```
# Welch's T test
t.test(elpasoVaustin$sales ~ elpasoVaustin$city )
```

```
##
##  Welch Two Sample t-test
##
## data:  elpasoVaustin$sales by elpasoVaustin$city
## t = 35.529, df = 196.25, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  1438.342 1607.402
## sample estimates:
##  mean in group Austin mean in group El Paso
##             1996.6898             473.8182
```

```
# Student's T Test
t.test(elpasoVaustin$sales ~ elpasoVaustin$city, var.equal = TRUE )
```

```
##
##  Two Sample t-test
##
## data:  elpasoVaustin$sales by elpasoVaustin$city
## t = 35.529, df = 372, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  1438.589 1607.154
## sample estimates:
##  mean in group Austin mean in group El Paso
##             1996.6898             473.8182
```

```
# Paired T-test
# ( Bad use here! )
t.test(elpasoVaustin$sales ~ elpasoVaustin$city, paired = TRUE )
```

```
##
##  Paired t-test
##
## data:  elpasoVaustin$sales by elpasoVaustin$city
## t = 40.301, df = 186, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  1448.325 1597.418
## sample estimates:
## mean of the differences
##                1522.872
```

Though t-tests are important, linear models are the more common model that most people will be running. Linear models model the relationship between one or more variables and another variable. I won't go into the the nitty gritty of the assumptions of linear models, as this is not a stats course, but we will go over how to get everything you need.

Basic Linear Models

Linear models are done with the `lm()` function and follow the general pattern of

> model <- lm(DV ~ IV, data = yourData)
>
> summary(model)

Models are saved as lists meaning you can access any part of the model with the `$` command on the object name. Let's try to run some models on our Texas Housing Data.

First, together let's run a model where we try to predict DV from IV. Once saving this to an object, we can get the output from `summary()`. Let's try to predict

```
options(scipen = 999) # Remove Scientfic notation from R
model1 <- lm(sales ~ listings, data = txhousing)
summary(model1)
```

```
##
## Call:
## lm(formula = sales ~ listings, data = txhousing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3225.0  -107.1   -42.2    20.9  4832.4
##
## Coefficients:
##               Estimate Std. Error t value             Pr(>|t|)
## (Intercept) 22.6613744  6.0499201   3.746             0.000181 ***
## listings     0.1792617  0.0008922 200.913 < 0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 451.1 on 7174 degrees of freedom
##   (1426 observations deleted due to missingness)
## Multiple R-squared:  0.8491, Adjusted R-squared:  0.8491
## F-statistic: 4.037e+04 on 1 and 7174 DF,  p-value: < 0.00000000000000022
```

Model Interpretation

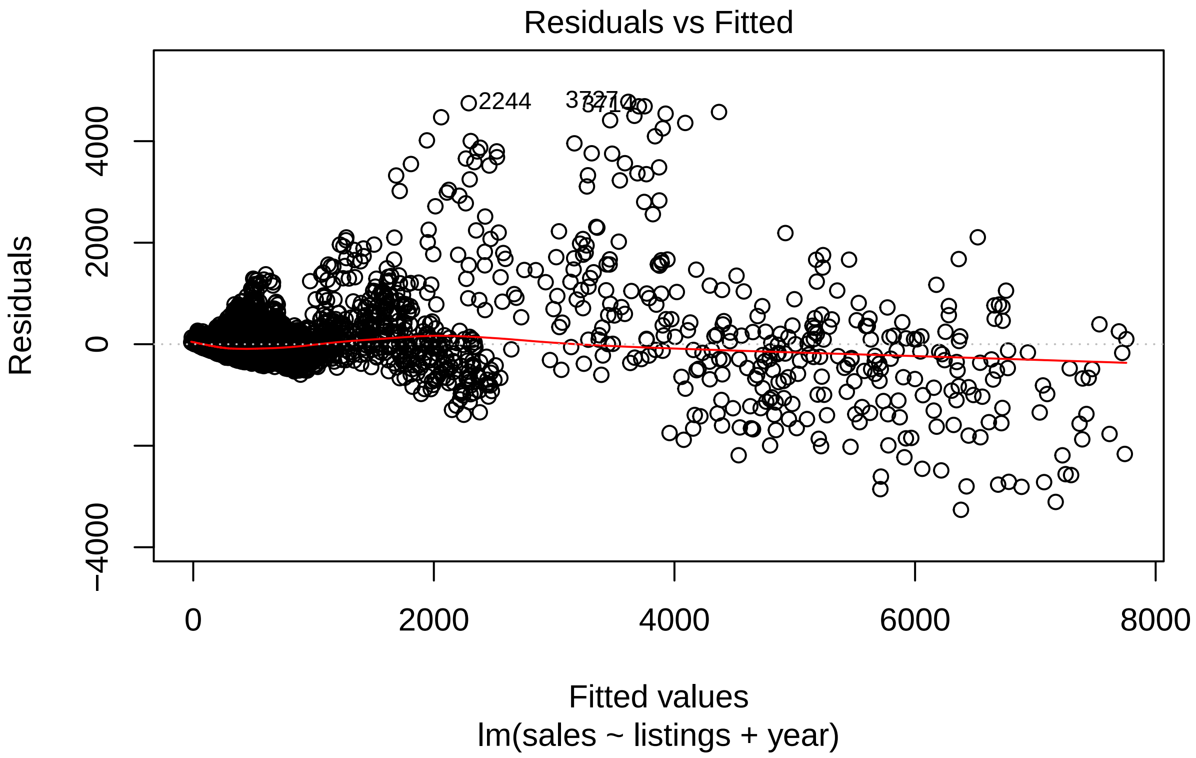
The call lets us know what we put into R. Our coefficients tell us the degree to which one variable effects another. The estimates are our intercept and beta-coefficient (non-standardized!). For everyone one unit increase in our IV (listings) we can expect a .179 rise in our DV (sales) in whatever unit they are! Your test statistics are to the right of that. Standard errors are calculated on your estimate (how much error there is on either side) You t-value is you test statistic. Your p value is the probability of getting a non-zero value given the null hypothesis that there is no effect. Degrees of freedom are listed below. Your multiple-R squared tells you how much variance you explain.

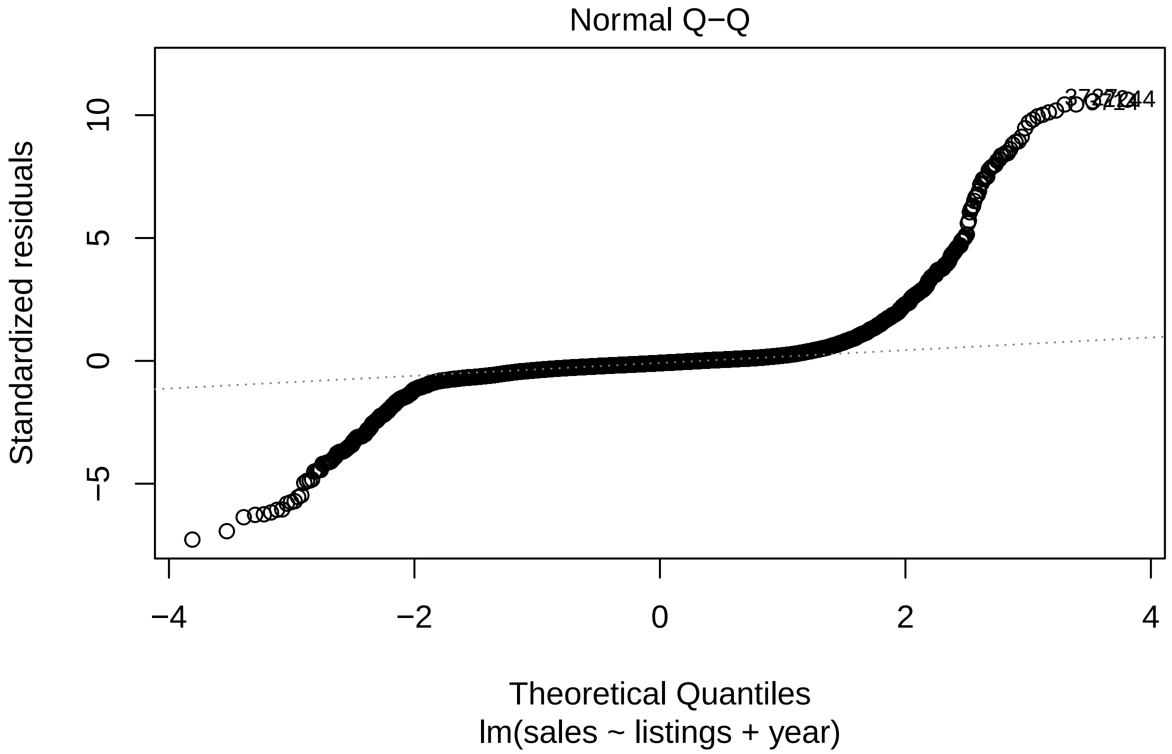
We can then add more variables to try to improve our prediction. For example, what happens when we add in the year as a variable?

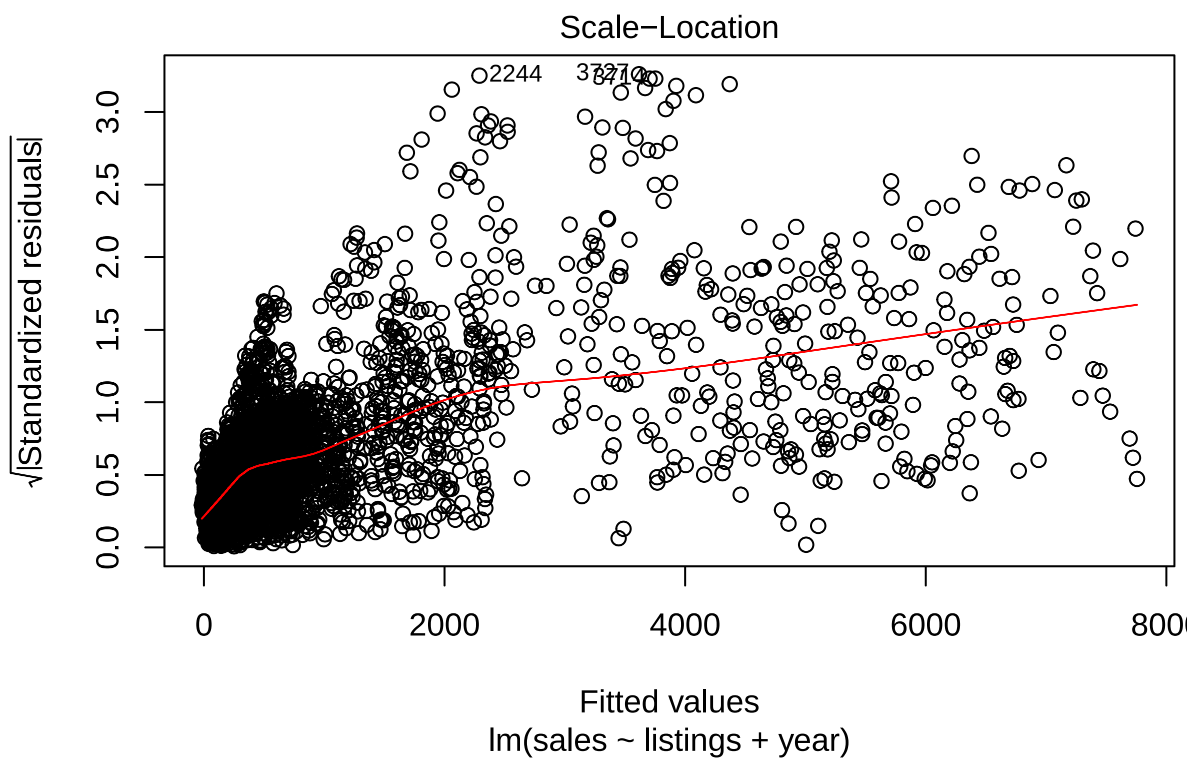
```r
model2 <- lm(sales ~ listings + year, data = txhousing)
summary(model2)
```

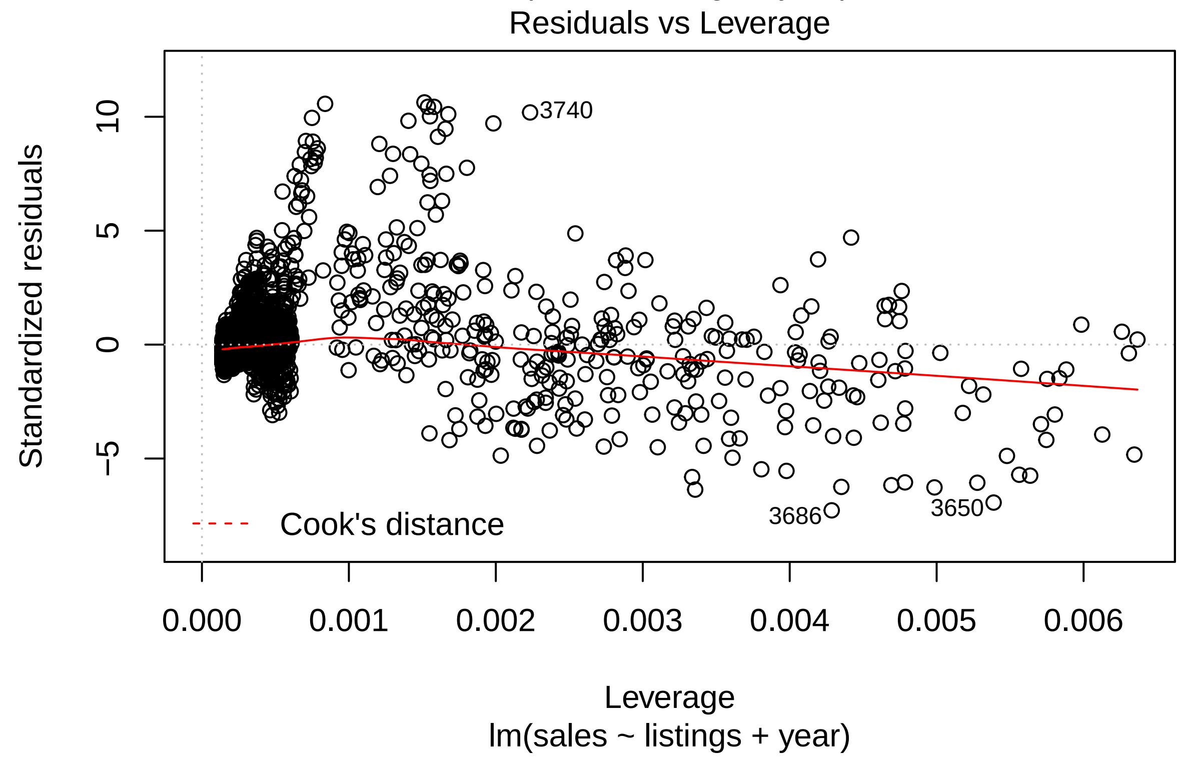
```
##
## Call:
## lm(formula = sales ~ listings + year, data = txhousing)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3262.5  -117.4   -40.1    40.2  4774.4
##
## Coefficients:
##                   Estimate     Std. Error t value             Pr(>|t|)
## (Intercept) -17217.0396880   2346.9861411  -7.336    0.000000000000245 ***
## listings         0.1796013      0.0008902 201.763 < 0.0000000000000002 ***
## year             8.5854200      1.1688019   7.345    0.000000000000228 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 449.5 on 7173 degrees of freedom
##   (1426 observations deleted due to missingness)
## Multiple R-squared:  0.8502, Adjusted R-squared:  0.8502
## F-statistic: 2.036e+04 on 2 and 7173 DF,  p-value: < 0.00000000000000022
```

```r
plot(model2)
```


Residuals vs Fitted
Residuals
Fitted values
lm(sales ~ listings + year)
2244
3727
3714
4000
2000
0
−4000
0
2000
4000
6000
8000


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(sales ~ listings + year)
3727
3714
2244
10
5
0
−5
−4
−2
0
2
4





Here we also find a significant effect of year and listing on sales. Our model fit improves a very small amount, yet the difference is significant. This *significance* should be interpreted with caution as we have a lot of data and the more data you have, the more likely a result is to come up as p <.01.

We also can get information from the model with the `plot()` function. Here we can get into the nitty gritty like checking if our residuals are normally distributed.

We can also test the degree that one model is better than another with the `anova()` function.

```
anova(model1, model2)
```

```
## Analysis of Variance Table
##
## Model 1: sales ~ listings
## Model 2: sales ~ listings + year
##   Res.Df        RSS Df Sum of Sq      F             Pr(>F)
## 1   7174 1459972790
## 2   7173 1449072687  1  10900103 53.956 0.0000000000002276 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Own Analyses

For the rest of the class, I want everyone to try their own plotting and regression models. Use your own data, use data provided by R.

Future

In the future, check out Hadely Wickham's R For Data Science which will give you a solid introduction to R and all it can do. Much of your time will also be spent on Stack Overflow. I also very much suggest setting up a Facebook group or something of people here that are about on the same skill level so that you can help each other out.