Analytics of Business Intelligence

Chapter # 4 - Linear Regression for Business

Dr. Wajahat Gilani

Rutgers Business School

April 24, 2020

Linear Regression for Business

Linear Regression is a statistical technique to represent relationships between two or more variables using a linear equation. Linear regression can be used to predict outcomes. In this lesson the focus will be on:

- Understanding linear regression
- Checking model assumptions
- Using a simple linear regression
- Refining data for simple linear regression
- Introducing multiple linear regression

We will be using the text file Ch4_marketing.csv.

Linear Regression for Business

```
library(data.table)
library(ggplot2)

advert=copy(Ch4_marketing)
setDT(advert)
str(advert)
```

```
'data.frame':   172 obs. of  5 variables:
 $ google_adwords : num  65.7 39.1 174.8 34.4 78.2 ...
 $ facebook       : num  47.9 55.2 52 62 40.9 ...
 $ twitter        : num  52.5 77.4 68 86.9 30.4 ...
 $ marketing_total: num  166 172 295 183 150 ...
 $ revenues       : num  39.3 38.9 49.5 40.6 40.2 ...
```

All the columns are numbers, no need to declare categorical data.

View The Numeric Distributions

We see that 3 of the columns are for advertising on the platforms google_adwords, facebook, and twitter. We want to do a quick view of the distributions:

```
advert[,.(summary(google_adwords),summary(facebook),
    summary(twitter))]
```

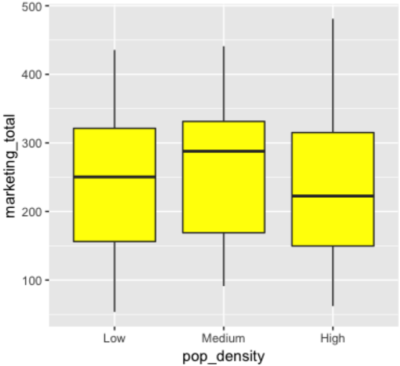
	V1	V2	V3
1:	23.65	8.00	5.89
2:	97.25	19.37	20.94
3:	169.47	33.66	34.59
4:	169.87	33.87	38.98
5:	243.10	47.80	52.94
6:	321.00	62.17	122.19

Google adwords is the most expensive and facebook seems the cheapest, and all 3 seem equally distributed.

View The Numeric Distributions

Now lets visually analyze the distribution. We can do boxplots for all 3 of them, but if we want to see them side by side, then we need to create a new table that has a categorical column. Remember from the previous lecture:

	marketing_total	pop_density
1:	165.98	High
2:	171.70	Medium
3:	294.83	Medium
4:	183.18	High
5:	149.53	Low
6:	62.07	High



The categorical column is the x-axis, and the y-axis is the numerical column. How do we do this with our advert table? We have to create a new table with a categorical column.

View The Numeric Distributions

First lets create a vector with all the data from the 3 columns:

```
g=advert[,google_adwords]
f=advert[,facebook]
tw=advert[,twitter]
nv=c(g,f,tw)
```

Now we need to create a vector that labels each piece of data according to what column belongs to (our categorical data):

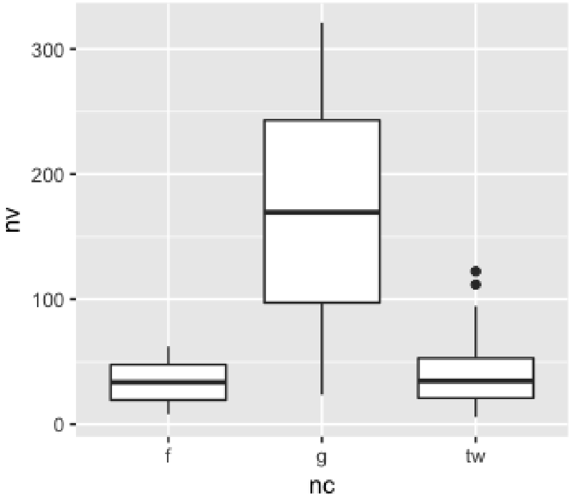
```
nc=c(rep('g',NROW(g)),rep('f',NROW(f)),rep('tw',NROW(
    tw)))
advert2=data.table(nv,nc)
```

	nv	nc
1	65.66	g
2	39.10	g
3	174.81	g
4	34.36	g
5	78.21	g

View The Numeric Distributions

Now we can view the distributions as box-plots:

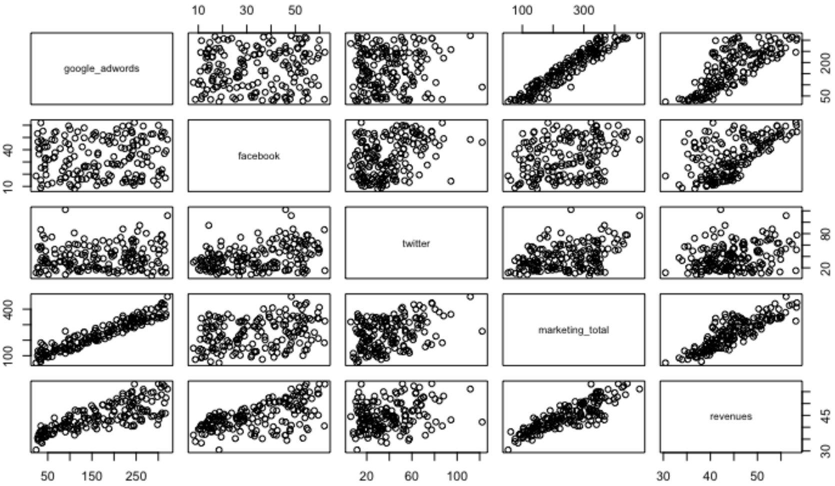
```
ggplot(advert2, aes(x=nc,y=nv)) + geom_boxplot()
```



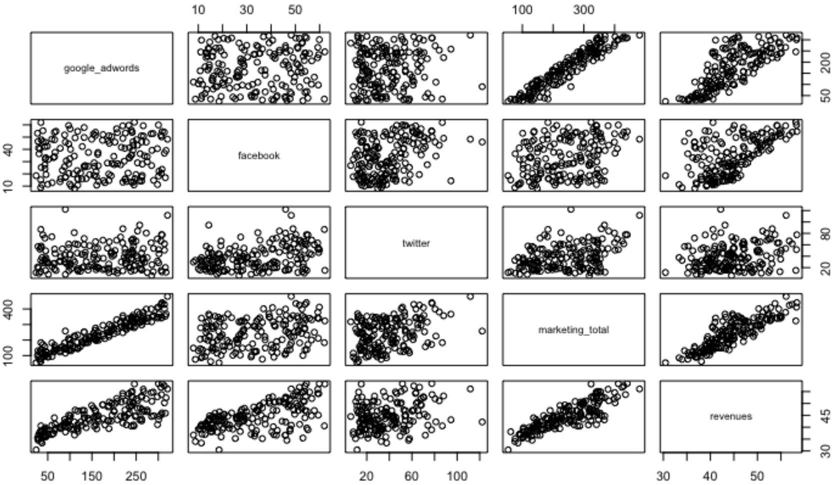
Relationship between Values

Use the pairs function again:

```r
pairs(advert)
```



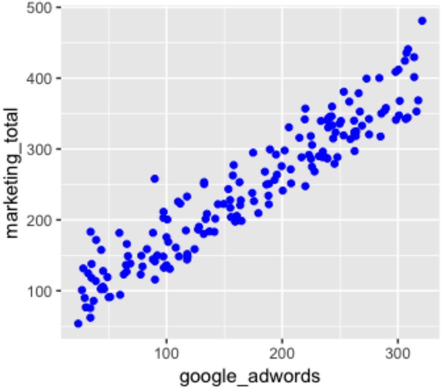
Relationship between Values

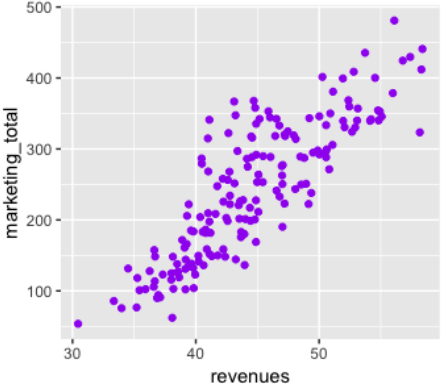


We notice a strong correlations between google_adwords and marketing_total, and revenue and marketing_total.

Relationship between Values - Deeper Look

```r
ggplot(advert, aes(x=google_adwords, y=marketing_total
    )) + geom_point(color='blue')
ggplot(advert, aes(x=revenues, y=marketing_total)) +
    geom_point(color='purple')
```





Simple Linear Regression

Linear regressions are linear models that are constructed according to the data points used in the data points. To construct a linear regression model in R

Regression analysis is used to:

- Predict the value of a dependent variable based on the value of at least one independent variable
- Explain the impact of changes in an independent variable on the dependent variable

Dependent variable

the variable we wish to predict or explain

Independent variable

the variable used to predict or explain the dependent variable

Simple Linear Regression

- Only one independent variable, X
- Relationship between X and Y is described by a linear function
- Changes in Y are assumed to be related to changes in X

Dependent Variable

Population Y intercept

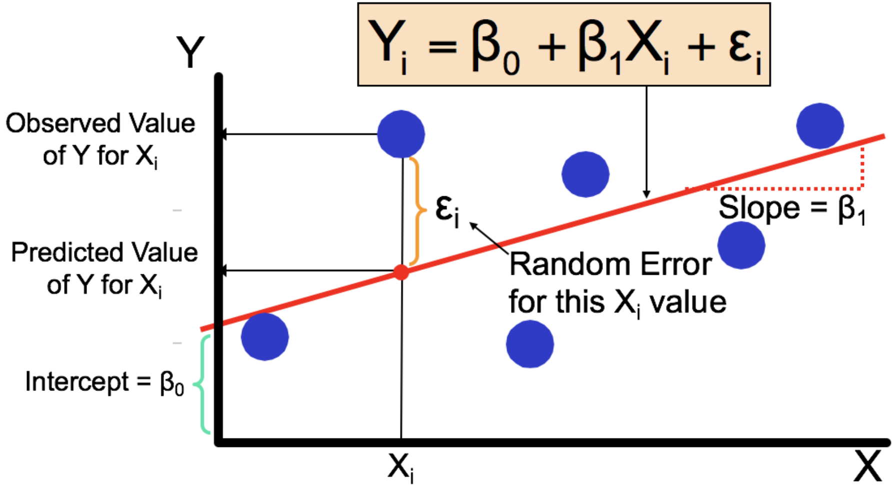
Population Slope Coefficient

Independent Variable

Random Error term

$$\boxed{Y_i = \underbrace{\beta_0 + \beta_1 X_i}_{\text{Linear component}} + \underbrace{\varepsilon_i}_{\text{Random Error component}}}$$

Simple Linear Regression



Simple Linear Regression

The simple linear regression equation provides an estimate of the population regression line

Estimated (or predicted) Y value for observation i

Estimate of the regression intercept

Estimate of the regression slope

Value of X for observation i

$$\boxed{\hat{Y}_i = b_0 + b_1 X_i}$$

Simple Linear Regression

- b_0 is the estimated average value of Y when the value of X is zero
- b_1 is the estimated change in the average value of Y as a result of a one-unit increase in X

lm() function

model = lm(Y∼X, data=data.table)

```
model1=lm(revenues~marketing_total, data = advert)
model1
```

```
Call:
lm(formula = revenues ~ marketing_total, data = advert)

Coefficients:
    (Intercept)  marketing_total
       32.00670          0.05193
```

Interpreting Simple Linear Regression

```
Call:
lm(formula = revenues ~ marketing_total, data = advert)

Coefficients:
    (Intercept)  marketing_total
       32.00670          0.05193
```

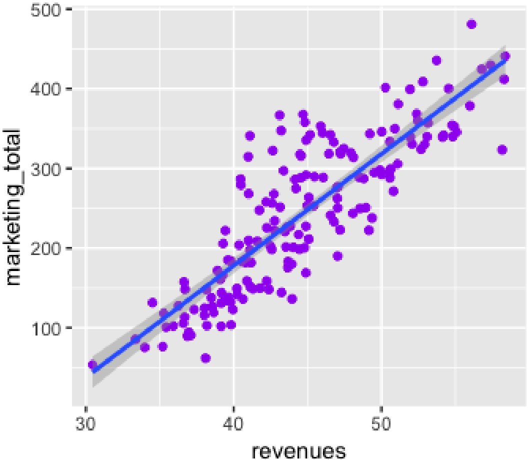
- Revenue increases by $51.93 for every $1,000 increase in total marketing
- Revenue is $32,007 when total marketing expenditure is $0

From here we can make a prediction model:
$Revenue = 32.0067 + (0.05193 * marketing\_total)$

Interpreting Simple Linear Regression

$Revenue = 32.0067 + (0.05193 * marketing\_total)$

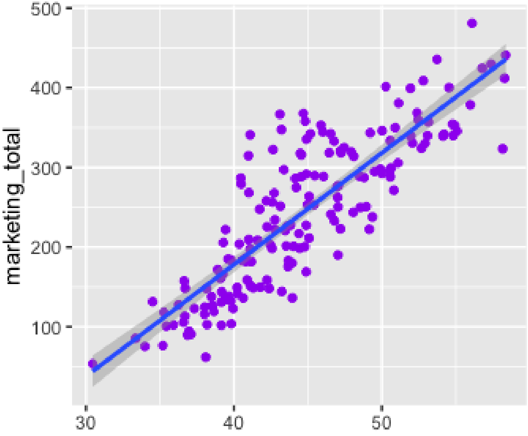
```
ggplot(advert, aes(x=revenues, y=marketing_total)) +
    geom_point(color='purple') +geom_smooth(method = "
    lm")
```



Interpreting Simple Linear Regression

To use linear regression, the data must satisfy 4 core assumptions:

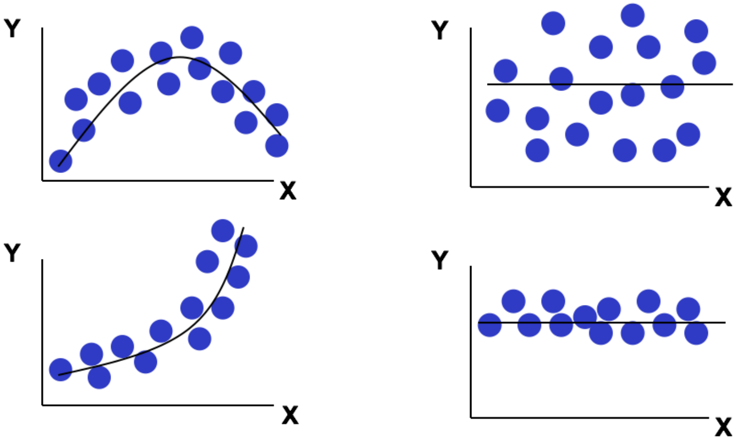
- **L**inearity
- **I**ndependence
- **N**ormality
- **E**qual variance



L.I.N.E Assumptions

Linearity

The relationship between the predictor and response variables is linear. Easy way to view this is by plotting the scatter plot and viewing to see if there is an image of a linear pattern.



L.I.N.E Assumptions

Independence

The relationship between the variables is independent of one another. Very difficult assumption to test. Typically you can handle this by understanding the data and using common sense. The current data uses data from different locations, its safe to say that the data from one location has effect on other locations.

An example of data that wouldn't meet the independence assumption would be a table that has variables income, age and occupation. There is probably some dependence between income and a person's age or occupation. Another thing to watch for if the data is a time-series. Those tend to be influenced by the previous time. We will deal with time-series in Chapter 6.

L.I.N.E Assumptions

Normality

The residuals form a normal distribution around the regression line with mean value of 0.

$$\boxed{e_i = Y_i - \hat{Y}_i}$$

Like in our previous lessons, we can use a histogram to view the distribution of the residuals. Another plot to check the normality is the normal quantile, or Q-Q plot. The Q-Q plot is a scatterplot of the residuals of the model and the Z-scores of those residuals. But where do we get our residuals from? The regression model model1 has the residuals and other data relevant to the results of the linear regression.

```
str(model1)
model1$residuals
```

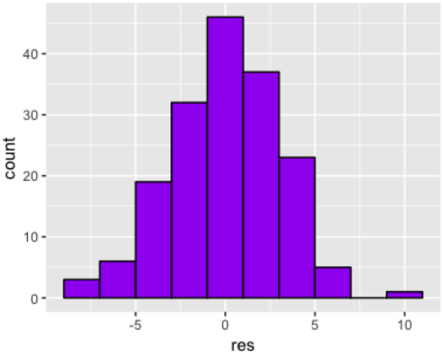
Problem is, to do both graphs, you need a table. No matter we will just create one with 1 columns for the residuals. We don't need a column for the **Z-score** of the residuals, because ggplot() will calculate that for us.

L.I.N.E Assumptions

Normality

We already have the residuals from model1, now create a table **resdf**, and graph the histogram:

```
resdf = data.table('res'=model1$residuals)
ggplot(resdf,aes(x=res)) + geom_histogram(bins=10,fill
    ='purple',color='black')
```



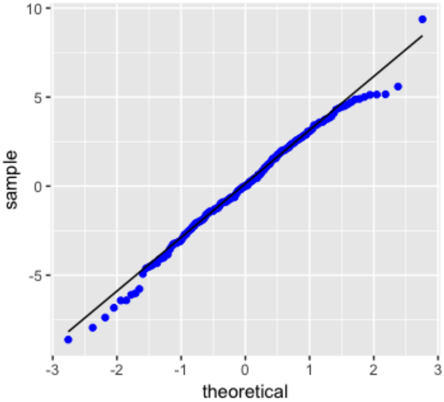
The residuals look somewhat normally distributed, with some sort of an outlier on the right side.

L.I.N.E Assumptions

Normality

We graph the Q-Q plot, using the stat graphs in ggplot():

```
ggplot(resdf,aes(sample=res)) + stat_qq(color='blue')
    + stat_qq_line()
```



The Z-scores mapped with the residuals look they are diagonally plotted for the most part. Notice the outlier from before.

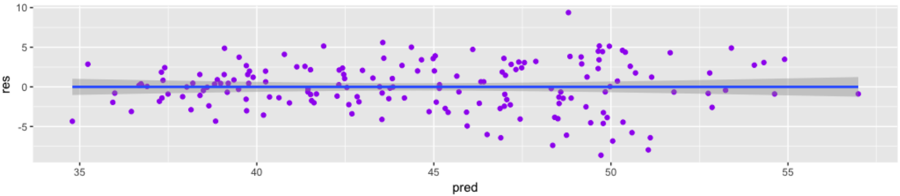
L.I.N.E Assumptions

Equal Variance

Residuals form a random pattern distributed around a mean of 0.

This can be visualized by plotting the residuals against predicted values of the model. Just like the residuals, the predicted values are in model1. We can use the **resdf** table, we can add a column pred that has the predicted values, and then create a scatter-plot with the residuals on the y-axis and the predicted values on the x-axis.

```
resdf[,pred:=model1$fitted.values]
ggplot(resdf,aes(x=pred,y=res)) + geom_point(color='
    purple') + geom_smooth(method='lm')
```



Model Output

There is a lot of information available within model1, using the summary function:

```r
summary(model1)
```

```
Call:
lm(formula = revenues ~ marketing_total, data = advert)

Residuals:
    Min      1Q  Median      3Q     Max
-8.6197 -1.8963 -0.0006  2.1705  9.3689

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     32.006696   0.635590   50.36   <2e-16 ***
marketing_total  0.051929   0.002437   21.31   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.054 on 170 degrees of freedom
Multiple R-squared:  0.7277,    Adjusted R-squared:  0.7261
F-statistic: 454.2 on 1 and 170 DF,  p-value: < 2.2e-16
```

Predicting Outputs

You can predict outputs using the SLR model, this works best if you predict using inputs inside the range of the **existing** model. Revenues can be predicted for values within the marketing_total of:

```
summary(advert$marketing_total)
```

The values between [53.65,481]. Suppose you want to predict the revenues if you want to spend $460,000 in marketing. First the data does not have an input of 460:

```
advert[marketing_total>430,marketing_total]
```

481.00 435.49 440.94

We can use the predict.lm() function to estimate based on other inputs. You need to put your prediction(s) in a table and then run the function:

```
newrev = data.table(marketing_total=seq(460,470,5))
predict.lm(model1,newrev,interval = 'predict')
```

Predicting Outputs

```r
newrev = data.table(marketing_total=seq(460,470,5))
predict.lm(model1,newrev,interval = 'predict')
```

```
     fit      lwr      upr
55.89403 49.75781 62.03025
56.15368 50.01331 62.29404
56.41332 50.26873 62.55791
```

For the value of $460,000 you get an estimate of revenue of $55,894. Now that can't be an exact number because it is an estimate. That is why you are given a 95% confidence interval. What the confidence interval is saying is that if you were to make 100 predictions with marketing_total being 460, then 95 out of the 100 results would fall in between 55.89403 and 62.03025. If you wanted a 99% interval, then you would use the parameter:

```r
predict.lm(model1,newrev,level=.99,interval = 'predict
    ')
```

```
     fit      lwr      upr
55.89403 47.79622 63.99184
56.15368 48.05040 64.25695
```

Confidence Intervals

Before MapReduce and Hadoop, statisticians would analyze massive datasets by taking random samples, crunching the numbers and then predicting what the values of the actual dataset would be. Because you are using a random sample of data from a larger dataset, you cannot be sure that your answer is correct, because its just an estimate of unknown data. The cofint() function, allows you to see the estimate of error. For example, assume that we use 30% of the data to analyze the rest of the data:

```
set.seed(4510)
liladvert = advert[sample(.N,.3*.N)]
samp_model = lm(revenues~marketing_total,data=
    liladvert)
samp_model
```

```
Call:
lm(formula = revenues ~ marketing_total, data = liladvert)

Coefficients:
    (Intercept)  marketing_total
       30.65396          0.05739
```

Confidence Intervals - sample() and .N Functions

The sample() function takes **random** samples from a data container. Here is a simple example using a vector:

```
vv=5:15
sample(vv,5)
```

The sample() function will randomly choose 5 different values from 5 to 15. Every time you run that line of code, you will see you keep getting different values (Random).

If you are doing a particular simulation and you would like to see the same results simulated, that is when you use the set.seed() feature. You can put any number within the parenthesis (example: 4510), and it will simulate the same numbers for you.

```
set.seed(7)
sample(vv,5)

set.seed(7)
sample(vv,5)
```

Confidence Intervals - sample() and .N Functions

If you just ran:

```
sample(vv,5)
sample(vv,5)
sample(vv,5)
```

Each time you ran it, you got different values.

Now using the sample() function in a data.table():

```
liladvert = advert[sample(.N,.3*.N)]
```

The .N is a strange command in data.table() that replaces the function NROW(). Anywhere you see it, it is giving you the number of rows in the table. In the context of this function, it is saying from all the rows of the advert table, choose only 30% of them or only 51 rows (decimal place is dropped).

Confidence Intervals

```r
set.seed(4510)
liladvert = advert[sample(.N,.3*.N)]
samp_model = lm(revenues~marketing_total,data=
    liladvert)
samp_model
```

```
Call:
lm(formula = revenues ~ marketing_total, data = liladvert)

Coefficients:
    (Intercept)  marketing_total
       30.65396          0.05739
```

```r
confint(samp_model)
```

```
                    2.5 %      97.5 %
(Intercept)     28.2397634 33.06815353
marketing_total  0.0480053  0.06676511
```

```
Coefficients:
    (Intercept)  marketing_total
       32.00670          0.05193
```

Confidence Intervals - Interpretation

```
Call:
lm(formula = revenues ~ marketing_total, data = liladvert)

Coefficients:
    (Intercept)  marketing_total
       30.65396          0.05739
```

```
confint(samp_model)
```

```
                      2.5 %      97.5 %
(Intercept)      28.2397634 33.06815353
marketing_total   0.0480053  0.06676511
```

```
Coefficients:
    (Intercept)  marketing_total
       32.00670          0.05193
```

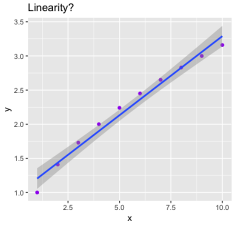
This shows that if you take 100 different random samples from the advert table, 95% of them will estimate the slope of marketing_total between .0480053 and .06676511.

This is the main concept behind sampling and statistics!

Refining data for Simple Linear Regression

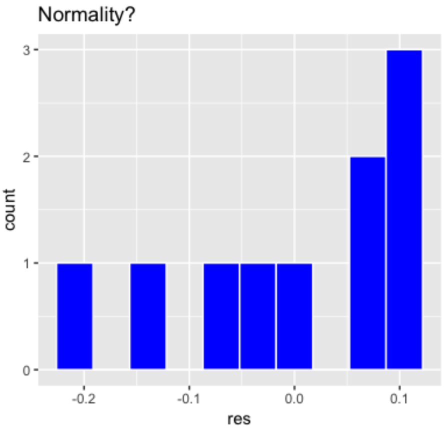
There are times when the **L.I.N.E.** (Linearity, Independence, Normality, and Equal Variance) assumptions are violated. Consider the sample dataset below. Run a SLR and generate the diagnostic plots:

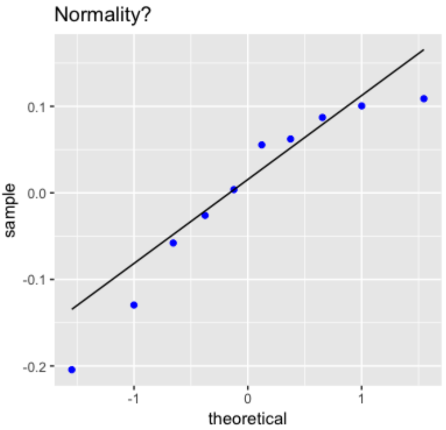
```r
x=1:10
y=c(1,1.41,1.73,2,2.24,2.45,2.65,2.83,3,3.16)
fit = lm(y~x)
sampdt = data.table(x,y)

ggplot(sampdt,aes(x=x,y=y)) + geom_point(color = 'purple') + geom_smooth(method = "lm") + labs(title="Linearity?")
```



Refining data for Simple Linear Regression

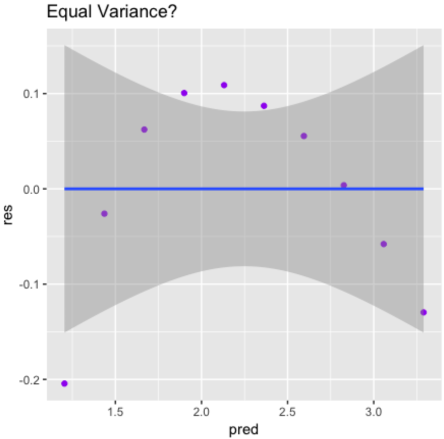
```
sampdt[,res:=fit$residuals]
ggplot(sampdt,aes(x=res)) + geom_histogram(bins=10,
    fill='blue',color='white') + labs(title= "Normality
    ?")
```



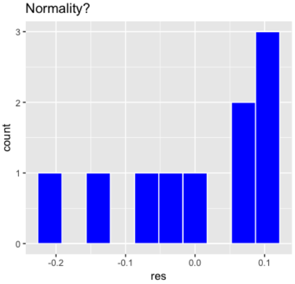


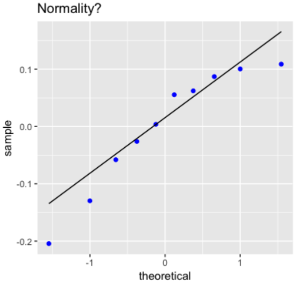
Refining data for Simple Linear Regression

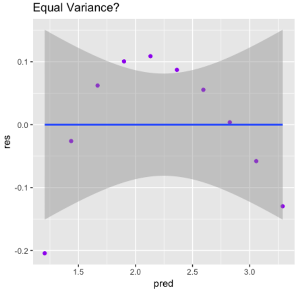
```
sampdt[,pred:=fit$fitted.values]
ggplot(sampdt,aes(x=pred,y=res)) + geom_point(color='
    purple') + geom_smooth(method = 'lm') + labs(title=
     "Equal Variance?")
```



Refining data for Simple Linear Regression





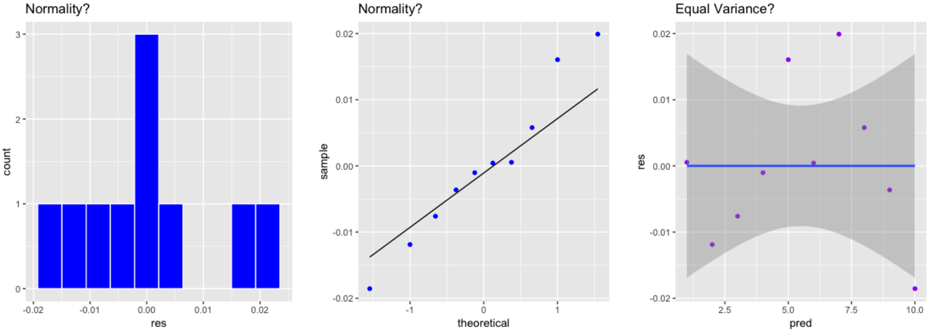


The first plot shows linearity, but the other graphs do show some violations to the Normality and Equal Variance properties. The data is simulated, the y variable are the square roots of the x variable. This is where data transformation comes into play.

Transforming Data

```r
y2 = y^2
fit2=lm(y2~x)
sampdt2 = data.table(x,y2)
sampdt2[,res:=fit2$residuals]
sampdt2[,pred:=fit2$fitted.values]
ggplot(sampdt,aes(x=res)) + geom_histogram(bins=10, fill='blue',color='white') + labs(title= "Normality?")
ggplot(sampdt,aes(sample=res)) + stat_qq(color="blue") +stat_qq_line() +labs(title= "Normality?")
ggplot(sampdt2,aes(x=pred,y=res)) + geom_point(color='purple') + geom_smooth(method = 'lm') + labs(title= "Equal Variance?")
```

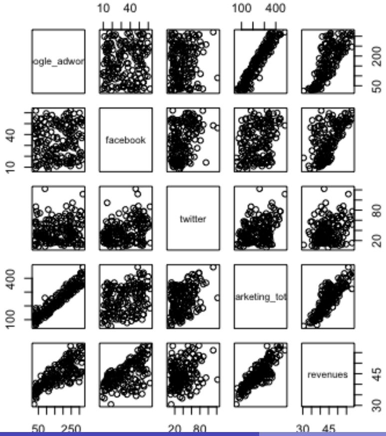
Transforming Data



Multiple Regression

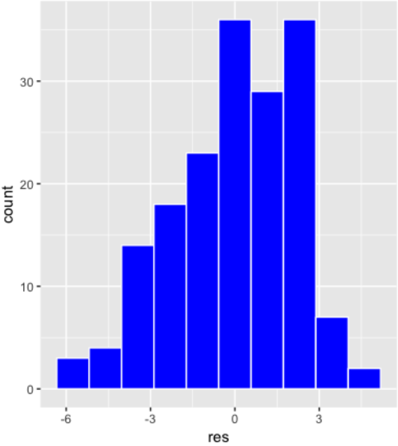
```
model2 = lm(revenues ~ google_adwords + facebook +
    twitter, data=advert)
plot(advert)
```

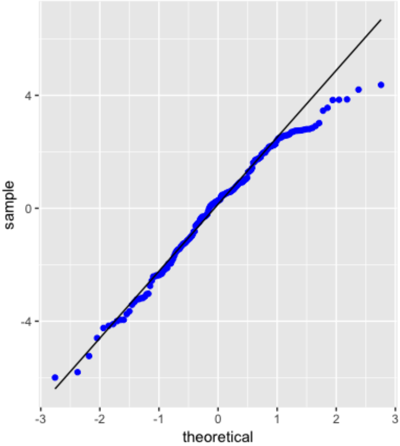
Linearity:



Multiple Regression

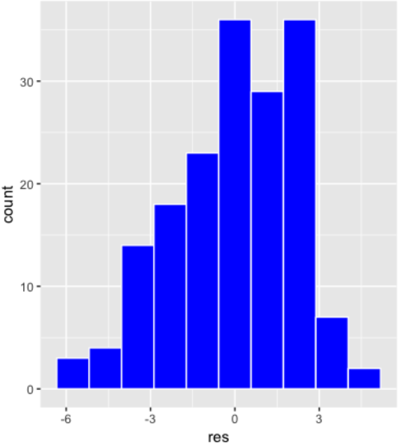
```
ggplot(resdf2,aes(x=res)) + geom_histogram(bins=10,
    fill='blue',color='white')
ggplot(resdf2,aes(sample=res)) + stat_qq(color="blue")
     +stat_qq_line()
```

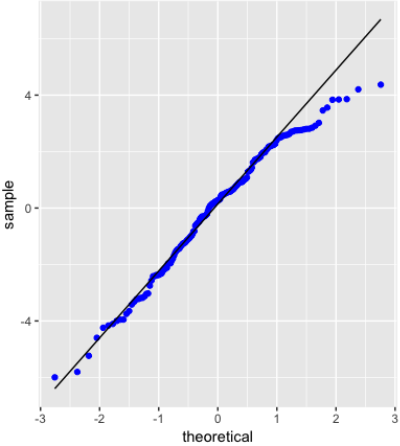




Multiple Regression

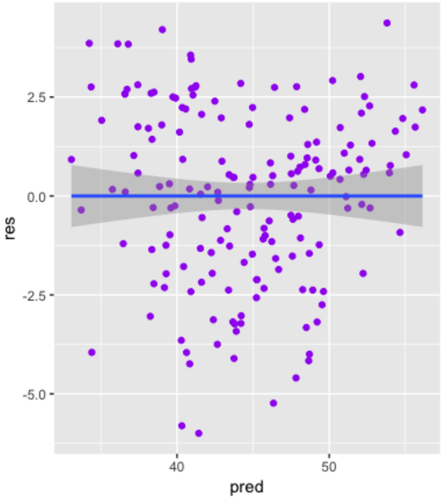
```
ggplot(resdf2,aes(x=res)) + geom_histogram(bins=10,
    fill='blue',color='white')
ggplot(resdf2,aes(sample=res)) + stat_qq(color="blue")
    +stat_qq_line()
```





Multiple Regression

```
ggplot(resdf2,aes(x=pred,y=res)) + geom_point(color='purple') + geom_smooth(method = 'lm'))
```



Multiple Regression - Interpreting Results

```
summary(model2)
```

```
Call:
lm(formula = revenues ~ google_adwords + facebook + twitter,
    data = advert)

Residuals:
    Min      1Q  Median      3Q     Max
-5.9971 -1.4566  0.2791  1.7428  4.3711

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)   29.545988   0.533523   55.38   <2e-16 ***
google_adwords 0.048384   0.001947   24.85   <2e-16 ***
facebook       0.197651   0.011871   16.65   <2e-16 ***
twitter        0.003889   0.008270    0.47    0.639
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.214 on 168 degrees of freedom
Multiple R-squared:  0.8585,    Adjusted R-squared:  0.856
F-statistic: 339.8 on 3 and 168 DF,  p-value: < 2.2e-16
```

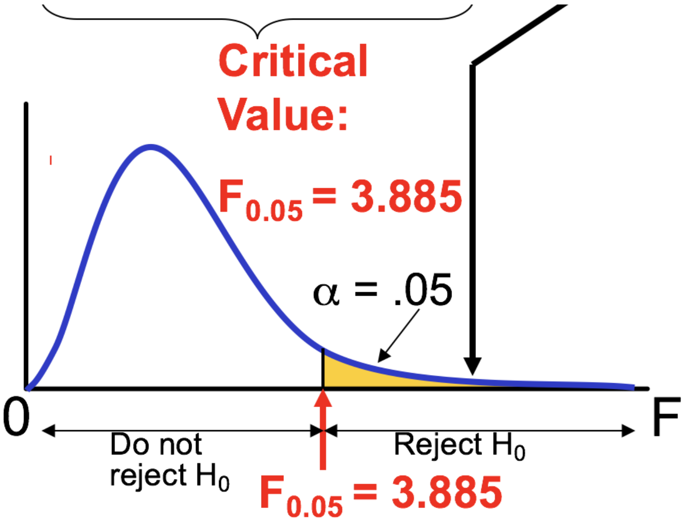
Multiple Regression - F statistic

- F Test for Overall Significance of the Model
- Shows if there is a linear relationship between all of the X variables considered together and Y
- Use F-test statistic
- Hypotheses:

H_0: $\beta_1 = \beta_2 = \cdots = \beta_k = 0$ (no linear relationship)

H_1: at least one $\beta_i \neq 0$ (at least one independent variable affects Y)

Multiple Regression - F statistic



Multiple Regression - Interpreting Results

```r
qf(.95, df1=3, df2=168)
```

2.658399

```
Call:
lm(formula = revenues ~ google_adwords + facebook + twitter,
    data = advert)

Residuals:
    Min      1Q  Median      3Q     Max
-5.9971 -1.4566  0.2791  1.7428  4.3711

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    29.545988   0.533523   55.38   <2e-16 ***
google_adwords  0.048384   0.001947   24.85   <2e-16 ***
facebook        0.197651   0.011871   16.65   <2e-16 ***
twitter         0.003889   0.008270    0.47    0.639
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.214 on 168 degrees of freedom
Multiple R-squared:  0.8585,    Adjusted R-squared:  0.856
F-statistic: 339.8 on 3 and 168 DF,  p-value: < 2.2e-16
```

Multiple Regression - Interpreting Results

The adjusted R-squared value shows how much of the variation in the dependent variable, is explained by the variation in the independent variables. It is a good explanation of fit. Our variables explain a lot of the variance, hence, a good fit.

The 3 predictors also generate their own p-values via the t-statistic. Low p-values indicate that the variable helps explain the model better than a high p-value. Which variable can be considered unnecessary?

```
Call:
lm(formula = revenues ~ google_adwords + facebook + twitter,
    data = advert)

Residuals:
    Min      1Q  Median      3Q     Max
-5.9971 -1.4566  0.2791  1.7428  4.3711

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    29.545988   0.533523   55.38   <2e-16 ***
google_adwords  0.048384   0.001947   24.85   <2e-16 ***
facebook        0.197651   0.011871   16.65   <2e-16 ***
twitter         0.003889   0.008270    0.47    0.639
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.214 on 168 degrees of freedom
Multiple R-squared:  0.8585,    Adjusted R-squared:  0.856
F-statistic: 339.8 on 3 and 168 DF,  p-value: < 2.2e-16
```