STAT40620: Data Programming with R - Final

Wilson Tai 21204700

22/12/2022

1 Introduction

This dataset looks at the emotions and the number of tweets of 28 world events. These events include accidents, deaths and sport events but to name 3, and the emotions provided are Sarcasm, Irony and Humour. This dataset was sourced from the Loughborough University dataset website: https://repository.lboro.ac.uk/articles/dataset/Overview_of_the_25_Unique_Events_Topics/13084817 (https://repository.lboro.ac.uk/articles/dataset/Overview_of_the_25_Unique_Events_Topics/13084817) .

The dataset is not big: as aforementioned, there are only 28 events (observations). In spite of this, I as able to derive some interesting results, although possibly bias due to such a small sample. Let's begin!

2 Cleaning

Before I get into my first section of the instruction list, Analysis, I want to clean and set up the data appropriately first.

```
Event <- read.csv("/Users/wilsontai/Downloads/overview_of_events_datasets (1).csv")
attach(Event)

##loading libraries
library(stargazer)
library(lmtest)
library(sandwich)
library(portes)
library(Hmisc)
library(PerformanceAnalytics)
library(olsrr)
library(plyr)
library(car)
library(ggcorrplot)
```

In the above, I am loading in my dataset and libraries required for this project. In the following, I am cleaning up some components of the data to allow an easy understanding for both the reader and I:

```
##getting general overview
head(Event)
```

```
##            Dataset Total..N. Emotional.Tweets.... Emotional.Tweets.....1
## 1 helicopter crash     25387               0.1399                  13.99
## 2     #september11     88739               0.0962                   9.62
## 3      #twintowers     28168               0.1632                  16.32
## 4  #ChineseNewYear     22466               0.3613                  36.13
## 5     #bankholiday      7862               0.1171                  11.71
## 6           #sleep     36139               0.0365                   3.65
##                                                        Event     Event.Type
## 1 Helicopter crashes into crane in central London (16th Jan)       accident
## 2                            September 11th 2013 anniversary    anniversary
## 3                            September 11th 2013 anniversary    anniversary
## 4                            Chinese New Year, 31st Jan 2014 cultural event
## 5                     Bankholiday - public holiday in the UK     daily life
## 6                                   An eight day long period     daily life
##             Time.Period Sarcasm Irony Humour
## 1   16 Jan-17th Jan 2013       2     2      0
## 2 11th Sep-12th Sep 2013       0     0      0
## 3 11th Sep-12th Sep 2013       0     0      1
## 4  31st Jan-1st Feb 2014       0     0      1
## 5          24th May 2013       0     0      1
## 6 23rd Oct-31st Oct 2013       0     0      1
```

```
tail(Event)
```

```
##             Dataset Total..N. Emotional.Tweets.... Emotional.Tweets.....1
## 23             #NSA    381402               0.0508                   5.08
## 24           #prism    106432               0.0496                   4.96
## 25        Horsemeat     56970               0.0747                   7.47
## 26 #ClosingCeremony     87943               0.1155                  11.55
## 27     #paralympics     27993               0.1397                  13.97
## 28        #woolwich     98969               0.1263                  12.63
##                                                                         Even
t
## 23 National Security Agency PRISM surveillance program (initially leaked early Ju
n)
## 24 National Security Agency PRISM surveillance program (initially leaked early Ju
n)
## 25                     Horsemeat missold as beef (issue came to light on 15th Ja
n)
## 26                                         London 2012 Olympics - Closing ceremon
y
## 27                    London 2012 Olympics - Paralympic games (29th Aug - 9th Se
p)
## 28              Attack and murder of Drummer Lee Rigby in Woolwich, by extremist
s
##                  Event.Type            Time.Period Sarcasm Irony Humour
## 23                  scandal 13th Jun-15th Jul 2013       0     0      0
## 24                  scandal 13th Jun-15th Jul 2013       1     1      1
## 25                  scandal 16th Jan-18th Jan 2013       3     0     10
## 26              sport event 12th Aug-17th Aug 2012       0     0      3
## 27              sport event   4th Sep-6th Sep 2012       0     0      0
## 28 terror incident / murder  23rd May-24thMay 2013       1     1      0
```

```
summary(Event)
```

```
##    Dataset            Total..N.      Emotional.Tweets....
##  Length:28          Min.   :  1047   Min.   :0.03650
##  Class :character   1st Qu.: 11506   1st Qu.:0.05050
##  Mode  :character   Median : 30422   Median :0.09045
##                     Mean   : 56082   Mean   :0.12044
##                     3rd Qu.: 88142   3rd Qu.:0.14572
##                     Max.   :381402   Max.   :0.37910
##  Emotional.Tweets.....1    Event             Event.Type
##  Min.   : 3.650        Length:28          Length:28
##  1st Qu.: 5.050        Class :character   Class :character
##  Median : 9.045        Mode  :character   Mode  :character
##  Mean   :12.044
##  3rd Qu.:14.572
##  Max.   :37.910
##  Time.Period           Sarcasm        Irony            Humour
##  Length:28          Min.   :0.0   Min.   :0.0000   Min.   : 0
##  Class :character   1st Qu.:0.0   1st Qu.:0.0000   1st Qu.: 0
##  Mode  :character   Median :0.0   Median :0.0000   Median : 0
##                     Mean   :0.5   Mean   :0.2143   Mean   : 1
##                     3rd Qu.:1.0   3rd Qu.:0.0000   3rd Qu.: 1
##                     Max.   :3.0   Max.   :2.0000   Max.   :10
```

```
str(Event) #lots of characters: they are quite awkward
```

```
## 'data.frame':    28 obs. of  10 variables:
##  $ Dataset               : chr  "helicopter crash" "#september11" "#twintowers" "#
ChineseNewYear" ...
##  $ Total..N.             : int  25387 88739 28168 22466 7862 36139 79253 11975 906
03 108794 ...
##  $ Emotional.Tweets....  : num  0.1399 0.0962 0.1632 0.3613 0.1171 ...
##  $ Emotional.Tweets.....1: num  13.99 9.62 16.32 36.13 11.71 ...
##  $ Event                 : chr  "Helicopter crashes into crane in central London
(16th Jan)" "September 11th 2013 anniversary" "September 11th 2013 anniversary" "Chin
ese New Year, 31st Jan 2014" ...
##  $ Event.Type            : chr  "accident" "anniversary" "anniversary" "cultural e
vent" ...
##  $ Time.Period           : chr  "16 Jan-17th Jan 2013" "11th Sep-12th Sep 2013" "1
1th Sep-12th Sep 2013" "31st Jan-1st Feb 2014" ...
##  $ Sarcasm               : int  2 0 0 0 0 0 0 0 0 0 ...
##  $ Irony                 : int  2 0 0 0 0 0 0 1 0 0 ...
##  $ Humour                : int  0 0 1 1 1 1 1 0 0 1 ...
```

```
dim(Event) #28 rows/observations and 10 variables
```

```
## [1] 28 10
```

```
colSums(is.na(Event)) ##are there any NAs? No
```

```
##                Dataset              Total..N.   Emotional.Tweets....
##                      0                      0                      0
## Emotional.Tweets.....1                  Event             Event.Type
##                      0                      0                      0
##            Time.Period                Sarcasm                  Irony
##                      0                      0                      0
##                 Humour
##                      0
```

The data has 28 observations and 10 variables in place. It also has only two type of variables, characters and integers, which will make our analysis a bit awkward, particularly the characters. I thus clean this up by turning characters into factors.

```
##Re-naming columns more appropriately
colnames(Event)[2] = 'Total_Tweets'
colnames(Event)[3] = 'Emotional_Tweets'
colnames(Event)[4] = 'Emotional_Tweets_Percentage'
colnames(Event)[5] = 'Type'

# Converting all 'character' variables to 'factor'
Event <- as.data.frame(unclass(Event),
                       stringsAsFactors = TRUE)
Event$Total_Tweets <- as.numeric(Event$Total_Tweets) ##setting as numeric rather than
integer

str(Event) ##Checking they are now factors
```

```
## 'data.frame':    28 obs. of  10 variables:
##  $ Dataset                    : Factor w/ 28 levels "'Daniel Pelka'",..: 26 16 20
4 3 17 18 9 23 28 ...
##  $ Total_Tweets               : num  25387 88739 28168 22466 7862 ...
##  $ Emotional_Tweets           : num  0.1399 0.0962 0.1632 0.3613 0.1171 ...
##  $ Emotional_Tweets_Percentage: num  13.99 9.62 16.32 36.13 11.71 ...
##  $ Type                       : Factor w/ 22 levels "39th G8 Summit in UK on 17th-
18th June",..: 11 22 22 6 5 2 2 8 10 9 ...
##  $ Event.Type                 : Factor w/ 15 levels "accident","anniversary",..: 1
2 2 3 4 4 4 5 5 5 ...
##  $ Time.Period                : Factor w/ 20 levels "11th Jan-15th Jan 2014",..: 6
2 2 16 15 14 14 12 1 18 ...
##  $ Sarcasm                    : int  2 0 0 0 0 0 0 0 0 0 ...
##  $ Irony                      : int  2 0 0 0 0 0 0 1 0 0 ...
##  $ Humour                     : int  0 0 1 1 1 1 1 0 0 1 ...
```

```
head(Event) ##ensuring new column names are correct
```

```
##            Dataset Total_Tweets Emotional_Tweets Emotional_Tweets_Percentage
## 1 helicopter crash        25387           0.1399                       13.99
## 2     #september11        88739           0.0962                        9.62
## 3      #twintowers        28168           0.1632                       16.32
## 4  #ChineseNewYear        22466           0.3613                       36.13
## 5     #bankholiday         7862           0.1171                       11.71
## 6           #sleep        36139           0.0365                        3.65
##                                                   Type     Event.Type
## 1 Helicopter crashes into crane in central London (16th Jan)       accident
## 2                           September 11th 2013 anniversary    anniversary
## 3                           September 11th 2013 anniversary    anniversary
## 4                           Chinese New Year, 31st Jan 2014 cultural event
## 5                    Bankholiday - public holiday in the UK     daily life
## 6                                  An eight day long period     daily life
##              Time.Period Sarcasm Irony Humour
## 1   16 Jan-17th Jan 2013       2     2      0
## 2 11th Sep-12th Sep 2013       0     0      0
## 3 11th Sep-12th Sep 2013       0     0      1
## 4  31st Jan-1st Feb 2014       0     0      1
## 5          24th May 2013       0     0      1
## 6 23rd Oct-31st Oct 2013       0     0      1
```

As above.

3 Analysis

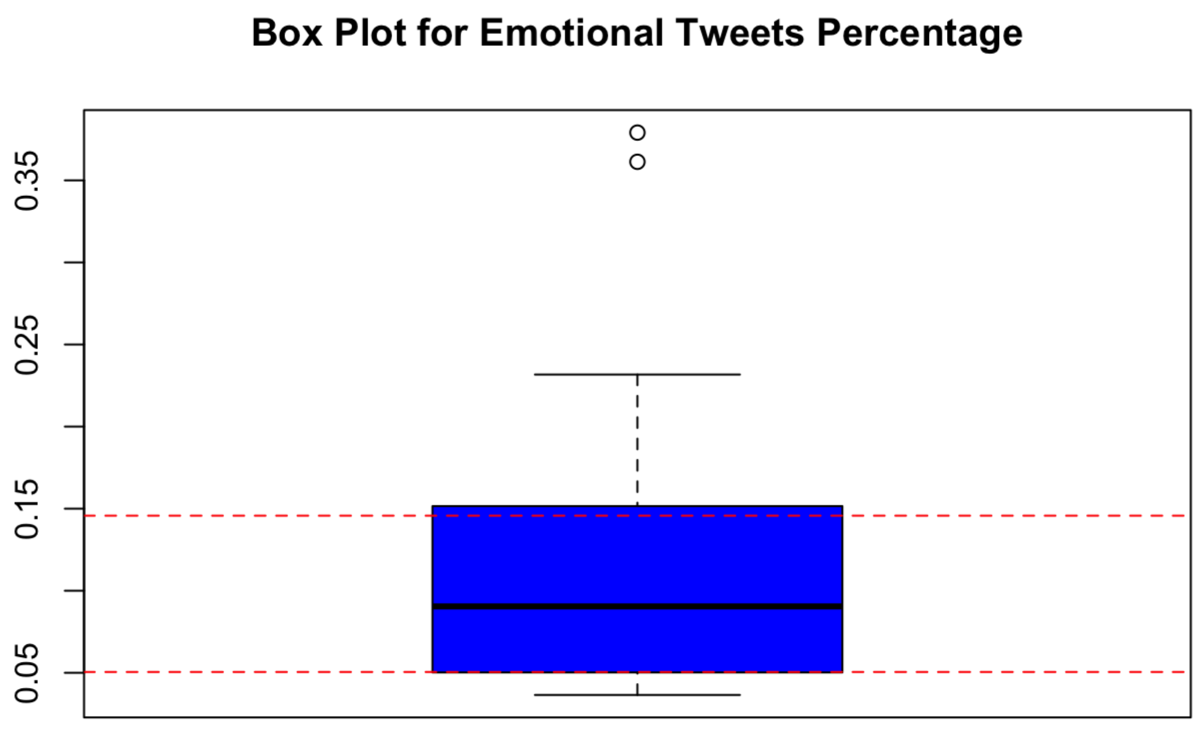
In this section, I will first implement some visuals to interpret our data. I will then construct a for-loop to look at the proportions, and then some regressions to infer my dependent variables of interest.

3.1 Data Visualisation

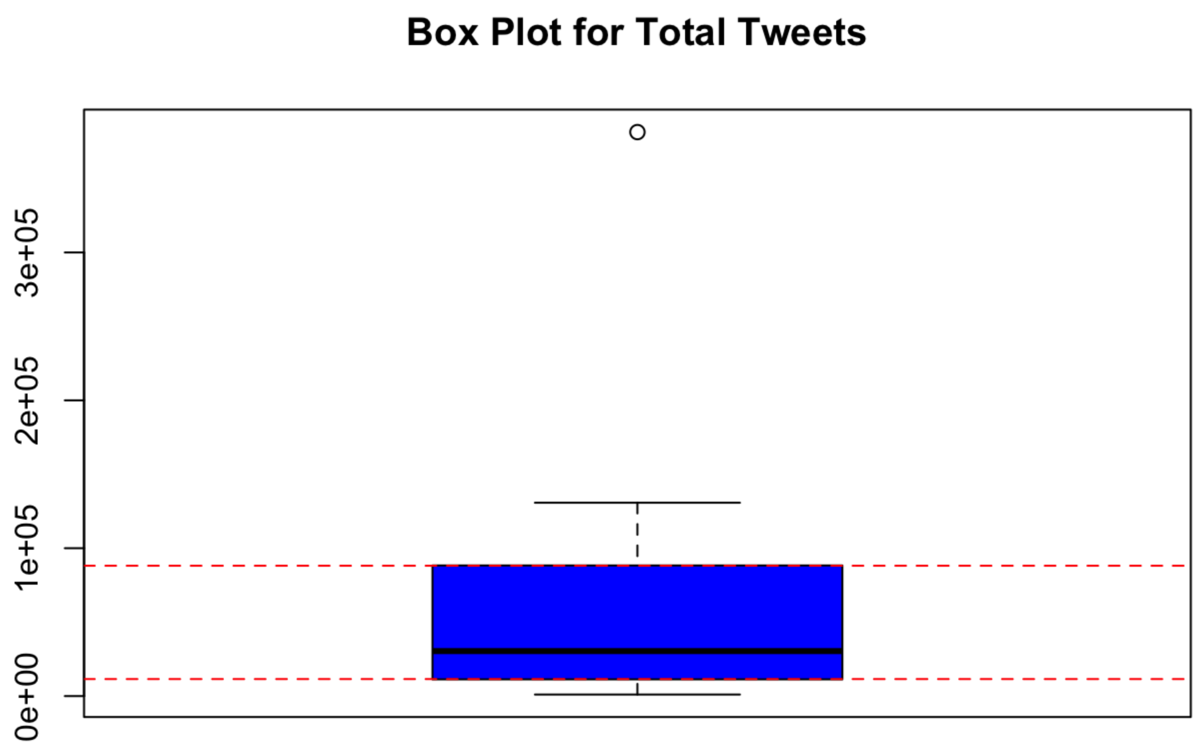
3.1.1 Boxplots

As aforementioned, beginning with the visualisation of my data:

```
par(mfrow = c(1,1))
boxplot(Event$Emotional_Tweets, col = "blue", main = "Box Plot for Emotional Tweets P
ercentage")
abline(h=quantile(Event$Emotional_Tweets,0.25), col="red", lty=2)
abline(h=quantile(Event$Emotional_Tweets,0.75), col="red", lty=2)
```



```
boxplot(Event$Total_Tweets, col = "blue", main = "Box Plot for Total Tweets")
abline(h=quantile(Event$Total_Tweets,0.25), col="red", lty=2)
abline(h=quantile(Event$Total_Tweets,0.75), col="red", lty=2)
```

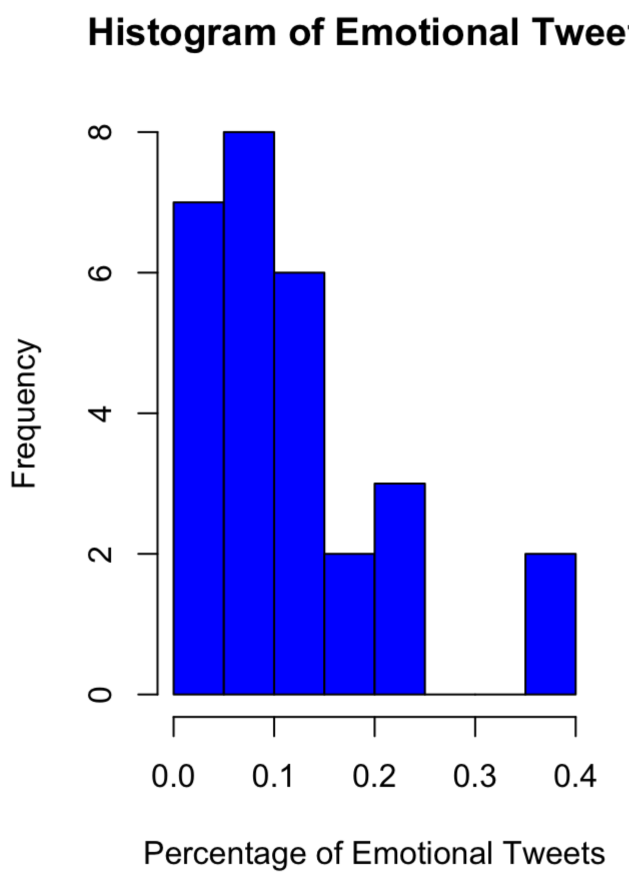


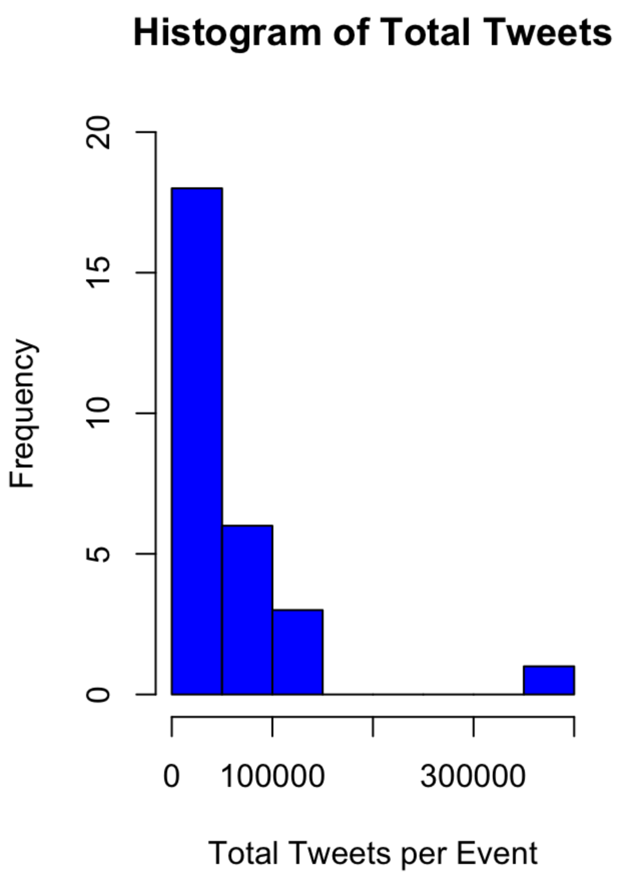
I begin with boxplots, as it is quite robust to dataset of small sample sizes. We can see that two of the plots have pretty egregious outliers. Moreover, the first to third quantile don't have much of a gap. We can loosely say that they are clustered around the one area. As with the minimum and maximum, it provides us a good spread which allow us to have a broader confidence interval to infer from.

3.1.2 Histograms

Now looking at histograms:

```
par(mfrow = c(1,2))
hist(Event$Emotional_Tweets, col = "blue",
     xlab = "Percentage of Emotional Tweets", main = "Histogram of Emotional Tweets")
options(scipen=3)
hist(Event$Total_Tweets, col = "blue",
     xlab = "Total Tweets per Event", main = "Histogram of Total Tweets",
     ylim =c(0,20))
```





Rather than being normally distributed, we can see that the bar charts are skewed to the left. This implies that it should be quite rare for an event to have over x amount of emotional tweets and total tweets about the events: it is loosely speaking, bounded and thus again, loosely speaking, as aforementioned, “clustered around the one area”.

3.1.3 Emotions

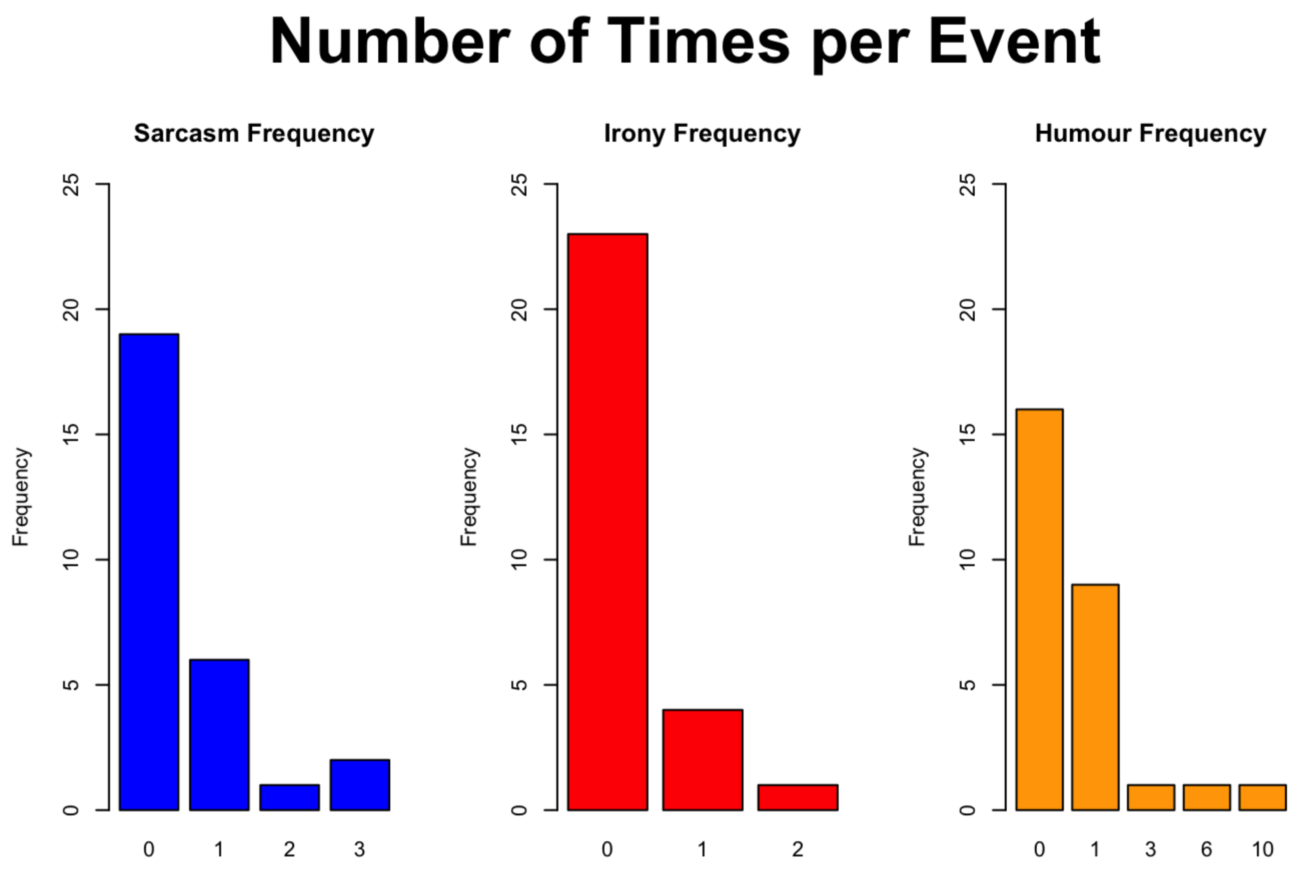
It is quite complex to look at the categorical variables (events) as there are too many levels. As a result, it would be difficult to infer from these graphs.

However, I can turn the emotion variables into factors because they are essentially yes/no variables. In the following, I thus graph the emotion variables to see how often each feeling for events.

```
#changing plot layout so all three graphs could fit on one plot
par(mfrow = c(1,3))
par(oma=c(0,0,4,0))

##changing variables as factor to be able to graph it
Event$Sarcasm <- as.factor(Event$Sarcasm)
Event$Irony <- as.factor(Event$Irony)
Event$Humour <- as.factor(Event$Humour)

##graphing feelings
plot(Event$Sarcasm, col = "blue", ylim = c(0,25), ylab = "Frequency",
     main = "Sarcasm Frequency")
plot(Event$Irony, col = "red", ylim = c(0,25), ylab = "Frequency",
     main = "Irony Frequency")
plot(Event$Humour, col = "orange", ylim = c(0,25), ylab = "Frequency",
     main = "Humour Frequency")
mtext(~bold("Number of Times per Event"), # Add main title
      side = 3,
      line = 0,
      cex = 2,
      outer = TRUE)
```



```
##Setting back to original layout
par(mfrow = c(1,1))
par(oma=c(0,0,0,0))
```

We can see that when it comes to tweeting about events, it is more often than not that users tend to not use these feelings. There are two possible reasons to this: 1) we have more than three feelings and 2) the person who collected this data may found the feeling of the tweet a bit ambiguous and probably allocated it incorrectly. Nevertheless, we have derived a glimpse of the distribution of feelings.

3.2 Proportion

I now want to look at the proportion of each emotions: how much each emotions represent out of the total. To do this, I must get a total of them all first.

```
##cumulative number of tweets regarding emotions
emotion_total <- sum(Sarcasm) + sum(Irony) + sum(Humour)
```

I then create a small for-loop to implement the sum. I then do a robustness check to ensure that my results are correct through a simple ‘sum’ function (for summation) and divide it by the total.

```
proportion <- function(x){
  sum<-0
  for(i in 1:(length(x))){ ##beginning from first observation
    sum=sum+x[i] ##using a loop to sum first obs with the 0, and then the next one, a
nd so on
  }
  proportion = sum/emotion_total ##grab for loop results and find proportion
  return(proportion)
}
proportion(Sarcasm)
```

```
## [1] 0.2916667
```

```
proportion(Irony)
```

```
## [1] 0.125
```

```
proportion(Humour)
```

```
## [1] 0.5833333
```

```
##ensuring my above are correct
sum(Sarcasm)/emotion_total
```

```
## [1] 0.2916667
```

```
sum(Irony)/emotion_total
```

```
## [1] 0.125
```

```
sum(Humour)/emotion_total
```

```
## [1] 0.5833333
```

Thus, we see humorous tweets account for the most, followed by sarcasm, and then irony. This makes sense, perhaps subjectively, as people tend to ordinal emotions with respect to time: people are more humorous than sarcastic, and more sarcastic to ironic.

3.3 Regression

This section works through some regressions.

3.3.1 Linear Regression

I first implement a linear regression model and ask: can the quantity of emotional tweets predict the number of tweets of an event? I first create dummies for the three variables of feelings in order to make the regression work.

```
##Creating Dummies to make Regression work
Event$SarcasmDummy <- ifelse(Event$Sarcasm != 0, 1, 0)
Event$IronyDummy <- ifelse(Event$Irony != 0, 1, 0)
Event$HumourDummy <- ifelse(Event$Humour != 0, 1, 0)
```

I then ensure there is no multicollinearity for the variables I am using for my regression as this can create a higher standard error and thus deliver improper conclusions.

```
##subsetting dataset to have a look at correlations: ensure no multicolliinearity
EventNumeric <- dplyr::select_if(Event, is.numeric)
head(EventNumeric) ##looking at top few observations to ensure everything is correct
```

```
##   Total_Tweets Emotional_Tweets Emotional_Tweets_Percentage SarcasmDummy
## 1        25387           0.1399                       13.99            1
## 2        88739           0.0962                        9.62            0
## 3        28168           0.1632                       16.32            0
## 4        22466           0.3613                       36.13            0
## 5         7862           0.1171                       11.71            0
## 6        36139           0.0365                        3.65            0
##   IronyDummy HumourDummy
## 1          1           0
## 2          0           0
## 3          0           1
## 4          0           1
## 5          0           1
## 6          0           1
```

```
EventNumeric <- EventNumeric[,-2] ##can eliminate one of the emotional_tweets: pretty
arbitary
head(EventNumeric) ##ensuring it worked
```

```
##   Total_Tweets Emotional_Tweets_Percentage SarcasmDummy IronyDummy HumourDummy
## 1        25387                       13.99            1          1           0
## 2        88739                        9.62            0          0           0
## 3        28168                       16.32            0          0           1
## 4        22466                       36.13            0          0           1
## 5         7862                       11.71            0          0           1
## 6        36139                        3.65            0          0           1
```

```
str(EventNumeric) ##checking structure
```

```
## 'data.frame':    28 obs. of  5 variables:
##  $ Total_Tweets               : num  25387 88739 28168 22466 7862 ...
##  $ Emotional_Tweets_Percentage: num  13.99 9.62 16.32 36.13 11.71 ...
##  $ SarcasmDummy               : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ IronyDummy                 : num  1 0 0 0 0 0 0 1 0 0 ...
##  $ HumourDummy                : num  0 0 1 1 1 1 1 0 0 1 ...
```

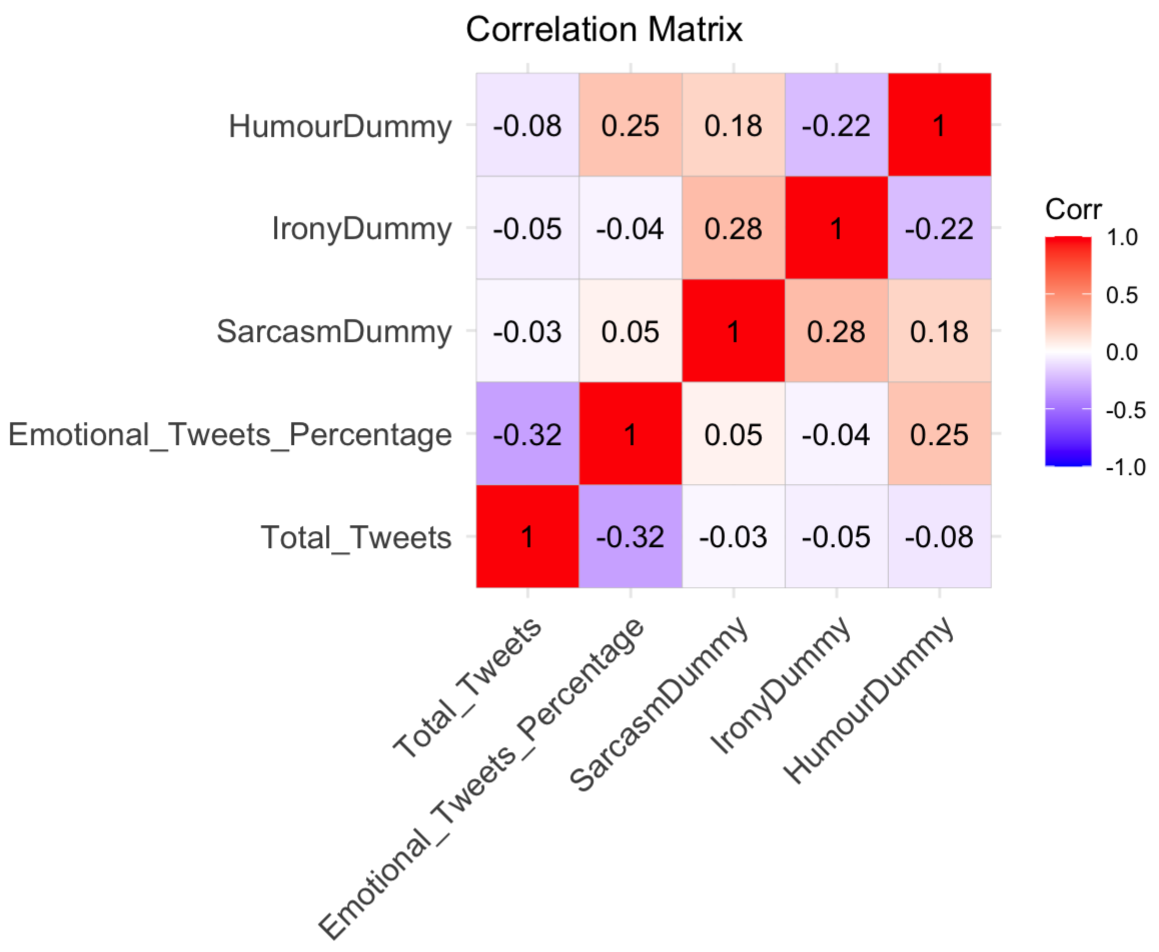
```
dim(EventNumeric) ##dimensions
```

```
## [1] 28  5
```

```
rcorr(as.matrix(EventNumeric)) ##correlation matrix
```

```
##                             Total_Tweets Emotional_Tweets_Percentage
## Total_Tweets                        1.00                       -0.32
## Emotional_Tweets_Percentage        -0.32                        1.00
## SarcasmDummy                       -0.03                        0.05
## IronyDummy                         -0.05                       -0.04
## HumourDummy                        -0.08                        0.25
##                             SarcasmDummy IronyDummy HumourDummy
## Total_Tweets                       -0.03      -0.05       -0.08
## Emotional_Tweets_Percentage         0.05      -0.04        0.25
## SarcasmDummy                        1.00       0.28        0.18
## IronyDummy                          0.28       1.00       -0.22
## HumourDummy                         0.18      -0.22        1.00
##
## n= 28
##
##
## P
##                             Total_Tweets Emotional_Tweets_Percentage
## Total_Tweets                             0.0999
## Emotional_Tweets_Percentage 0.0999
## SarcasmDummy                0.8988       0.8096
## IronyDummy                  0.8139       0.8221
## HumourDummy                 0.6695       0.1990
##                             SarcasmDummy IronyDummy HumourDummy
## Total_Tweets                0.8988       0.8139     0.6695
## Emotional_Tweets_Percentage 0.8096       0.8221     0.1990
## SarcasmDummy                             0.1519     0.3687
## IronyDummy                  0.1519                  0.2711
## HumourDummy                 0.3687       0.2711
```

```
eventnumeric_cor <- cor(EventNumeric)
ggcorrplot(eventnumeric_cor,
           title = "Correlation Matrix",
           lab = TRUE) ##correlation plot
```



As we can see, there is not a high correlation amongst the variables, ensuring multicollinearity will not be present.

We thus run the regression where we are trying to understand which variable has a significant effect on the percentage of emotional tweets:

```
##Running Regression
lm1 <- lm(Emotional_Tweets_Percentage ~ Total_Tweets + SarcasmDummy +
            IronyDummy + HumourDummy, Event)
bptest(lm1) ##Heteroskedasticity test
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lm1
## BP = 3.8914, df = 4, p-value = 0.4209
```

```
dwtest(lm1) ##autocorrelation test
```

```
##
##  Durbin-Watson test
##
## data:  lm1
## DW = 1.5321, p-value = 0.07792
## alternative hypothesis: true autocorrelation is greater than 0
```

```
vif(lm1) ##all less than the rule of thumb of 5: multicollinearity not present
```

```
## Total_Tweets SarcasmDummy   IronyDummy  HumourDummy
##     1.011796     1.157572     1.181108     1.130384
```

```
summary(lm1) ##no interesting results: not statistically significant
```

```
##
## Call:
## lm(formula = Emotional_Tweets_Percentage ~ Total_Tweets + SarcasmDummy +
##     IronyDummy + HumourDummy, data = Event)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -11.440  -6.562  -1.275   4.120  21.460
##
## Coefficients:
##                Estimate  Std. Error t value Pr(>|t|)
## (Intercept)  12.39685189  3.02924111   4.092 0.000447 ***
## Total_Tweets -0.00003657  0.00002363  -1.547 0.135418
## SarcasmDummy  0.08264583  3.96649361   0.021 0.983556
## IronyDummy   -0.27825312  4.88569815  -0.057 0.955075
## HumourDummy   4.01505228  3.69907545   1.085 0.288982
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.111 on 23 degrees of freedom
## Multiple R-squared:  0.1511, Adjusted R-squared:  0.003461
## F-statistic: 1.023 on 4 and 23 DF,  p-value: 0.4161
```

We passed both heteroskedasticity and autocorrelation tests: these are not present as we have failed to reject the null for each. Variance Inflation Factor for multiocollinearity is less than 5: not present. (I noticed that the standard errors are astronomically high, despite passing my checks: I'm not sure what is going on).

We also found nothing significant from the regression: there is no strong causation and ceterus paribus in our model and thus weakens our analysis. Again, this is more than likely due to a small sample size. To avoid dissappointing the reader however, the following section, Paclage, will provide much more interesting results with respect to linear regression and emotional tweets percentage.

3.3.2 Poisson Regression

Let's look for an alternative regression: the Poisson Regression, where the requirement is for our dependent variable to be a count variable. Thus, we will use Total_Tweets as our dependent variable, which was hinted in the prior when showing graphs. But first, as an unspoken rule for econometrics, we always use OLS first to get a possible overview:

```
lm2 <- lm(Total_Tweets ~ Emotional_Tweets_Percentage +SarcasmDummy +
                  IronyDummy + HumourDummy, data = Event) ##linear regression
poisson1 <- glm(Total_Tweets ~ Emotional_Tweets_Percentage +SarcasmDummy +
                  IronyDummy + HumourDummy, data = Event, family = "poisson") ##poiss
on regression
stargazer(lm2, poisson1,
          title = "OLS vs Poisson",
          dep.var.labels.include = FALSE,
          dep.var.caption  = "Dependent Variable: Total Tweets",
          type = "text")
```

```
##
## OLS vs Poisson
## ==============================================================
##                              Dependent Variable: Total Tweets
##                             ----------------------------------
##                                    OLS              Poisson
##                                    (1)                (2)
## --------------------------------------------------------------
## Emotional_Tweets_Percentage      -2,578.387        -0.071***
##                                 (1,666.272)        (0.0001)
##
## SarcasmDummy                     2,054.248         -0.022***
##                                 (33,303.520)        (0.002)
##
## IronyDummy                      -13,227.400        -0.141***
##                                 (40,934.440)        (0.002)
##
## HumourDummy                      -3,355.018        -0.050***
##                                 (31,838.390)        (0.002)
##
## Constant                        90,274.840***       11.688***
##                                 (27,636.490)        (0.002)
##
## --------------------------------------------------------------
## Observations                         28                28
## R2                                 0.105
## Adjusted R2                        -0.051
## Log Likelihood                                   -729,825.900
## Akaike Inf. Crit.                                1,459,662.000
## Residual Std. Error         76,500.980 (df = 23)
## F Statistic                  0.673 (df = 4; 23)
## ==============================================================
## Note:                              *p<0.1; **p<0.05; ***p<0.01
```

For OLS, no independent variables are statistically significant. Perhaps because we are implying the wrong model. For Poisson, we thus see that all variables are statistically significant at the one percent level.

To interpret these coefficients, we say that given a 1% increase in emotional tweets, we can expect total tweets to fall by 7.1%, holding all else constant. Alternatively, if emotional tweets fall by 1%, total tweets increases by 7.1%. This may be due to the fact people want to avoid commotion and tweeting things that could get them in trouble.

Thus, rather than using descrpitive statistics to draw ‘robust’ conclusions, we have used inferential regressions for this instead, a pretty interesting one for that matter.

4 Package

The package I have selected is ‘lmtest’.

The purpose of this package is used mainly for regression purposes. I don’t know about the statistician, but we econometricians are particularly conscientious on the rules of our model. lmtest has aided me with some of these: in the previous section, I used bptest (Breusch-Pagan) to test for heteroskedasticity and dwtest for autocorrelation (In which I don’t think base R has a function for any of these), and a few more I will show in this section.

4.1 RESET Test: reset()

Let me run the first linear regression again, but I want to ensure the specification (loosely speaking - form) of my model is correct. I thus use a Ramsey RESET Test from this package.

```
library(lmtest)
reset(Emotional_Tweets_Percentage ~ Total_Tweets + SarcasmDummy +
        IronyDummy + HumourDummy, data = Event, power = 2:3)
```

```
##
##  RESET test
##
## data:  Emotional_Tweets_Percentage ~ Total_Tweets + SarcasmDummy + IronyDummy +
HumourDummy
## RESET = 3.497, df1 = 2, df2 = 21, p-value = 0.04888
```

Where the null says our model is not misspecified and the alternative says it is, but we don’t know how ie what interaction terms or squares etc. We thus reject the null that our model is not misspecified.

Let’s try it with Total_Tweets being squared (an inflection point for Total_Tweets):

```
Total_Tweets2 <- Event$Total_Tweets^2
reset(Emotional_Tweets_Percentage ~ Total_Tweets + Total_Tweets2 + SarcasmDummy +
        IronyDummy + HumourDummy, data = Event, power = 2:3)
```

```
##
##  RESET test
##
## data:  Emotional_Tweets_Percentage ~ Total_Tweets + Total_Tweets2 +     SarcasmDum
my + IronyDummy + HumourDummy
## RESET = 3.1079, df1 = 2, df2 = 20, p-value = 0.06678
```

Interesting. This is our ideal model according to this test. Let’s run our regression now:

```
lm1 <- lm(Emotional_Tweets_Percentage ~ Total_Tweets + Total_Tweets2 + SarcasmDummy +
            IronyDummy + HumourDummy, Event)
summary(lm1)
```

```
## 
## Call:
## lm(formula = Emotional_Tweets_Percentage ~ Total_Tweets + Total_Tweets2 + 
##     SarcasmDummy + IronyDummy + HumourDummy, data = Event)
## 
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -11.5907  -6.5763   0.7341   4.0618  19.3107 
## 
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)    
## (Intercept)    1.447e+01  3.180e+00   4.550 0.000158 ***
## Total_Tweets  -1.322e-04  6.226e-05  -2.123 0.045281 *  
## Total_Tweets2  2.850e-10  1.728e-10   1.650 0.113188    
## SarcasmDummy   8.646e-01  3.855e+00   0.224 0.824615    
## IronyDummy     3.626e-01  4.728e+00   0.077 0.939572    
## HumourDummy    5.178e+00  3.637e+00   1.424 0.168534    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 8.788 on 22 degrees of freedom
## Multiple R-squared:  0.2446, Adjusted R-squared:  0.07287 
## F-statistic: 1.424 on 5 and 22 DF,  p-value: 0.2546
```

We can now see Total_Tweets is statistically significant at the 5% level. However, the coefficient is quite small indicating low economic explanatory power. Nevertheless, we have showed that our previous specification was wrong and with the assist of the RESET Test, we have constructed a better model.

4.2 Autocorrelation: dwtest() and bgtest()

Let's again test for autocorrelation with dwtest function from this package:

```
dwtest(lm1)
```

```
## 
##  Durbin-Watson test
## 
## data:  lm1
## DW = 1.457, p-value = 0.05344
## alternative hypothesis: true autocorrelation is greater than 0
```

We fail to reject the null that autocorrelation is not present.

And also with the Breusch-Godfrey test for serial correlation in the error terms:

```
bgtest(lm1)
```

```
## 
##  Breusch-Godfrey test for serial correlation of order up to 1
## 
## data:  lm1
## LM test = 2.1152, df = 1, p-value = 0.1458
```

Where we again, fail to reject null that autocorrelation is not present.

4.3 Heteroskedasticity: bptest() and Robust Standard Errors

Now, testing heteroskedasticity with bptest function from the lmtest package.

```
bptest(lm1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lm1
## BP = 7.0455, df = 5, p-value = 0.2173
```

Again, our residuals are homoskedastic. But let’s assume our residuals are heteroskedastic (for the purpose of showcasing this function), which will imply our standard errors being wrong (more specifically, OLS overrejects null when we have heteroskedasticity) and thus interpretation of causal is not possible. We can use robust standard errors from the lmtest package which takes account of heteroskedastic residuals.

```
lm1robust <- coeftest(lm1, function(x) vcovHC(x, type="HC0"))
lm1robust
```

```
##
## t test of coefficients:
##
##                  Estimate  Std. Error t value   Pr(>|t|)
## (Intercept)    1.4466e+01  2.8435e+00  5.0876 0.00004262 ***
## Total_Tweets  -1.3216e-04  5.3333e-05 -2.4780    0.02137 *
## Total_Tweets2  2.8503e-10  1.2817e-10  2.2238    0.03674 *
## SarcasmDummy   8.6460e-01  3.7974e+00  0.2277    0.82200
## IronyDummy     3.6258e-01  3.6919e+00  0.0982    0.92266
## HumourDummy    5.1782e+00  3.3590e+00  1.5416    0.13743
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(lm1)
```

```
##
## Call:
## lm(formula = Emotional_Tweets_Percentage ~ Total_Tweets + Total_Tweets2 +
##     SarcasmDummy + IronyDummy + HumourDummy, data = Event)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.5907  -6.5763   0.7341   4.0618  19.3107
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.447e+01  3.180e+00   4.550 0.000158 ***
## Total_Tweets  -1.322e-04  6.226e-05  -2.123 0.045281 *
## Total_Tweets2  2.850e-10  1.728e-10   1.650 0.113188
## SarcasmDummy   8.646e-01  3.855e+00   0.224 0.824615
## IronyDummy     3.626e-01  4.728e+00   0.077 0.939572
## HumourDummy    5.178e+00  3.637e+00   1.424 0.168534
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.788 on 22 degrees of freedom
## Multiple R-squared:  0.2446, Adjusted R-squared:  0.07287
## F-statistic: 1.424 on 5 and 22 DF,  p-value: 0.2546
```

```
stargazer(lm1, lm1robust,
          title = "OLS vs Robust Standard Errors",
          dep.var.labels.include = FALSE,
          column.labels=c("OLS","Robust Standard Errors"),
          dep.var.caption  = "Dependent Variable: Emotional Tweets Percentage",
          type = "text", model.names = FALSE)
```

```
##
## OLS vs Robust Standard Errors
## ===================================================================
##                     Dependent Variable: Emotional Tweets Percentage
##                     -----------------------------------------------
##                               OLS           Robust Standard Errors
##                               (1)                     (2)
## -------------------------------------------------------------------
## Total_Tweets               -0.0001**               -0.0001**
##                             (0.0001)                (0.0001)
##
## Total_Tweets2                0.000                  0.000**
##                             (0.000)                 (0.000)
##
## SarcasmDummy                 0.865                   0.865
##                             (3.855)                 (3.797)
##
## IronyDummy                   0.363                   0.363
##                             (4.728)                 (3.692)
##
## HumourDummy                  5.178                   5.178
##                             (3.637)                 (3.359)
##
## Constant                   14.466***               14.466***
##                             (3.180)                 (2.843)
##
## -------------------------------------------------------------------
## Observations                   28
## R2                           0.245
## Adjusted R2                  0.073
## Residual Std. Error     8.788 (df = 22)
## F Statistic           1.424 (df = 5; 22)
## ===================================================================
## Note:                                   *p<0.1; **p<0.05; ***p<0.01
```

As a result, we can see that Total_Tweets^2 is now also significant, although with very, very low explanatory power. Furthermore, we can see that all the other standard errors has fallen but we are taking account of heteroskedasticity.

All in all, we economists use lmtest package a lot in order to avoid, or take account of the violations we have made.

5 Function

For this section, we are required to create some functions, or S3 classes from what we learned in class. The requirements are to 'extend' print, summary and plot functions. I first turn my dataset into a list:

```
attach(EventNumeric)
EventNumeric <- as.list(EventNumeric) ##creating EventNumeric as 'list' just like was
taught in class
class(EventNumeric) <- "events"
is.list(EventNumeric) ## did my list function really work?
```

```
## [1] TRUE
```

```
class(EventNumeric) ##ensuring it is now a list called 'events'
```

```
## [1] "events"
```

```
str(EventNumeric)
```

```
## List of 5
##  $ Total_Tweets                : num [1:28] 25387 88739 28168 22466 7862 ...
##  $ Emotional_Tweets_Percentage: num [1:28] 13.99 9.62 16.32 36.13 11.71 ...
##  $ SarcasmDummy               : num [1:28] 1 0 0 0 0 0 0 0 0 0 ...
##  $ IronyDummy                 : num [1:28] 1 0 0 0 0 0 0 1 0 0 ...
##  $ HumourDummy                : num [1:28] 0 0 1 1 1 1 1 0 0 1 ...
##  - attr(*, "class")= chr "events"
```

5.1 Print

For my first function/S3 class, I am going to create a print function, which gives us a very, very, very broad overview of what we are calling, relative to summary function which provides us more information.

```
##Print
print.events <- function(x){
  return(t(do.call(rbind, x))) ##converting list into dataframe and then transposing
it for easier view
}
print(EventNumeric)
```

```
##       Total_Tweets Emotional_Tweets_Percentage SarcasmDummy IronyDummy
##  [1,]        25387                       13.99            1          1
##  [2,]        88739                        9.62            0          0
##  [3,]        28168                       16.32            0          0
##  [4,]        22466                       36.13            0          0
##  [5,]         7862                       11.71            0          0
##  [6,]        36139                        3.65            0          0
##  [7,]        79253                        4.49            0          0
##  [8,]        11975                       18.92            0          1
##  [9,]        90603                        8.18            0          0
## [10,]       108794                       12.51            0          0
## [11,]        11708                       21.54            0          0
## [12,]         4309                        6.75            0          0
## [13,]        41176                        8.47            1          0
## [14,]        43509                        4.27            0          0
## [15,]         1047                        5.44            0          1
## [16,]        10898                       20.43            1          0
## [17,]         1216                       37.91            1          0
## [18,]        10459                       23.17            0          0
## [19,]        32676                        4.24            0          0
## [20,]         8824                        3.90            1          0
## [21,]        14638                        5.70            0          0
## [22,]       130748                        4.22            1          0
## [23,]       381402                        5.08            0          0
## [24,]       106432                        4.96            1          1
## [25,]        56970                        7.47            1          0
## [26,]        87943                       11.55            0          0
## [27,]        27993                       13.97            0          0
## [28,]        98969                       12.63            1          1
##       HumourDummy
##  [1,]           0
##  [2,]           0
##  [3,]           1
##  [4,]           1
##  [5,]           1
##  [6,]           1
##  [7,]           1
##  [8,]           0
##  [9,]           0
## [10,]           1
## [11,]           0
## [12,]           0
## [13,]           1
## [14,]           0
## [15,]           0
## [16,]           1
## [17,]           1
## [18,]           0
## [19,]           0
## [20,]           0
## [21,]           0
## [22,]           0
## [23,]           0
## [24,]           1
## [25,]           1
## [26,]            1
## [27,]            0
## [28,]            0
```

Thus, by implementing a specific print function, we have derived a general overview of the list.

5.2 Summary

Second S3 class, I am going to create a summary of my data. It is nothing grandiloquent: just giving an overview of my data at hand.

```r
##summary
summary.events <- function(x){
  print(length(x[[1]])) ##number of observations
  ##function for mean, median, min, max, quantiles and range
  multiple.func <- function(x) {
    c(min = min(x), mean = mean(x), max = max(x),
      quantile = quantile(x, probs = c(0.25, 0.5, 0.75), range = max(x) - min(x)))
  }
  x <- lapply(x, multiple.func) ##applying this to list
  return(x) ##spitting results out
}
summary(EventNumeric)
```

```
## [1] 28
```

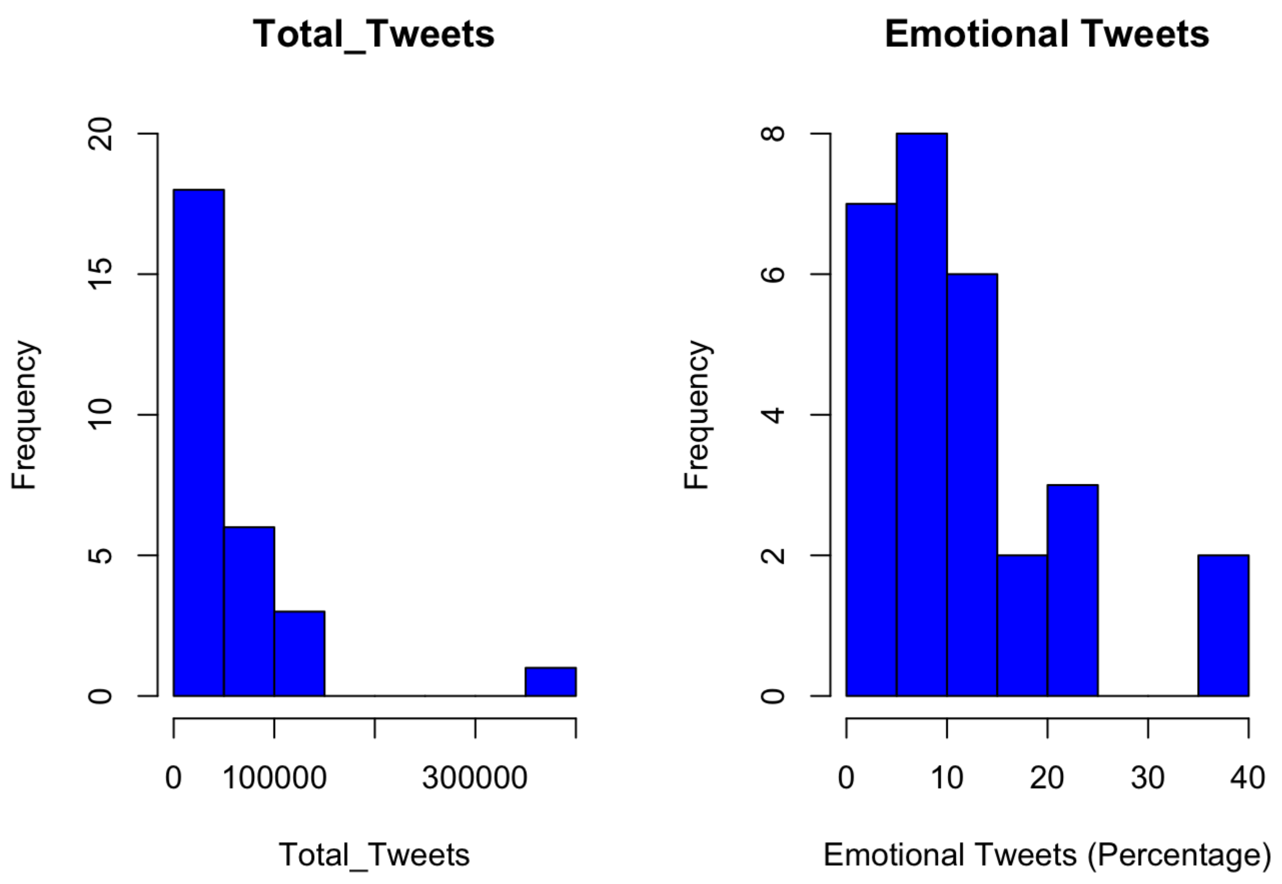
```
## $Total_Tweets
##           min          mean           max quantile.25% quantile.50% quantile.75%
##       1047.00      56082.25     381402.00     11505.50     30422.00     88142.00
##
## $Emotional_Tweets_Percentage
##           min          mean           max quantile.25% quantile.50% quantile.75%
##       3.65000      12.04357      37.91000      5.05000      9.04500     14.57250
##
## $SarcasmDummy
##           min          mean           max quantile.25% quantile.50% quantile.75%
##     0.0000000     0.3214286     1.0000000    0.0000000    0.0000000    1.0000000
##
## $IronyDummy
##           min          mean           max quantile.25% quantile.50% quantile.75%
##     0.0000000     0.1785714     1.0000000    0.0000000    0.0000000    0.0000000
##
## $HumourDummy
##           min          mean           max quantile.25% quantile.50% quantile.75%
##     0.0000000     0.4285714     1.0000000    0.0000000    0.0000000    1.0000000
```

As the results produce, I have successfully implemented an S3 class for this summary.

5.3 Plot

My final one is for graphs, more specifically, a histogram of my two continuous variables because they are much more interesting than the binomial variables.

```
##plot
plot.events <- function(EventNumeric){
  par(mfrow = c(1,2)) #one row two columns
  ##x and y axis titles are personalised, else my axis will look slightly indecorous
  hist(EventNumeric$Total_Tweets, xlab = "Total_Tweets", main =  "Total_Tweets",
       col = "blue", ylim = c(0,20))
  hist(EventNumeric$Emotional_Tweets_Percentage, xlab = "Emotional Tweets (Percentag
e)", main = "Emotional Tweets",
       col = "blue")
  minmax <- function(x){
    c(min = min(x), max = max(x), range = max(x) - min(x)) ##producing an overview of
range
  }
  return(lapply(EventNumeric[1:2], minmax))
}
plot(EventNumeric)
```



```
## $Total_Tweets
##    min    max  range
##   1047 381402 380355
##
## $Emotional_Tweets_Percentage
##   min   max range
##  3.65 37.91 34.26
```

I have also constructed a range element to give an overview of the range of the data, in case one might not be able to intepret the exact amount correctly.

I have implement S3 classes as part of the project instructions. However, I may be naive when I say this, but it seems that S3 classes are only applicable for one dataset and thus not versatile (again, I may be naive). Moreover, the implemented S3 classes was not an “analysis of interest” to me. I therefore create functions that does interest me: regressions.

I want to showcase these functions because prior to beginning of the course, it would be an understatement to say that I was egregious at functions and if-else statements: I want to illustrate my progess.

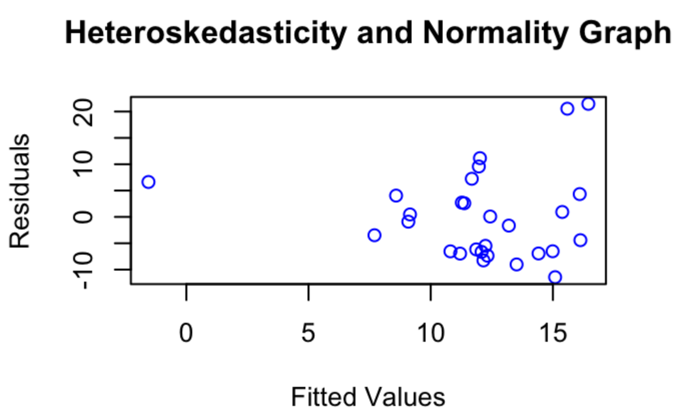
5.4 Extension: Functions for the Economists’ Regression

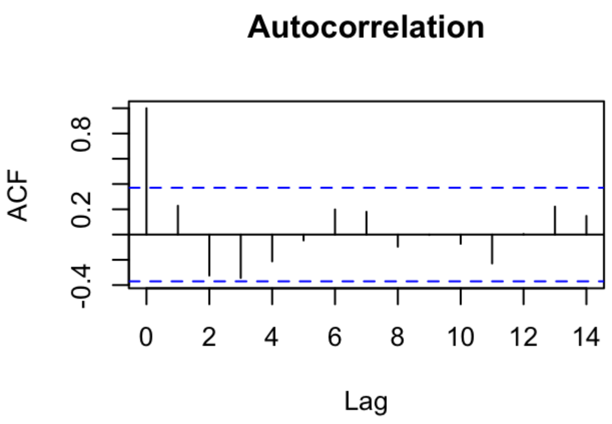
5.4.1 Gauss-Markov

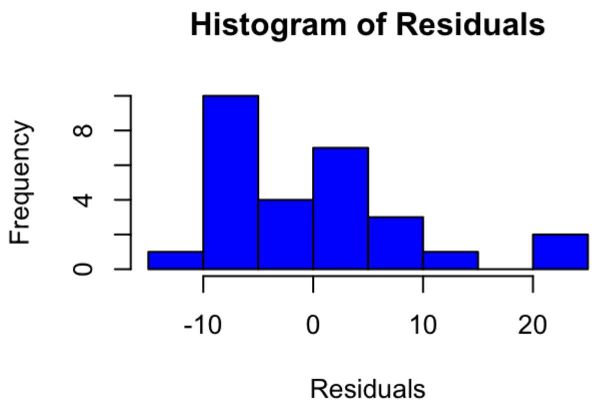
As aforementioned, we economists are quite conscientious in our models, particularly OLS. For OLS, we need our model to be 1) homoskedastic, 2) not serially correlated and 3) normally distributed. I create a quick and dirty function to display my results, and also visualise them. After all, we understand better through graphics.

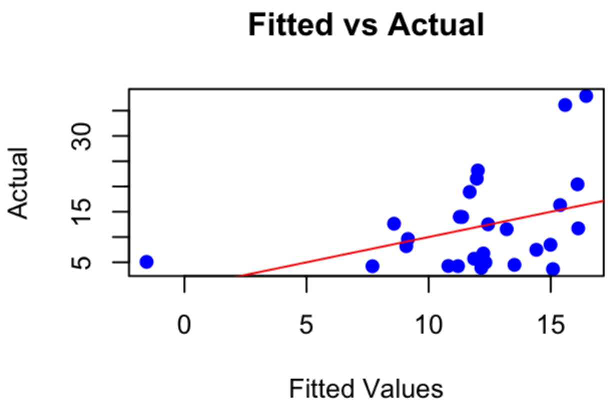
```
#######
GaussMarkov <- function(y, ...){ ##y is dependent variable, '...' is one or more inde
pendent variables
  x <- data.frame(...) ##creating a dataframe
  lm <- lm(y ~ ., data = x) ##running regression
  ehat <- lm$resid ##retrieving residuals, all the unexplained variables agglomerated
into
  yhat <- lm$fitted.values ##predicted values
  print(bptest(lm)) ##heteroskedasticity test
  print(dwtest(lm)) ##autocorrelation test (Durbin Watson)
  print(bgtest(lm)) ##autocorrelation test (Breusch Godfrey)
  print(ols_test_normality(ehat)) ##normality test
  par(mfrow = c(2,2)) ##setting plot layout
  plot(yhat, ehat, col = "blue",
       main = "Heteroskedasticity and Normality Graph",
       xlab = "Fitted Values", ylab = "Residuals") ##Heteroskedasticity and Normality
Graph
  acf(ehat, main = "Autocorrelation") ## Autocorrelation Graph
  hist(ehat, xlab = "Residuals",
       main = "Histogram of Residuals", col = "blue") ##histogram of reesiduals: shou
ld be normally distributed
  plot(yhat, y, main= "Fitted vs Actual",
       xlab = "Fitted Values", ylab = "Actual",
       pch = 19,
       col = "blue")
  abline(a = 0, b = 1,col='red') ##a plot of heteroskedasticity
}
GaussMarkov(Emotional_Tweets_Percentage, SarcasmDummy, Total_Tweets, IronyDummy, Humo
urDummy) ##testing it
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lm
## BP = 3.8914, df = 4, p-value = 0.4209
##
##
##  Durbin-Watson test
##
## data:  lm
## DW = 1.5321, p-value = 0.07792
## alternative hypothesis: true autocorrelation is greater than 0
##
##
##  Breusch-Godfrey test for serial correlation of order up to 1
##
## data:  lm
## LM test = 1.5466, df = 1, p-value = 0.2136
##
## -----------------------------------------------
##        Test             Statistic       pvalue
## -----------------------------------------------
## Shapiro-Wilk              0.8998          0.0113
## Kolmogorov-Smirnov        0.1359          0.6303
## Cramer-von Mises          2.2006          0.0000
## Anderson-Darling          0.8744          0.0217
## -----------------------------------------------
```









The above comments explain clearly what the purpose is and so I want to avoid a tautology. As I said, the purpose is to test whether the Gauss-Markov Assumptions hold, not only through descriptive statistics, but also more robust approaches such as the Breusch Pagan and the Durbin Watson tests.

5.4.2 Best Model?

I now ask whether we should use OLS, Poisson, or Robust Standard Errors OLS given certain conditions. Of course, there are more than these three models, but it would be a good overview given I have used them throughout this assignment. I create a function for this called 'best_model':

```r
best_model <- function(y,...){
  x <- data.frame(...)
  lm <- lm(y ~ ., data = x)
  specify_decimal <- function(x, k) trimws(format(round(x, k), nsmall=k)) ##use funct
ion (more specifically the bptest function) to grab a number that is not considered a
number when we use a function
  BPpvalue <- specify_decimal(bptest(lm)$p.value, 5) ##creating object called BPvalue
and rounding it to 5 decimal points
  if(BPpvalue > 0.05){ ##heteroskedasticity test: we want to fail this test
    print("Failed to reject null that model is homoskedastic")
    print(paste0("Breusch Pagan P-Value is ", BPpvalue))
    pvalue <- as.data.frame(summary(lm)$coefficients[,4])[-1, ]
    ##getting p-values for significance from lm but remove intercept p-value: interce
pt is only there to make equation work
    if(y %% 1 == 0){ ##count variables dont have remainders
        poisson <- glm(y~., data = x, family = "poisson")
        pvalue <- as.data.frame(summary(poisson)$coefficients[,4])[-1, ]
   ##getting pvalues for significance from poisson but remove intercept p-value
        if(pvalue < 0.05){
          summary(poisson)}
      }
    else if (y %% 1 != 0 & pvalue < 0.05) {  ##OLS is versatile: try with OLS
      ##also, this p-value follows the OLS one, not poisson, as its inside different
statements
      summary(lm) ##show this model (OLS)
    }
##But what if the above conditions dont hold? Then:
    else {
      print("No good model: OLS not significant and Dependent Variable is not count v
ariable")
    }
  }
##If BP > 0.05, so heteroskedasticity is present, then:
  else{
    print("Reject null that model is homoskedastic and correct with robust standard e
rrors") #if we reject Breusch Pagan Test, then our model is heteroskedastic and we th
us need a robust standard errors model
    print(paste0("Breusch Pagan P-Value is ", BPpvalue))
    lmrobust <- coeftest(lm, function(x) vcovHC(x, type="HC0"))
    print(lmrobust)
  }
}
best_model(Total_Tweets, SarcasmDummy, Emotional_Tweets_Percentage, IronyDummy, Humou
rDummy) ##Poisson Example
```

```
## [1] "Failed to reject null that model is homoskedastic"
## [1] "Breusch Pagan P-Value is 0.70725"
```

```
##
## Call:
## glm(formula = y ~ ., family = "poisson", data = x)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -360.20  -173.96   -83.53   112.18   752.40
##
## Coefficients:
##                               Estimate Std. Error z value Pr(>|z|)
## (Intercept)                  11.6878723  0.0015870 7364.82   <2e-16 ***
## SarcasmDummy                 -0.0223017  0.0018306  -12.18   <2e-16 ***
## Emotional_Tweets_Percentage -0.0709912  0.0001443 -492.10   <2e-16 ***
## IronyDummy                   -0.1407896  0.0023236  -60.59   <2e-16 ***
## HumourDummy                  -0.0496411  0.0016989  -29.22   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 1819981  on 27  degrees of freedom
## Residual deviance: 1459314  on 23  degrees of freedom
## AIC: 1459662
##
## Number of Fisher Scoring iterations: 6
```

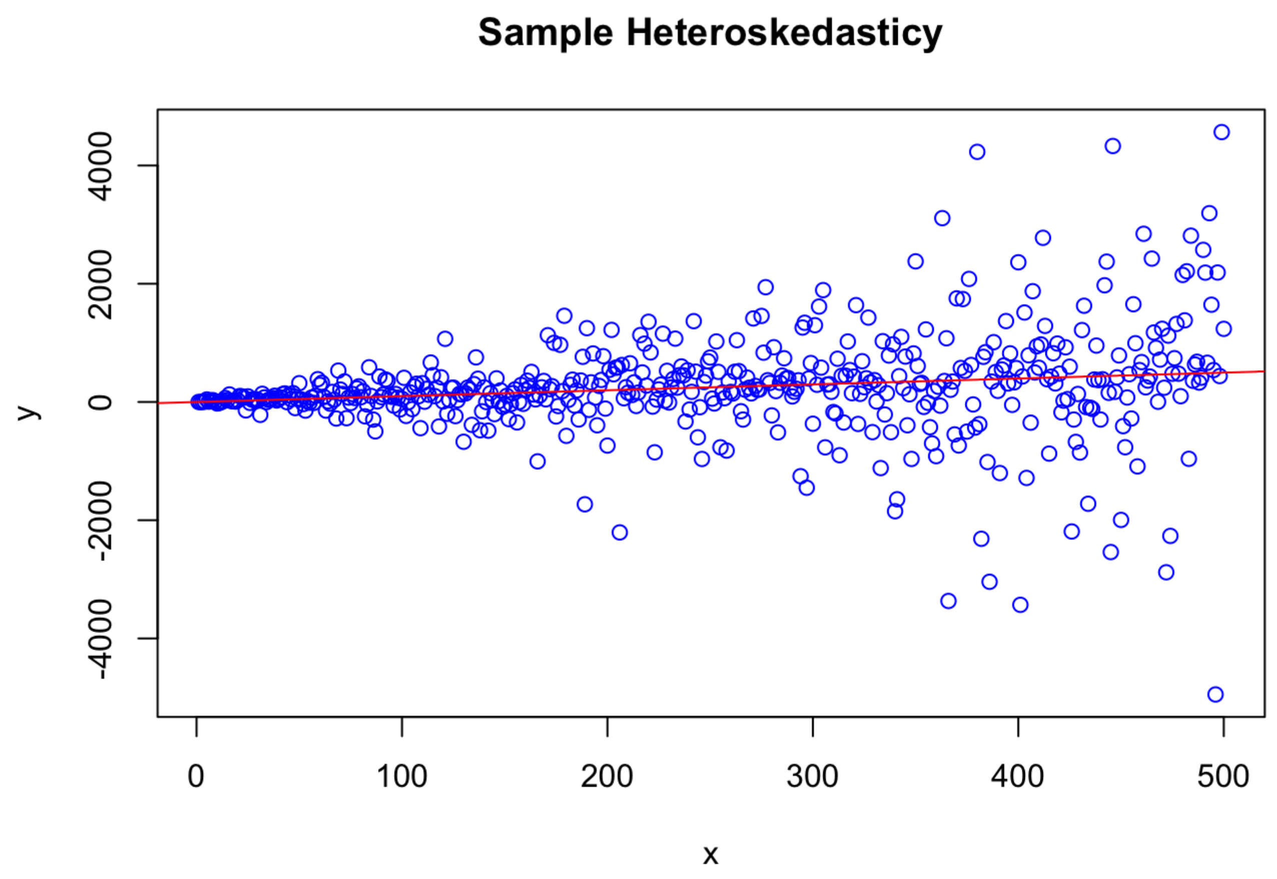
```
best_model(Emotional_Tweets_Percentage, SarcasmDummy, HumourDummy, IronyDummy, Total_
Tweets) ##OLS/Insignificance example
```

```
## [1] "Failed to reject null that model is homoskedastic"
## [1] "Breusch Pagan P-Value is 0.42090"
## [1] "No good model: OLS not significant and Dependent Variable is not count variab
le"
```

As the above comments, I have many exploited if-else functions. I also tried to find a model that was statistically significant and for the continuos variable case as we saw in the Analysis section, this was not the case and thus no model was good enough. I was fortunately able to showcase choosing between OLS and Poisson, but not for heteroskedasticity example.

Now, creating sample variables to showcase the heteroskedastic case:

```
set.seed(123)
n = 500
x <- 1:n
sd<-runif(n, min=0, max=5)
error<-rnorm(n,0,sd*x)
y <- x+error
plot(x,y, main = "Sample Heteroskedasticy", col = "blue")
  abline(0,1, col="red")
```



```
best_model(y, x)
```

```
## [1] "Reject null that model is homoskedastic and correct with robust standard erro
rs"
## [1] "Breusch Pagan P-Value is 0.00000"
##
## t test of coefficients:
##
##               Estimate Std. Error t value  Pr(>|t|)
## (Intercept) -15.75967   52.24094 -0.3017 0.7630275
## x             1.09562    0.31922  3.4322 0.0006485 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As we can see in the above graph, we have a non-constant variance, and our Breusch-Pagan Test shows it also. More importantly, my function has produced the correct result: to use robust standard errors rather than OLS as I am taking account of heteroskedastic errors.

I will be honest, the above function is actually not too bad, but definitely room for advancement. For instance, I have not used for-while loops that could somehow improve it (I'm just not sure how yet!).

However, I could definitely advance this in 'Limited Dependent Variables' in econometrics that will hopefully allow me to compare and contrast econometric models (such as the Logit, Probit, Tobit, Heckamn, Average Treatment Effect amongst others that I could have used, but would probably more apt for another question of interest), but more sophisticated than this one.

6 Conclusion

I have derived quite a lot from this small dataset: particularly from the regression models. Most importantly however, is that I learned how to implement functions very well due to this project (I was pretty egregious at them before!) which is quintessential in the world of computer and data science!