OLS

575 C1 Team 3

2020/10/1

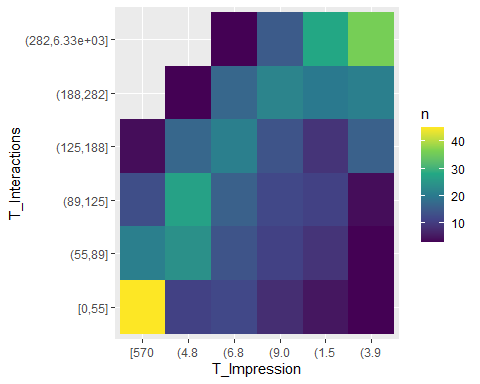
Team Members Dingjie Chen, Siwen He, Hanzi Yu, Jiaqi Yin, Runsheng Wang

In this deliverable, we are performing OLS on the facebook dataset. We first load the packages needed to perform the analysis and read in the delimited file. We modified the column names of the CSV file so that column names would not contain space, as space is not a valid name character in ggplot. We used the complete.cases() function to handle NA values. Also note that “Category” and “Paid” variables are being interpreted as a double by the col\_guess() function. To use these two variables as categorical, we could call the as.factor() function.

#loading packages and dataset  
suppressPackageStartupMessages(library(tidyverse))  
suppressPackageStartupMessages(library(modelr))  
fb <- read\_delim("dataset\_Facebook.csv", delim = ";")  
fb <- fb[complete.cases(fb), ]

To identify potentially meaningful relationships, we construct a heatmap for T\_Impression and T\_Interactions. Since facebook does a horrible job at explaining their metrics, it makes sense for us to first define clearly what each variable is accounting for. After doing some research online, we found that Facebook calculates

# construct heatmap  
fb %>% transmute(T\_Impression = cut\_number(T\_Impression, 6), T\_Interactions = cut\_number(T\_Interactions, 6)) %>% count(T\_Impression, T\_Interactions) %>% ggplot(aes(T\_Impression, T\_Interactions)) + geom\_tile(aes(fill = n)) + scale\_x\_discrete(labels = abbreviate) + scale\_fill\_viridis\_c()



On the heatmap, lighter colors correspond to higher number of posts that belongs to the bin with specific T\_Impression and T\_Interactions. We see that lighter colors are clustering along the diagonal of the heatmap, indicating a potentially positive correlation between the two variables. This heatmap motivate us to plot T\_Interactions against T\_Impression, as illustrated below.

First, we identify outliers and remove them

# clean outliers  
fb.clean <- fb %>% filter(T\_Impression <= 12500, T\_Interactions <= 400)   
# calculate percentage of datapoints considered  
removedx <- fb %>% filter(T\_Impression > 12500) %>% nrow()/nrow(fb)\*100  
removedy <- fb %>% filter(T\_Interactions > 400) %>% nrow()/nrow(fb)\*100  
removedt <- (1 - nrow(fb.clean)/nrow(fb))\*100  
percentage\_tibble <- tribble(~Variable, ~Percentage\_Removed, "T\_Impression", removedx, "T\_Interactions", removedy, "Overall", removedt)  
(percentage\_tibble)

## # A tibble: 3 x 2  
## Variable Percentage\_Removed  
## <chr> <dbl>  
## 1 T\_Impression 38.0  
## 2 T\_Interactions 10.7  
## 3 Overall 38.6

Since social media posts have the potential to be explosive, the disparity in data between posts that performs well and posts that don’t perform as well could be very high. Furthermore, the metrics becomes very unstable and uninformative as Total Impression grows large. Since we only have limited number of datapoints for large values of x and that these values have huge variations, we remove them for OLS. This choice is meaningful because we want to make conclusions for posts performances metrics that are more common rather than less common. The percentage of datapoints removed is displayed in the table above. It might seem that we over-removed data points, but there are simply not enough observations and too much variations for T\_Impression > 12500 and T\_Interactions > 400 for us to make meaningful inferences.

Now, we proceed to calculate OLS

# calculate OLS  
m.ols <- lm(T\_Interactions ~ T\_Impression, data = fb.clean)  
summary(m.ols)

##   
## Call:  
## lm(formula = T\_Interactions ~ T\_Impression, data = fb.clean)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -170.449 -34.262 -2.654 31.406 208.087   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -4.143139 9.269154 -0.447 0.655   
## T\_Impression 0.017804 0.001318 13.512 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 61.91 on 302 degrees of freedom  
## Multiple R-squared: 0.3768, Adjusted R-squared: 0.3747   
## F-statistic: 182.6 on 1 and 302 DF, p-value: < 2.2e-16

round(confint(m.ols,level=0.95),6)

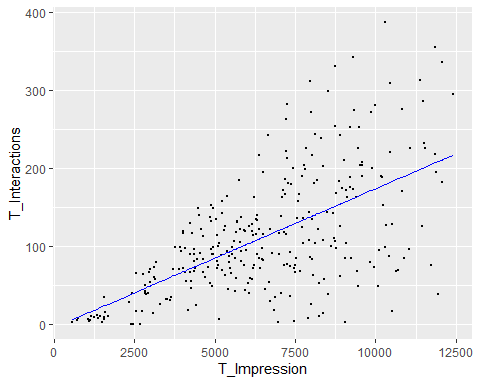
## 2.5 % 97.5 %  
## (Intercept) -22.383445 14.097168  
## T\_Impression 0.015211 0.020396

vcov(m.ols)

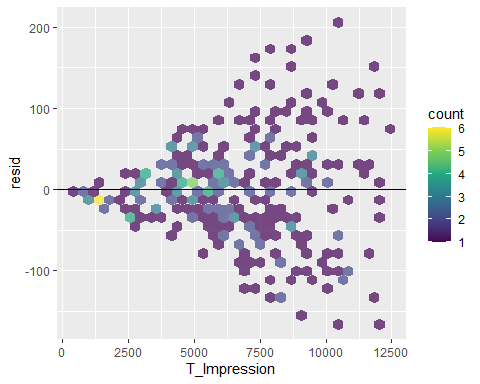
## (Intercept) T\_Impression  
## (Intercept) 85.91721304 -1.128171e-02  
## T\_Impression -0.01128171 1.736152e-06

As can be seen from the output, the fit is not quite good, with an adjusted R-square value of 0.3747. The slope seems to have a high t value and a low p value, indicating that T\_Impression is a significant predictor. However, the relationship under the linear model is not a strong one. The confidence interval for the intercept is very wide, whereas the confidence interval on slope is very narrow. This outcome agrees with our previous analysis that the relationship between T\_Impression and T\_Interactions might not be linear. We will plot the OLS model below with residuals, and offer an alternative solution to fitting this data later in this document.

#plot OLS with predictions  
ggplot(fb.clean %>% add\_predictions(m.ols), aes(T\_Impression, T\_Interactions)) + geom\_point(size = 0.1) + geom\_line(aes(y=pred), color = "blue")



# plot residual  
ggplot(fb.clean %>% add\_residuals(m.ols), aes(T\_Impression, resid))+geom\_hex(alpha = 0.7)+geom\_hline(aes(yintercept = 0))+scale\_fill\_viridis\_c()



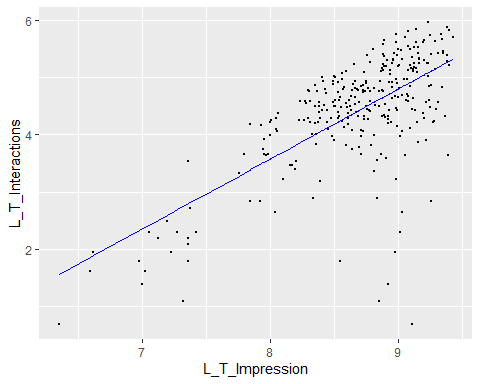
As seen in the graphs above, the linear model doesn’t fit the data quite well. The residual plot also indicate that we have the assumption of equal variances do not hold in this case, as residuals tend to be greater for higher values of T\_Impression. The output of this plot and the visual trend of the scatterplot motivates us to do a log transformation on the variable

# log transform  
logfb <- fb.clean %>% transmute(L\_T\_Interactions = log(T\_Interactions), L\_T\_Impression = log(T\_Impression)) %>% filter(!is.infinite(L\_T\_Interactions))  
m.log <- lm(L\_T\_Interactions ~ L\_T\_Impression, data = logfb)  
summary(m.log)

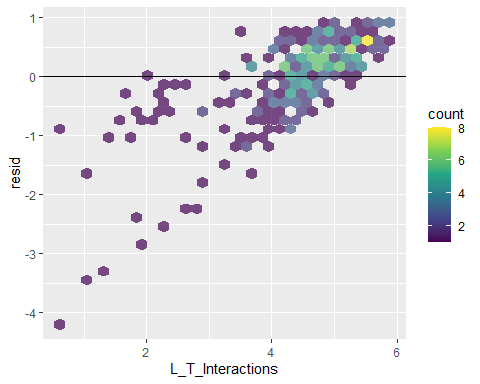
##   
## Call:  
## lm(formula = L\_T\_Interactions ~ L\_T\_Impression, data = logfb)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -4.2396 -0.3061 0.1709 0.4665 0.9808   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -6.16940 0.68040 -9.067 <2e-16 \*\*\*  
## L\_T\_Impression 1.21835 0.07829 15.563 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.7345 on 297 degrees of freedom  
## Multiple R-squared: 0.4492, Adjusted R-squared: 0.4473   
## F-statistic: 242.2 on 1 and 297 DF, p-value: < 2.2e-16

This model is considerably better than the previous one, with very low p values for both parameters of interest. This output indicates that the both beta0 and beta1 are significant. Adjusted R-squared value is also high, at 0.4473, indicating that this fit is better compared to the linear fit. The graphs for the log-transformed variables are given below.

# plot log with predictions  
ggplot(logfb %>% add\_predictions(m.log), aes(L\_T\_Impression, L\_T\_Interactions)) + geom\_point(size = 0.1) + geom\_line(aes(y=pred), color = "blue")



# plot residual  
ggplot(logfb %>% add\_residuals(m.log), aes(L\_T\_Interactions, resid))+geom\_hex(alpha = 0.7)+geom\_hline(aes(yintercept = 0))+scale\_fill\_viridis\_c()



Notice that the log plot follows a roughly linear trend, with residuals roughly clustering around the y=0 line. If log transformation yields a considerably good linear fit, the non-transformed data could exhibit an exponential relationship. We won’t expand on this concept for this project deliverable, but it will be considered when we construct our final project.