Bernauer_Andrew_report2

Andrew Bernauer

February 12, 2019

Report 1

Introduction

For the semester long project I will be attempting to predict the known response variable ethereum price in terms of the predictors: block size, network hash rate growth rate, transactions, day of the week, and month. The project is regression based and not a classification task. Therefore the project will fall under the umbrella of supervised machine learning.

$$ethereumprice = \beta_o + \beta_1 \times blocksize + \beta_2 \times networkhashgrowthrate + \beta_3 \times transactions + \beta_4 \times day + \beta_5 \times month + \epsilon$$

Ethereum is a decentralized, open source, block chain technology, featuring smart contracts. The crypto currency that fuels the Ethereum block chain is Ether. Block size refers to the size of the block chain. Transactions are the number of transactions approved by the ledger. Hash rate is the rate at which the Cryptographic computation takes on the block chain.

Data

```r
library(scales)
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```r
library(purrr)
```

```
##
## Attaching package: 'purrr'

## The following object is masked from 'package:scales':
##
##     discard
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readr)
```

```
##
## Attaching package: 'readr'

## The following object is masked from 'package:scales':
##
##     col_factor
```

```
library(tibble)
library(errorist)
```

```
## Warnings and errors will automatically trigger a web search.
```

```
#substitute in the path on your machine to files
ethereum_transaction_history <- read_csv("C:\\Users\\andre\\Documents\\ECON_490_ML\\ML_report 1\\ethere
```

```
## Parsed with column specification:
## cols(
##   `Date(UTC)` = col_character(),
##   UnixTimeStamp = col_double(),
##   Value = col_double()
## )
```

```
ethereum_block_size <-read_csv("C:\\Users\\andre\\Documents\\ECON_490_ML\\ML_report 1\\ethereum-histori
)
```

```
## Parsed with column specification:
## cols(
##   `Date(UTC)` = col_character(),
##   UnixTimeStamp = col_double(),
##   Value = col_double()
## )
```

```r
ethereum_network_hash_rate <- read_csv("C:\\Users\\andre\\Documents\\ECON_490_ML\\ML_report 1\\ethereum-
```

```
## Parsed with column specification:
## cols(
##   `Date(UTC)` = col_character(),
##   UnixTimeStamp = col_double(),
##   Value = col_double()
## )
```

```r
ether_price <- read_csv("C:\\Users\\andre\\Documents\\ECON_490_ML\\ML_report 1\\ethereum-historical-dat
```

```
## Parsed with column specification:
## cols(
##   `Date(UTC)` = col_character(),
##   UnixTimeStamp = col_double(),
##   Value = col_double()
## )
```

```r
# creating variable for day
day_of_week <-
  lubridate::mdy(ether_price$`Date(UTC)`) %>%
  wday( ,label=TRUE) %>%
  sort()

# creating variable for month
month_of_year <-
  lubridate::mdy(ether_price$`Date(UTC)`) %>%
  month( ,label=TRUE) %>%
  sort()

# construct list of variables to coerce into a tibble
l_ether <- list( price = ether_price$Value,
                 transaction_history = as.integer(ethereum_transaction_history$Value),
                 block_size = as.integer(ethereum_block_size$Value),
                 hash_rate = ethereum_network_hash_rate$Value,
                 day = day_of_week,
                 month = month_of_year,
                 date_utc = mdy(ethereum_network_hash_rate$`Date(UTC)`),
                 unix_time_stamp = ethereum_network_hash_rate$UnixTimeStamp)

ether_df <- as_tibble(l_ether)

head(ether_df)
```

```
## # A tibble: 6 x 8
##   price transaction_his~ block_size hash_rate day   month date_utc
##   <dbl>            <int>      <int>     <dbl> <ord> <ord> <date>
## 1     0             8893        644      11.5 Sun   Jan   2015-07-30
## 2     0                0        582      51.5 Sun   Jan   2015-07-31
## 3     0                0        575      57.8 Sun   Jan   2015-08-01
## 4     0                0        581      67.9 Sun   Jan   2015-08-02
## 5     0                0        587      74.6 Sun   Jan   2015-08-03
```

```
## 6     0                 0        587      82.0 Sun   Jan   2015-08-04
## # ... with 1 more variable: unix_time_stamp <dbl>
```

```r
tail(ether_df)
```

```
## # A tibble: 6 x 8
##   price transaction_his~ block_size hash_rate day   month date_utc
##   <dbl>            <int>      <int>     <dbl> <ord> <ord> <date>
## 1  233.           479351      23846   263794. Sat   Dec   2018-09-30
## 2  231.           476308      24593   259027. Sat   Dec   2018-10-01
## 3  225.           490262      24107   259556. Sat   Dec   2018-10-02
## 4  220.           559006      21423   258087. Sat   Dec   2018-10-03
## 5  222.           559181      22655   261318. Sat   Dec   2018-10-04
## 6  228.           595361      20849   259922. Sat   Dec   2018-10-05
## # ... with 1 more variable: unix_time_stamp <dbl>
```

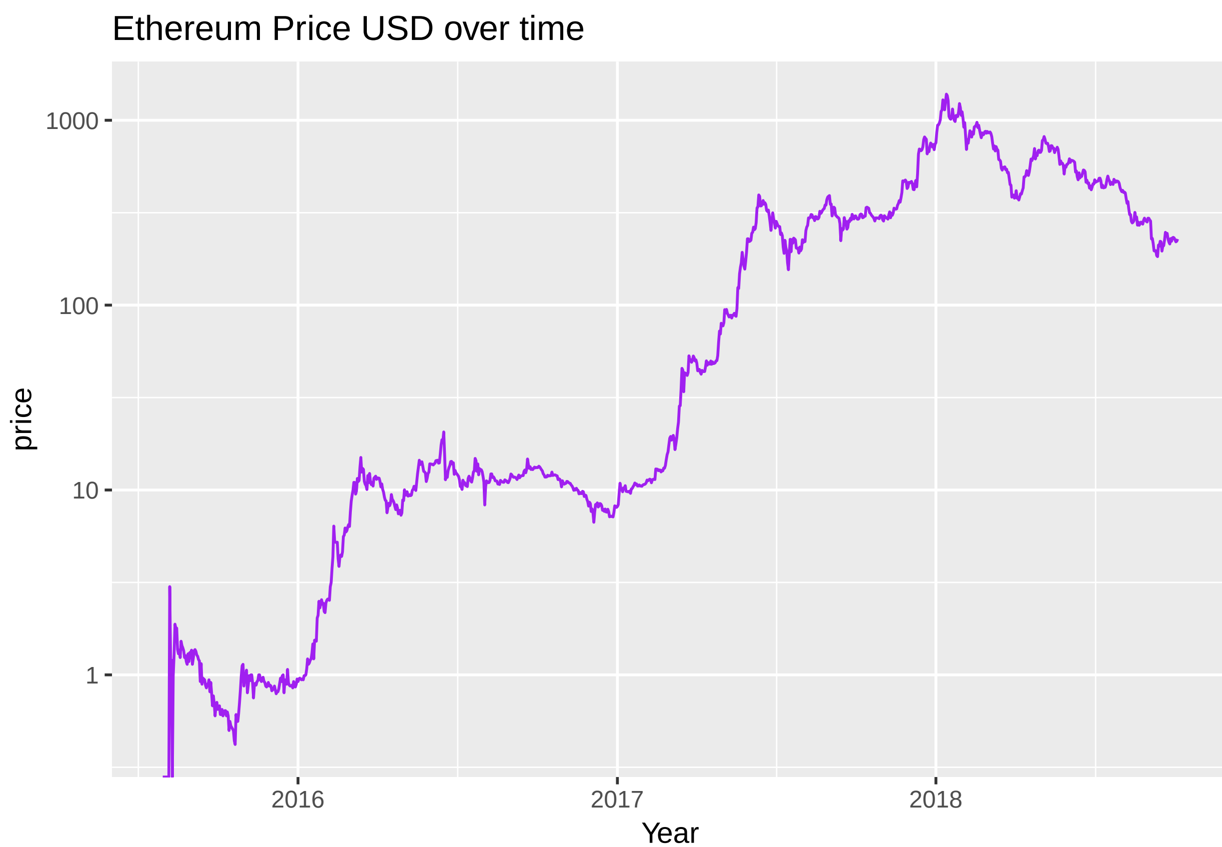
```r
summary(ether_df)
```

```
##      price          transaction_history   block_size
##  Min.   :   0.000   Min.   :      0     Min.   :  575
##  1st Qu.:   9.553   1st Qu.:  37166     1st Qu.: 1424
##  Median :  18.655   Median :  63518     Median : 2209
##  Mean   : 211.731   Mean   : 275452     Mean   : 9292
##  3rd Qu.: 333.202   3rd Qu.: 539046     3rd Qu.:19596
##  Max.   :1385.020   Max.   :1349890     Max.   :33681
##
##    hash_rate           day          month        date_utc
##  Min.   :     11.53   Sun:166   Aug    :124   Min.   :2015-07-30
##  1st Qu.:   2786.45   Mon:166   Sep    :120   1st Qu.:2016-05-15
##  Median :  11610.26   Tue:166   Oct    : 98   Median :2017-03-02
##  Mean   :  82768.27   Wed:166   Jul    : 95   Mean   :2017-03-02
##  3rd Qu.:143765.47   Thu:167   Jan    : 93   3rd Qu.:2017-12-18
##  Max.   :295912.00   Fri:167   Mar    : 93   Max.   :2018-10-05
##                      Sat:166   (Other):541
##  unix_time_stamp
##  Min.   :1.438e+09
##  1st Qu.:1.463e+09
##  Median :1.488e+09
##  Mean   :1.488e+09
##  3rd Qu.:1.514e+09
##  Max.   :1.539e+09
##
```

Plots

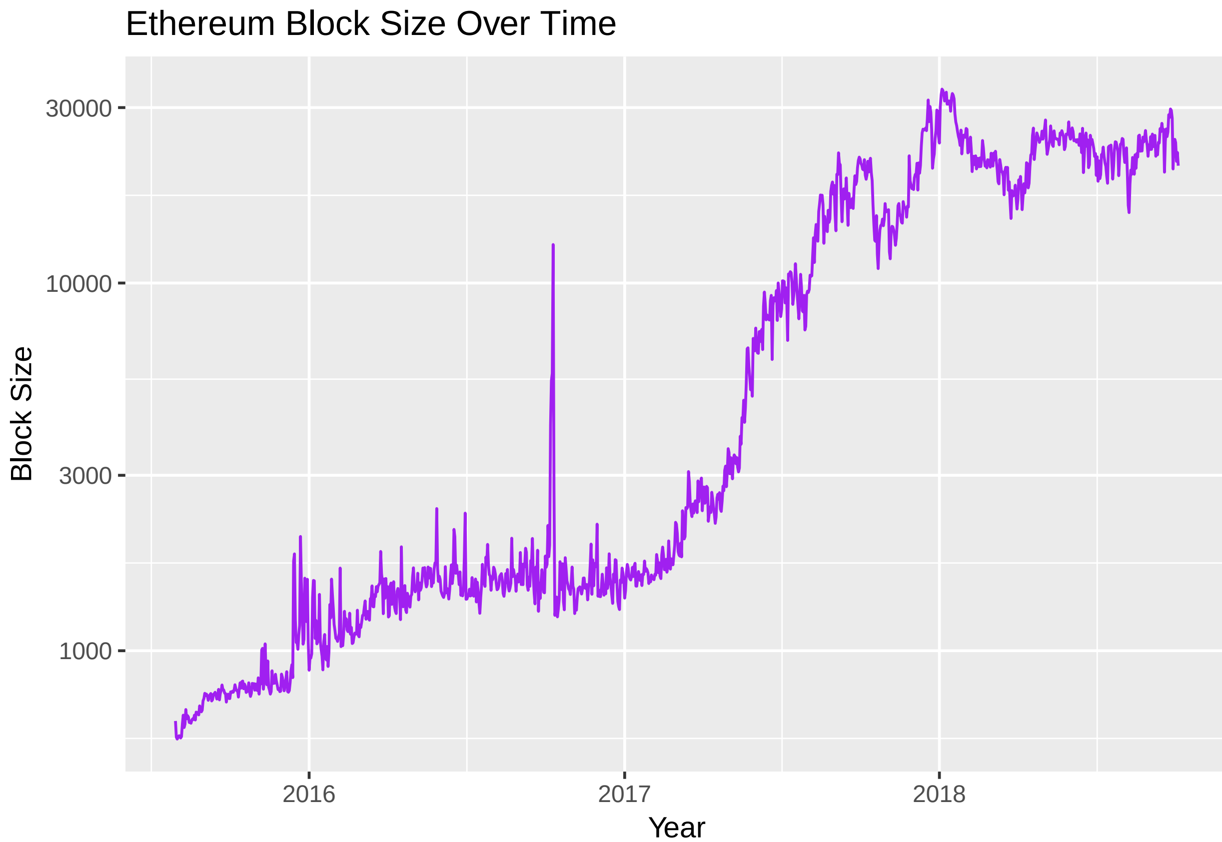
```r
price_history <- ggplot(ether_df) +
  aes(date_utc, price) +
  geom_line(color="purple") +
  xlab("Year")+
  ggtitle("Ethereum Price USD over time") +
  scale_y_log10()
```

```
price_history
```



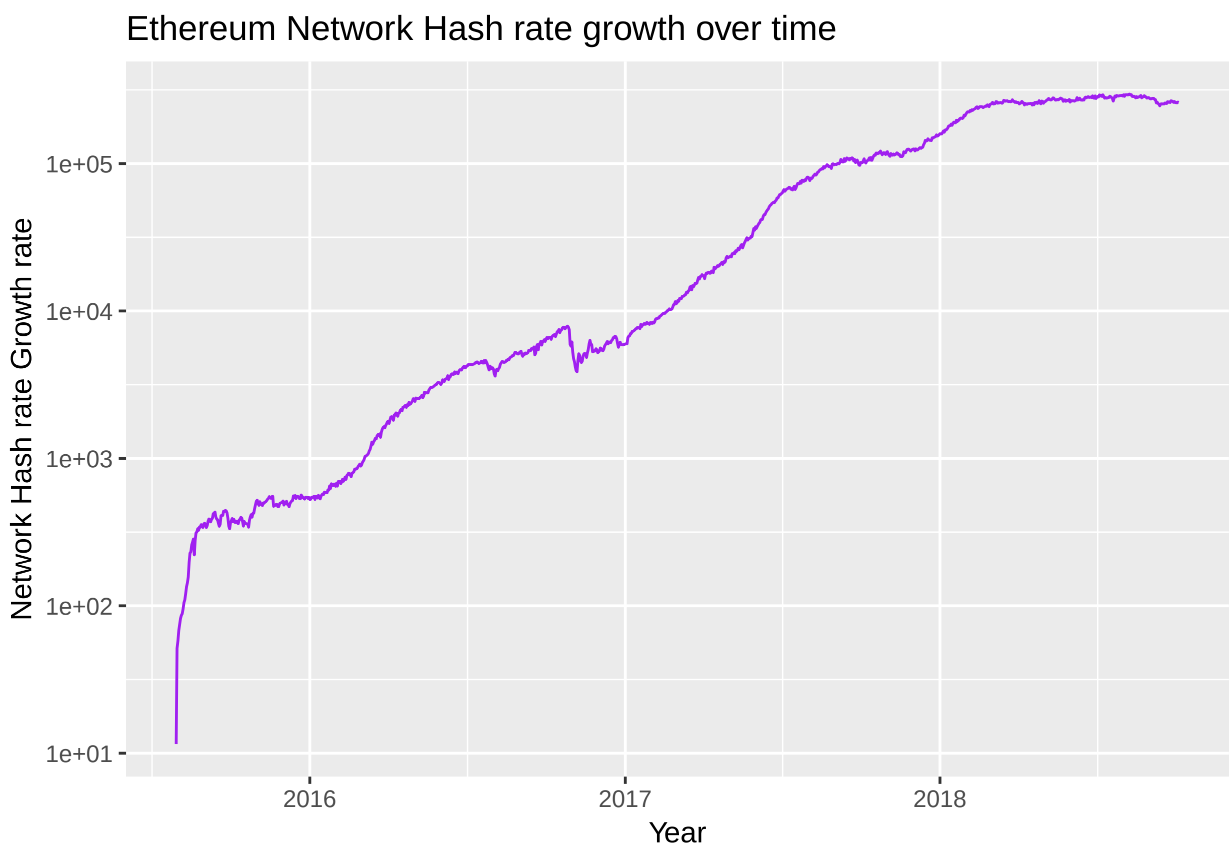
The price doesn't seem move at all significantly until 2016 so I decided to use a log transformation to the y axis. This removed a large amount the visual noise and made the price change overtime more evident. Besides that ethereum price starts to rise from around a dollar price to hovering around the ten dollars between 2016 and 2017. Peaks at a value of 1000 dollars in 2018.

```
block_history <- ggplot(ether_df) +
  aes(date_utc, block_size) +
  geom_line(color ="purple") +
  scale_y_log10() +
  xlab("Year") +
  ylab("Block Size") +
  ggtitle("Ethereum Block Size Over Time")


block_history
```



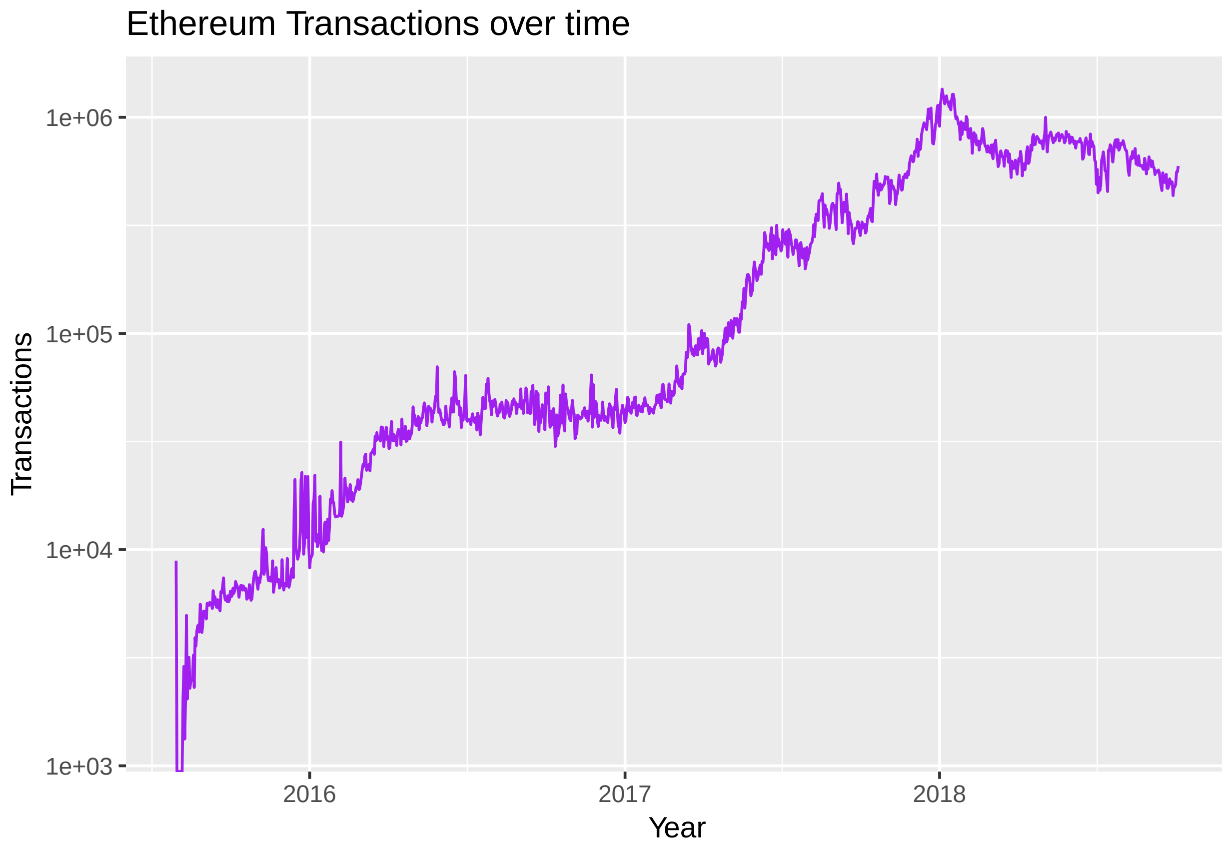
Block size rises exponentially begining in 2017. Prior to this it fluctuates up and down between 1000 and 3000 which might be pointing to autocorrelation. In this same time period it peaks violently at 10000. Reaching a maximum of over 30000 in 2018. Another log tranformation was used in this plot.

```
hashrate_history <- ggplot(ether_df) +
 aes(date_utc, hash_rate) +
 geom_line(color="purple") +
 scale_y_log10() +
 xlab("Year") +
 ylab("Network Hash rate Growth rate") +
 ggtitle("Ethereum Network Hash rate growth over time")

hashrate_history
```



Network hashrate growth blows up exponentially and seems to follow that trend overtime.

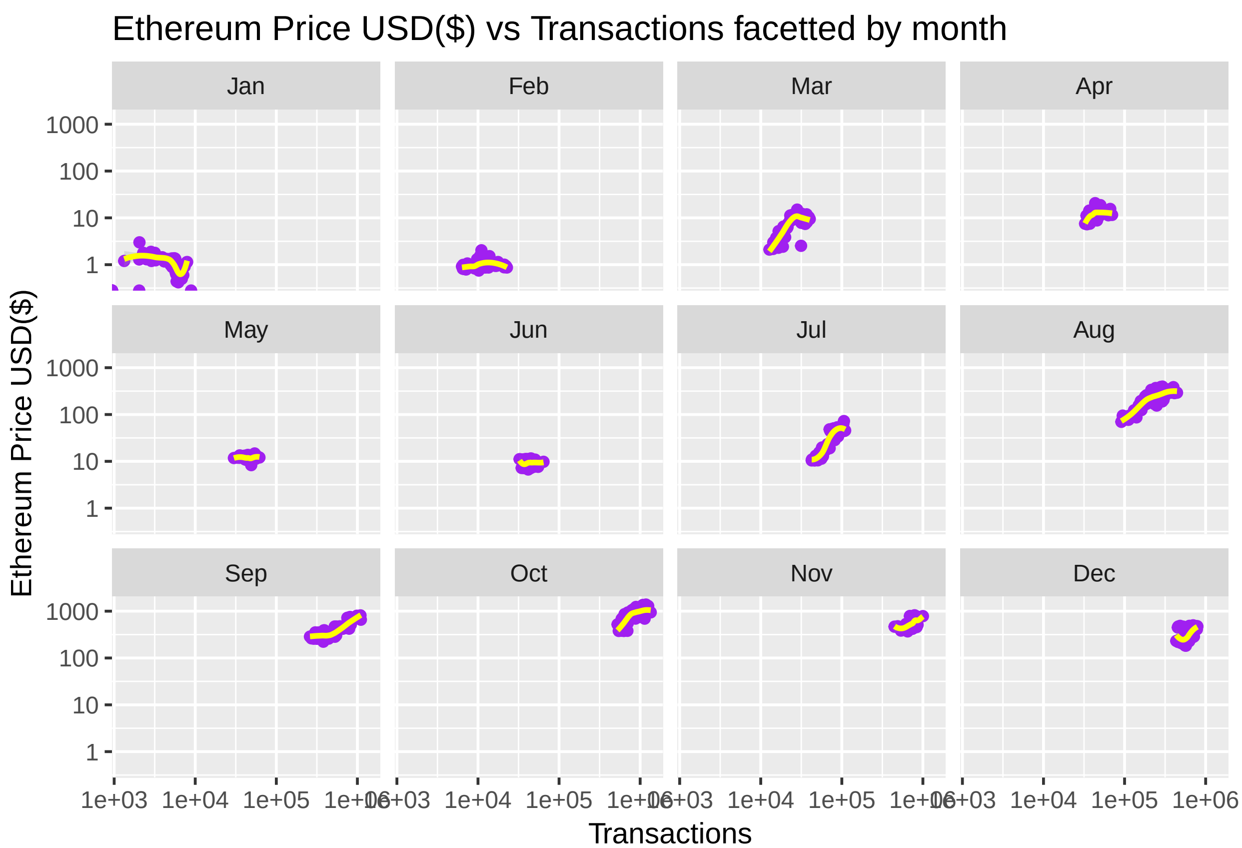
```
transaction_history <- ggplot(ether_df) +
 aes(date_utc, transaction_history) +
 geom_line(color="purple") +
 xlab("Year") +
 ylab("Transactions") +
 scale_y_log10() +
 ggtitle("Ethereum Transactions over time")

transaction_history
```



Transactions on the Ethereum network are growing very quickly and people are embracing the technology it is based on.

```
price_plot <- ggplot(ether_df) +
  aes(transaction_history, price) +
  geom_point(colour="purple") +
  facet_wrap(~month) +
  geom_smooth(colour="yellow", alpha = 0.25)

price_plot +
  scale_x_log10("Transactions") +
  scale_y_log10("Ethereum Price USD($)") +
  ggtitle("Ethereum Price USD($) vs Transactions facetted by month")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Ether Price appears to fall into clusters as transactions increase and you isolate for month of the year.

Summary Code

```
#summary stats for ethereum price
ether_df %>% summarise(mean_price = mean(price, na.rm = TRUE), median_price = median(price), sd_price =
```

```
## # A tibble: 1 x 8
##   mean_price median_price sd_price iqr_price     n mad_price min_price
##        <dbl>        <dbl>    <dbl>     <dbl> <int>     <dbl>     <dbl>
## 1       212.         18.7     282.      324.  1164      26.9         0
## # ... with 1 more variable: max_price <dbl>
```

```
#summary stats for ethereum transaction history
ether_df %>% summarise(mean_transaction_history = mean(transaction_history), median_transaction_history
```

```
## # A tibble: 1 x 8
##   mean_transactio~ median_transact~ sd_transaction_~ iqr_transaction~     n
##              <dbl>            <dbl>            <dbl>            <dbl> <int>
## 1          275451.           63518.          318759.          501880.  1164
## # ... with 3 more variables: mad_transaction_history <dbl>,
## #   min_transaction_history <dbl>, max_transaction_history <dbl>
```

```r
#summary stats for ethereum block size
ether_df %>% summarise(mean_block_size = mean(block_size), median_block_size = median(block_size), sd_bl
```

```
## # A tibble: 1 x 8
##   mean_block_size median_block_si~ sd_block_size iqr_block_size     n
##             <dbl>            <dbl>         <dbl>          <dbl> <int>
## 1           9292.             2209         9856.          18172  1164
## # ... with 3 more variables: mad_block_size <dbl>, min_block_size <dbl>,
## #   max_block_size <dbl>
```

```r
#summary stats for hash rate
ether_df %>% summarise(mean_hash_rate = mean(hash_rate), median_hash_rate = median(hash_rate), sd_hash_
```

```
## # A tibble: 1 x 8
##   mean_hash_rate median_hash_rate sd_hash_rate iqr_hash_rate     n
##            <dbl>            <dbl>        <dbl>         <dbl> <int>
## 1         82768.           11610.      105868.       140979.  1164
## # ... with 3 more variables: mad_hash_rate <dbl>, min_hash_rate <dbl>,
## #   max_hash_rate <dbl>
```

Conclusion

Ethereum data is quite complex on the surface level before the next report I should complete some more research on it's block chain technology. As I am more familiar with Bitcoin and it could lead to new insights on the data.

Report 2: Regression Modeling Ether Price

Regressions

```r
#running five regressions

poly_reg <- lm(price ~ poly(transaction_history, 5) + poly(block_size, 5) + poly(hash_rate, 5), ether_df


reg_obj <- lm(log(price) ~ log(block_size) + log(transaction_history) + log(hash_rate), data = ether_df

reg_obj_2 <- lm(log(price) ~ log(transaction_history) + log(hash_rate), data = ether_df, subset = price


reg_obj_3 <- lm(price ~ I(block_size)^2 + sqrt(transaction_history) + hash_rate, data = ether_df)

mols <- lm(price ~ block_size + transaction_history + hash_rate, data = ether_df)

log_ols <- lm(log(price) ~ log(transaction_history), data = ether_df, subset = price > 0)
```

Regression Diagnostics

The following code chunk returns tidyed summary for the best regression I ran.

```
library(broom)


tidy_reg_obj <- reg_obj %>%
  tidy()


tidy_reg_obj
```

```
## # A tibble: 4 x 5
##   term                      estimate std.error statistic   p.value
##   <chr>                        <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)               -10.6       0.211     -50.4  8.64e-294
## 2 log(block_size)             0.0720    0.0387      1.86 6.32e-  2
## 3 log(transaction_history)    1.01      0.0491     20.5  5.66e- 80
## 4 log(hash_rate)              0.222     0.0311      7.15 1.56e- 12
```

```
glance(reg_obj)
```

```
## # A tibble: 1 x 11
##   r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>  <dbl> <dbl> <dbl>
## 1     0.964         0.964 0.438    10269.       0     4  -684. 1378. 1404.
## # ... with 2 more variables: deviance <dbl>, df.residual <int>
```

All of the coefficients return p values significant at the .10 cut off level while log block size is close. The F statistic is significant; however the BIC and AIC below could be lower.

```
glance_tidy_poly <- glance(poly_reg)

glance_reg_obj <- glance(reg_obj)

glance_reg_obj_2 <- glance(reg_obj_2)

glance_reg_obj_3 <- glance(reg_obj_3)

glance_mols <- glance(mols)

glance_log_ols <- glance(log_ols)


glance_tidy_poly$adj.r.squared
```

```
## [1] 0.9396409
```

```
glance_reg_obj$adj.r.squared
```

```
## [1] 0.9638884
```

```
glance_reg_obj_2$adj.r.squared
```

```
## [1] 0.9638114
```

```
glance_reg_obj_3$adj.r.squared
```

```
## [1] 0.8428993
```

```
glance_mols$adj.r.squared
```

```
## [1] 0.9012137
```

```
glance_log_ols$adj.r.squared
```

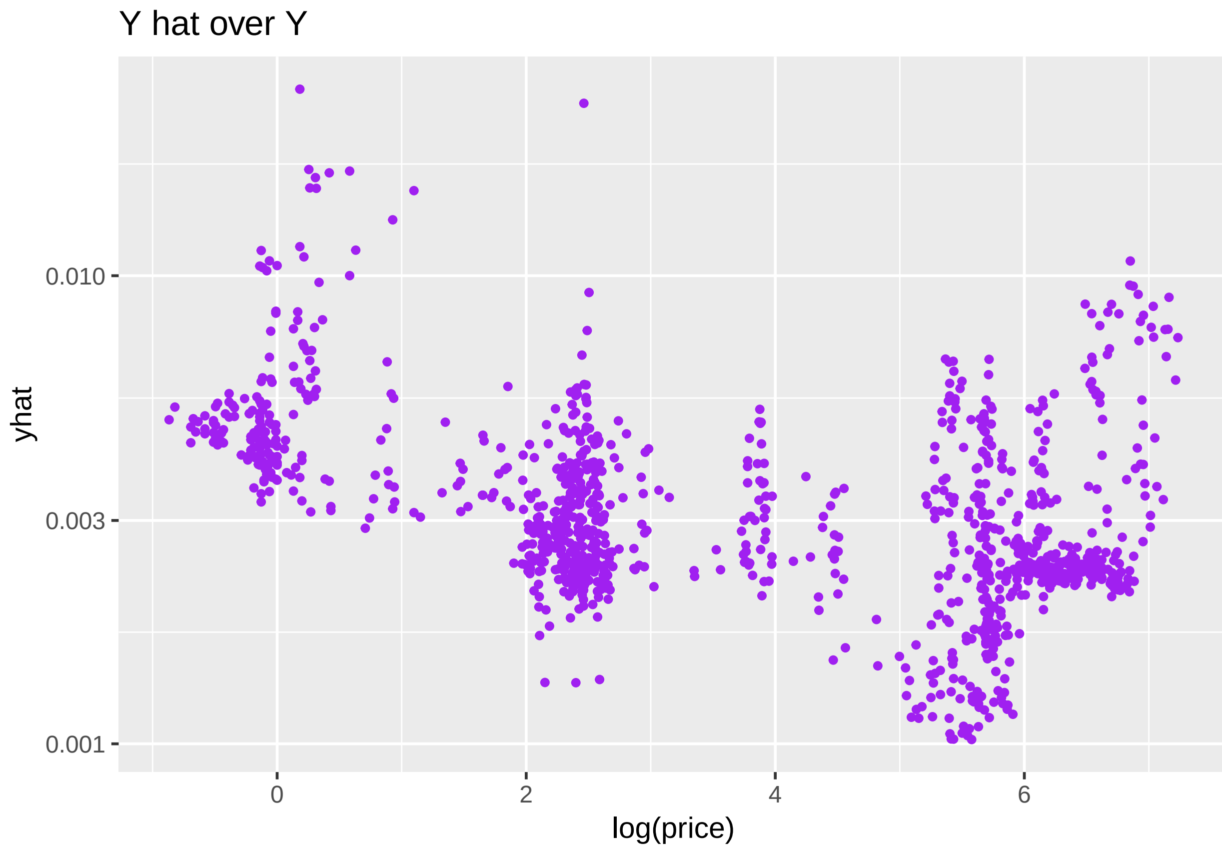
```
## [1] 0.9620687
```

The log-log model with all the continuous predictors included had the highest adjusted $R^2 $ of all the regressions run. A model excluding the log block size term comes in a close second. Another, potential option is the last model with just log transaction history as a predictor.

Predictors performing the best presented the following equation $\hat{y} = -10.65 + 0.072\log(BlockSize) + 1.0007\log(TransactionHistory) + 0.222\log(HashRate)$.

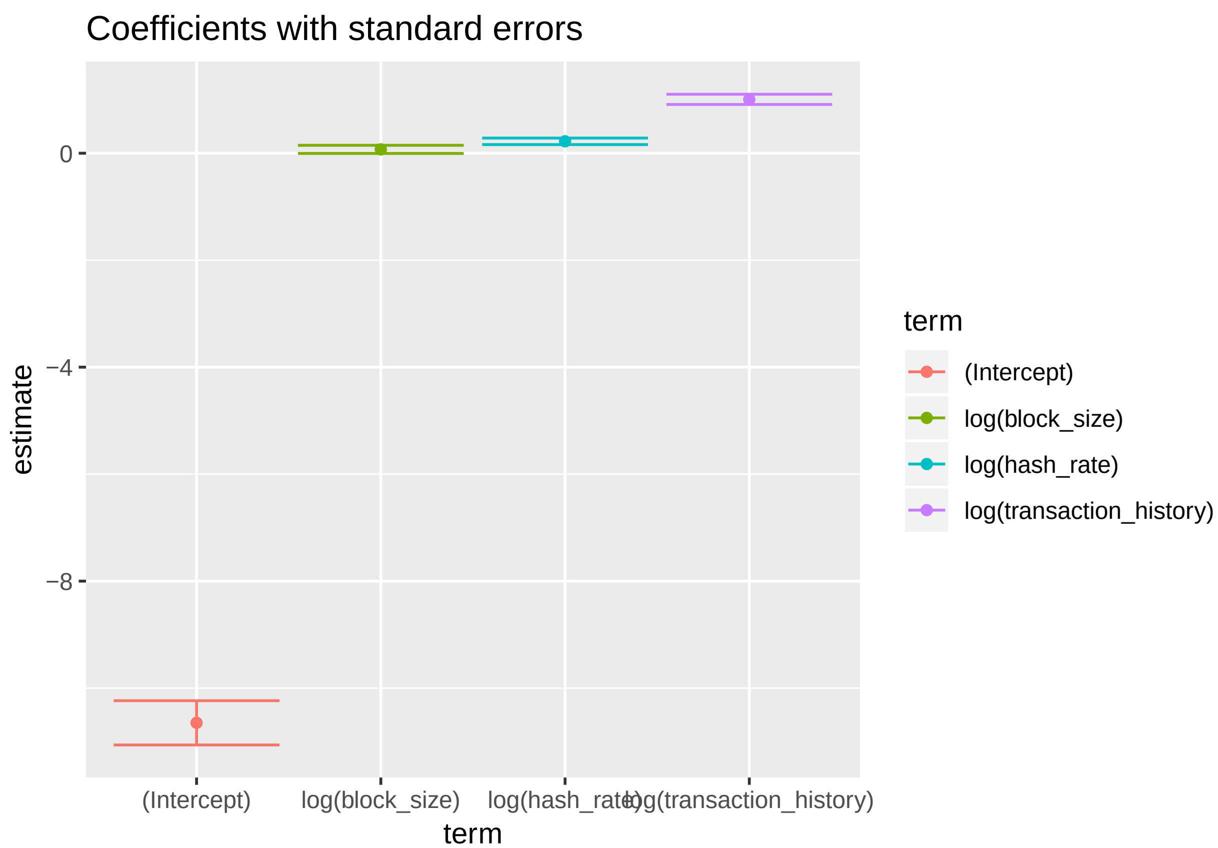
Which can be interpreted as a one percent increase in Blocksize resulting in a 0.072 percent increase in Ether price. The same interpretation applies to the other predictors with there respective coefficients.

```
augmented_reg_obj <- reg_obj %>%
             augment()

some_plot <- ggplot(data = augmented_reg_obj, aes(log.price., .hat)) +
  geom_point(color = "purple", size = 1) +
  labs(x="log(price)", y="yhat") +
  scale_y_log10() +
  ggtitle("Y hat over Y")

some_plot
```



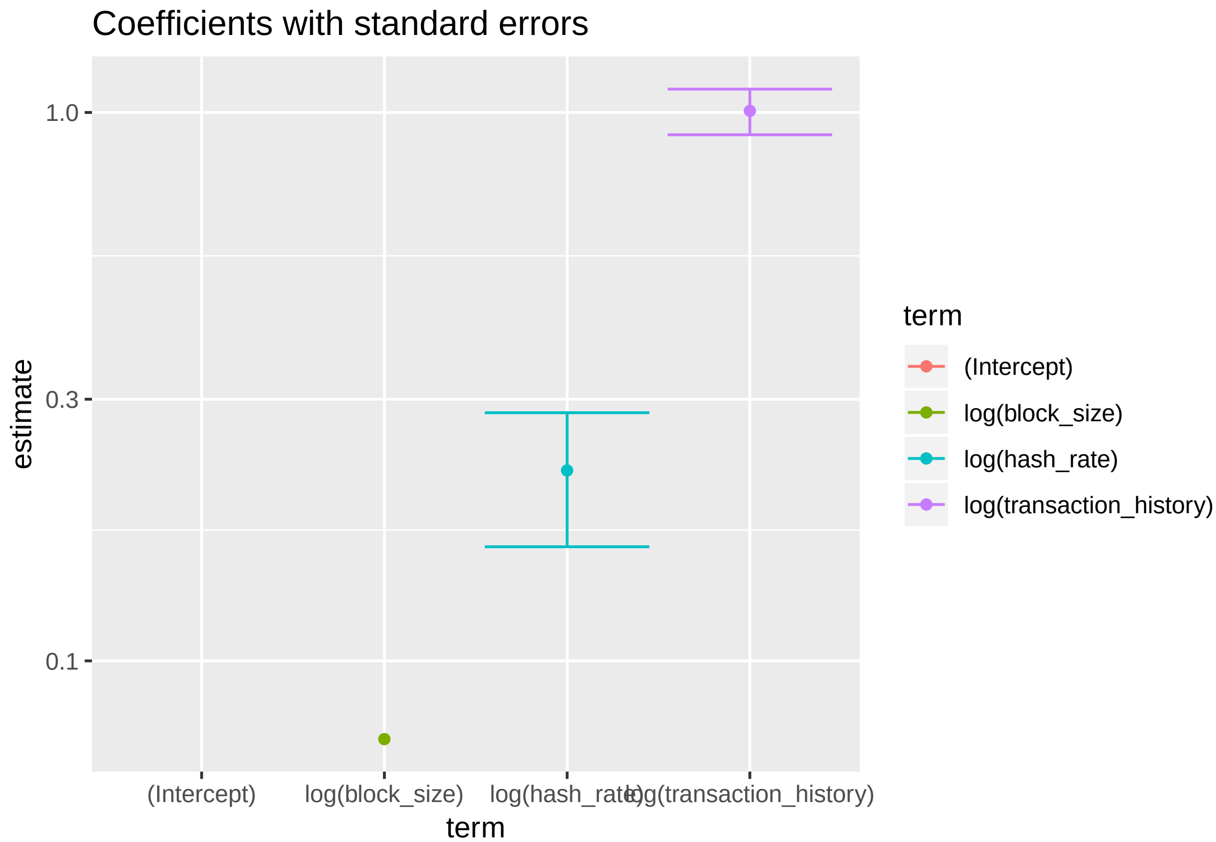
The preceding plot points to the non-linear nature of the data set. A regression model with a polynomial term could lead to future improvements.

```
tidy_conf_it <- tidy(reg_obj, conf.int = TRUE)

confin_it_plot <- ggplot(tidy_conf_it, aes(term, estimate, color=term)) +
  geom_point() +
  geom_errorbar(aes(ymin=estimate - 1.96*std.error, ymax=estimate + 1.96*std.error)) +
  ggtitle("Coefficients with standard errors")

  confin_it_plot
```



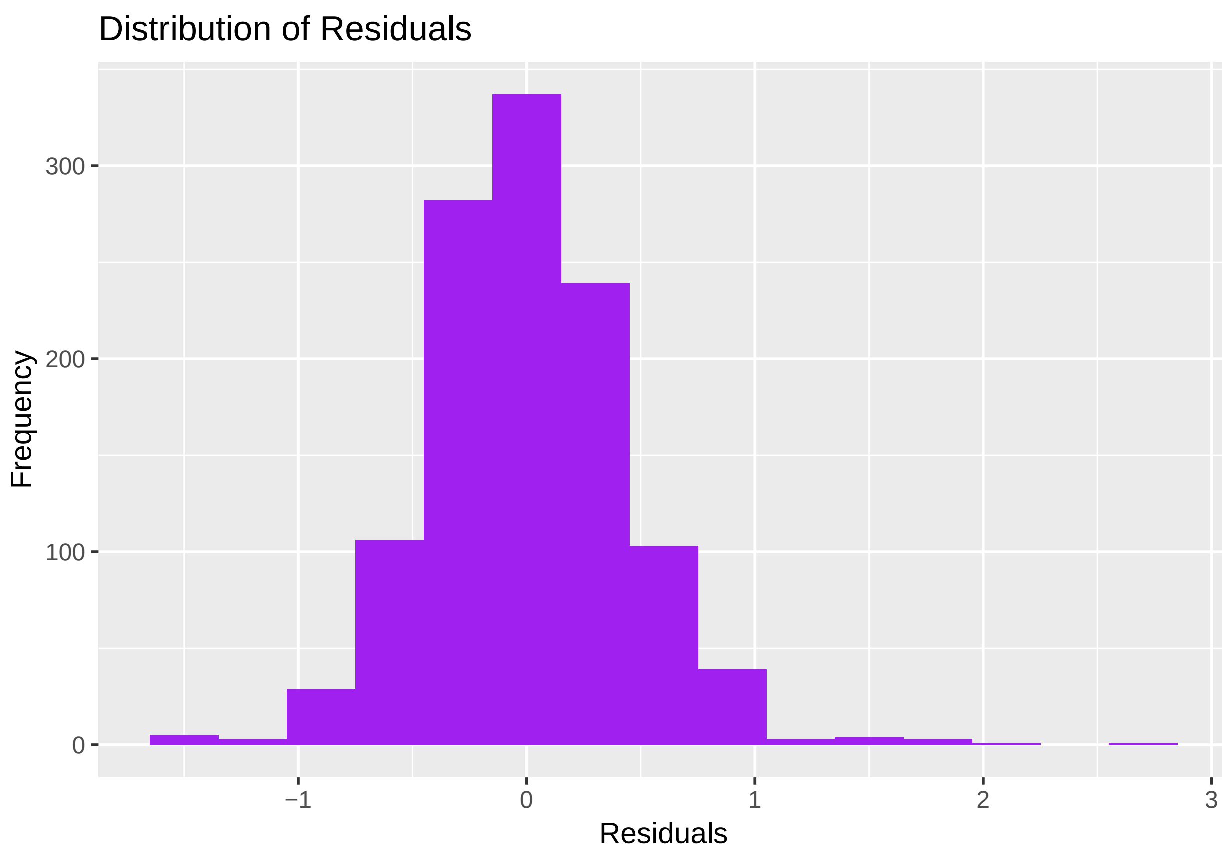
The previous graph is misleading due to the scale with a scale adjustment it is more insightful.

```
confin_it_plot +
  scale_y_log10()
```



This plot points to a potential issue with logblocksize term not being statistically significant.

```
some_plot_2 <- ggplot(data = augmented_reg_obj, aes(.resid)) +
  geom_histogram(binwidth = .3, fill = "purple") +
  labs(x="Residuals", y="Frequency") +
  ggtitle("Distribution of Residuals")



some_plot_2
```



The majority of the residuals are distributed between -1 and 1 peaking at zero.

```
some_plot_3 <- ggplot(augmented_reg_obj, aes(log.price., .resid)) +
  geom_point(color = "purple", size = 1) +
  geom_hline(color = "yellow", yintercept = 0) +
  labs(x="Y", y="Residuals") +
  ggtitle("Residuals over Ether Price")

some_plot_3
```

The Residuals plotted against Ether Price suggest there is possible non-normality in the distribution of the errors.

KNN Regression

```
library(caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift
```

```
set.seed(20)

train_index <- createDataPartition(ether_df$price, p = .8, list = FALSE, times = 1)

head(train_index)
```

```
##      Resample1
## [1,]         1
## [2,]         2
## [3,]         3
## [4,]         4
## [5,]         5
## [6,]         6
```

```r
ether_df_train <- ether_df[train_index, ]

ether_df_test <- ether_df[-train_index, ]

tr_control <- trainControl(method = "repeatedcv",
             number = 10,
             repeats = 3
             )
knn_model <-  train(price ~., data = ether_df,
                   method = "knn",
                   preProcess = c("center", "scale"),
                   trControl = tr_control,
                   metric = "RMSE",
                   tuneLength = 10
                   )

knn_model
```

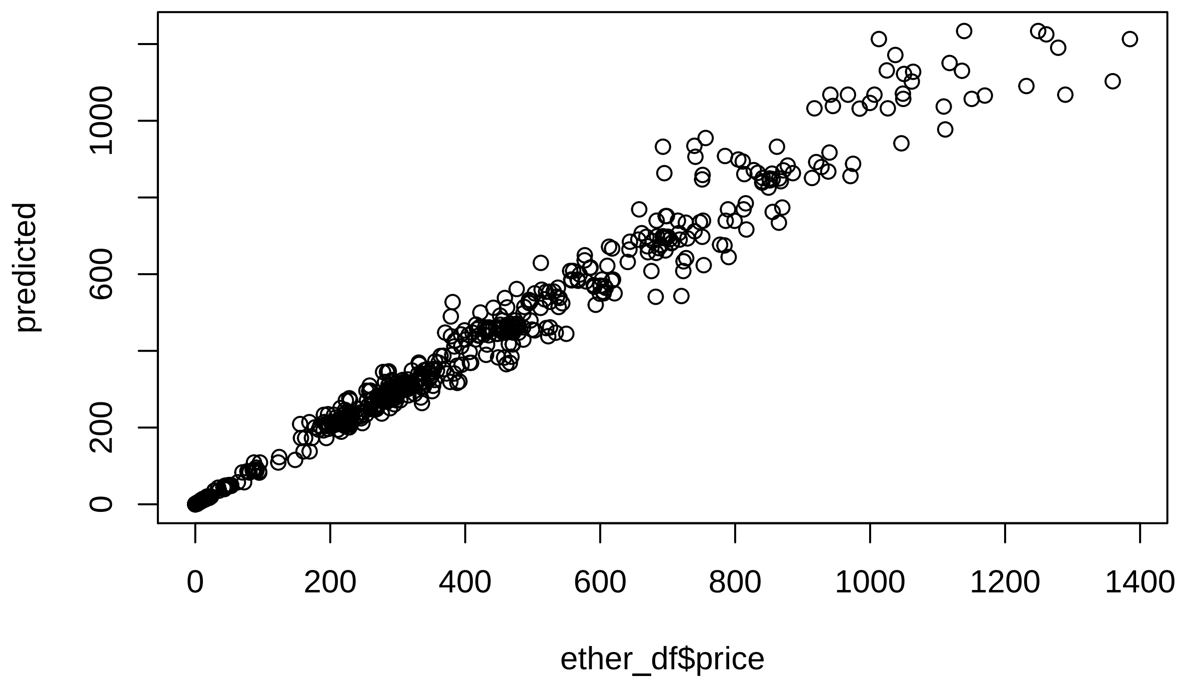
```
## k-Nearest Neighbors
##
## 1164 samples
##    7 predictor
##
## Pre-processing: centered (22), scaled (22)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 1047, 1048, 1048, 1047, 1046, 1048, ...
## Resampling results across tuning parameters:
##
##   k   RMSE      Rsquared   MAE
##    5  43.65234  0.9773973  19.24461
##    7  46.00590  0.9749599  20.23203
##    9  48.04977  0.9726511  21.03995
##   11  48.69456  0.9719216  21.57974
##   13  49.18474  0.9713990  22.01471
##   15  49.59965  0.9710149  22.39543
##   17  50.48596  0.9701045  23.14410
##   19  51.56428  0.9687621  24.00458
##   21  52.53695  0.9674595  24.75981
##   23  53.60075  0.9660714  25.57364
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 5.
```

```r
knn_model$results
```

```
##     k     RMSE  Rsquared      MAE   RMSESD  RsquaredSD    MAESD
```

```
## 1   5 43.65234 0.9773973 19.24461 9.459690 0.008050428 3.594521
## 2   7 46.00590 0.9749599 20.23203 9.269257 0.008140425 3.344544
## 3   9 48.04977 0.9726511 21.03995 9.844359 0.009036849 3.558092
## 4  11 48.69456 0.9719216 21.57974 9.467905 0.008738295 3.482206
## 5  13 49.18474 0.9713990 22.01471 9.215891 0.008266275 3.405115
## 6  15 49.59965 0.9710149 22.39543 9.032538 0.008051142 3.406538
## 7  17 50.48596 0.9701045 23.14410 9.013350 0.008171535 3.482041
## 8  19 51.56428 0.9687621 24.00458 8.869989 0.008001962 3.504933
## 9  21 52.53695 0.9674595 24.75981 8.824245 0.008295191 3.596823
## 10 23 53.60075 0.9660714 25.57364 8.831999 0.008684751 3.687776
```

```r
y_hat <- predict(knn_model, newdata = ether_df)

predicted <- predict(knn_model, ether_df)

plot(ether_df$price, predicted)
```




```r
sqrt(sum((predicted - ether_df$price) ^ 2) / length(ether_df))
```

```
## [1] 401.4716
```

Conclusion

Before my next report adding variables that are scaled with logs to my tibble would make future plotting easier. KNN does not appear to be a significant improvement in comparison to the standard OLS techniques. Regression models with a mix of different coefficient or transformation techniques could yield better results. Though this might complicate the interpretation of the model.