Analysis of Cryptocurrency Price Changes

Math23C Spring 2018 Final Project - Project Highlights

Patrick Watts, David Wihl

May 6, 2018

Abstract

Cryptocurrencies such as Bitcoin, Ethereum, etc. generated significant attention in 2017 due a dramatic price increases followed by a spectacular drop in 2018. Cryptocurrencies have significant volatility as there is rampant speculation combined with a lack of tangible resource to provide a proxy for real-world value. Given the high variance in prices, can data science methods explored in this class be used to model the market dynamics?

Contents

Introduction

We obtained historical price information for three major and one minor cryptocurrencies in order to analyze market dynamics. We also obtained historical pricing of eighteen traditional securities such as the S&P 500, Gold, LIBOR rates, and VIX index to see if we obtain correlations or predictive factors for the changes in the cryptocurrency prices.

Data Sources

Cryptocurrency Historical Pricing

For the cryptocurrency prices, we downloaded the data from Kaggle, specifically https://www.kaggle.com/sudalairajkumar/cryptocurrencypricehistory/data. The original provenance of the data is coinmarketcap, a well-reputed aggregator of cryptocurrency pricing on various exchanges around the world. Cryptocurrencies have several distinguishing pricing factors: there are a myriad of markets around the world leading to potential arbitrage opportunities and markets are open 24 hours per day. Thus, traditional Open, High, Low, Close (OHLC) metrics have to be artificially imposed on the data.

We selected four different cryptocurrencies Bitcoin (BTC), Ethereum (ETH), Monero (XMR), Ripple (XRP). Bitcoin and Ethereum represent the majority of overall cryptocurrency market capitalization. We chose Monero and Ripple to see if market dynamics affected all cryptocurrencies equally.

The following source CSV files are in the `data/` subdirectory:

- `bitcoin_price.csv` - Bitcoin (BTC) from April 2013 until Feb 2018 (1761 rows)
- `ethereum_price.csv` - Ethereum (ETH) from Aug 2015 until Feb 2018 (930 rows)
- `monero_price.csv` - Monero (XMR) from May 2014 until Feb 2018 (1372 rows)
- `ripple_price.csv` - Ripple (XRP) from Aug 2013 until Feb 2018 (1663 rows)

(REQ: a dataframe, at least two numeric columns, at least 20 rows)

Traditional, Noncryptocurrency Data

For comparison against traditional markets, we used a Python notebook to download many other historical financial metrics. The script can be found in `DownloadOtherMetrics.ipynb`. It is intended to be run once. The script collects 18 additional metrics including: S&P500, NASDAQ, DOW, Russell 2000, Foreign exchange rates (Yen-USD, Euro-USD, etc.), LIBOR rates (1M, 3M, 12M), commodity prices (gold, crude oil). Details of these different metrics can be found at `https://fred.stlouisfed.org/series/METRICNAME` e.g.

https://fred.stlouisfed.org/series/BAMLHYH0A0HYM2TRIV

The downloaded data can be found in `data/noncrypto.csv`

The combination of the two datasets results in over 600 observations of 20 variables for a 2.5 year period.

(REQ: lots of columns, samples can be taken)

Categorical Data

Since our data was entirely numeric, we engineered two categorical features.

We converted VIX, the Volatility Index into a logical variable, where +1 represents an increase and -1 represents a decrease in volatility respectively.

We converted daily variation in pricing for a subset of the data into a categorical variable representing number of standard deviations.

(REQ: at least two categorical or logical columns)

Clean up, Normalization

In order to process the cryptocurrency data, we performed the following transformations:

- Converted date strings into native R format
- Converted volume and market capital strings ("1,234") into R numeric fields

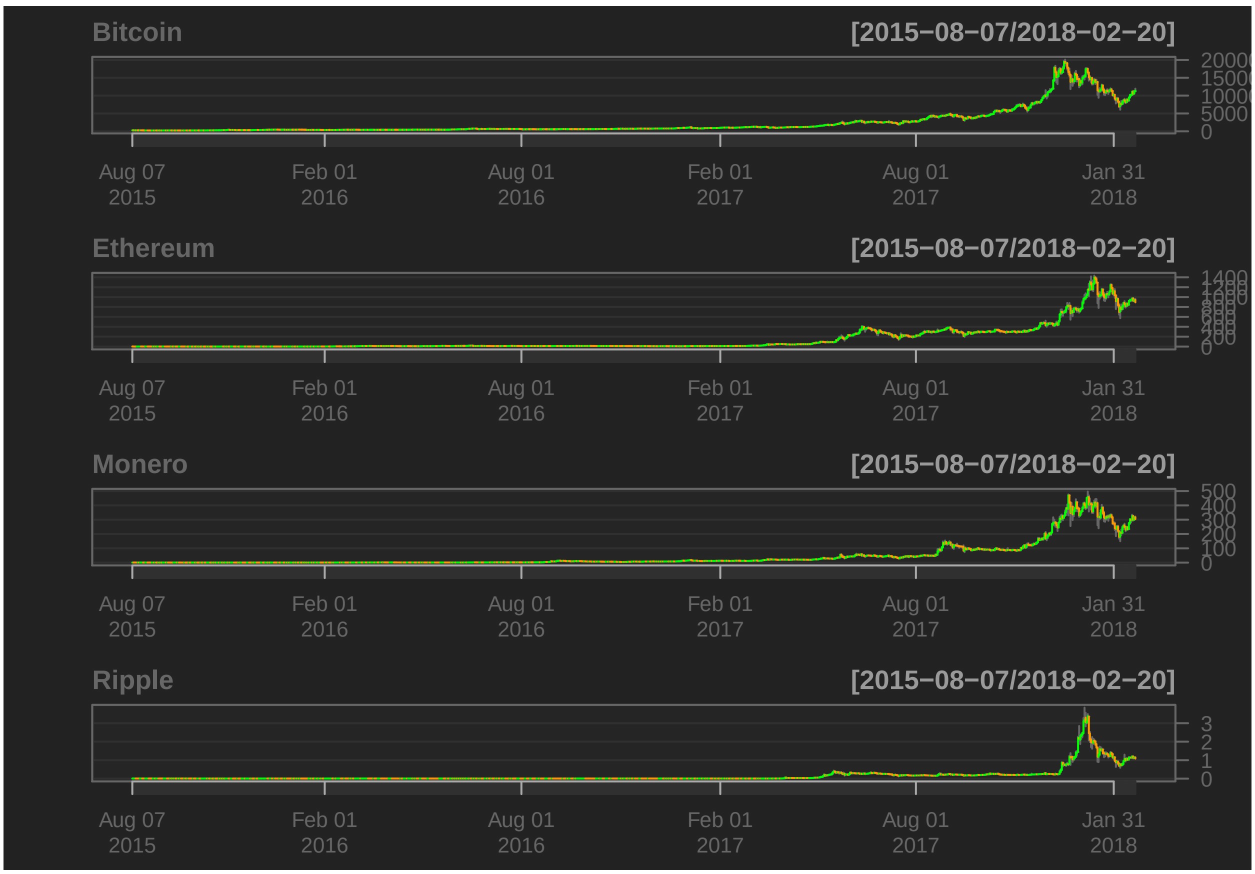
- Selected only a subset of data for which we had all values. This meant discarding early pricing information for older currencies such as Bitcoin. Since the market dynamics have changed significantly between 2010-2014 vs 2015-2016 vs late 2017-early 2018, we felt this was a worthwhile tradeoff. Modeling Bitcoin when it had 1/1000 the market capitalization would not be useful for either predictive or analytical purposes. In sum, our data ranged from 2015-08-07 until 2018-02-20.
- Sorted all data chronologically
- In order to account for holidays and other days when traditional markets are closed but cryptocurrency markets are open, we imputed values for the closed market days.

Depending on the context, we both standardized $\frac{x-\bar{x}}{\sigma}$ or normalized $\frac{x-x_{min}}{x_{max}-x_{min}}$ the data to perform appropriate comparison. For specific details, see the appropriate comparison.

Exploratory Data Analysis

Prices over time

For initial exploration, we examine closing price and volume over time for the four cryptocurrencies.

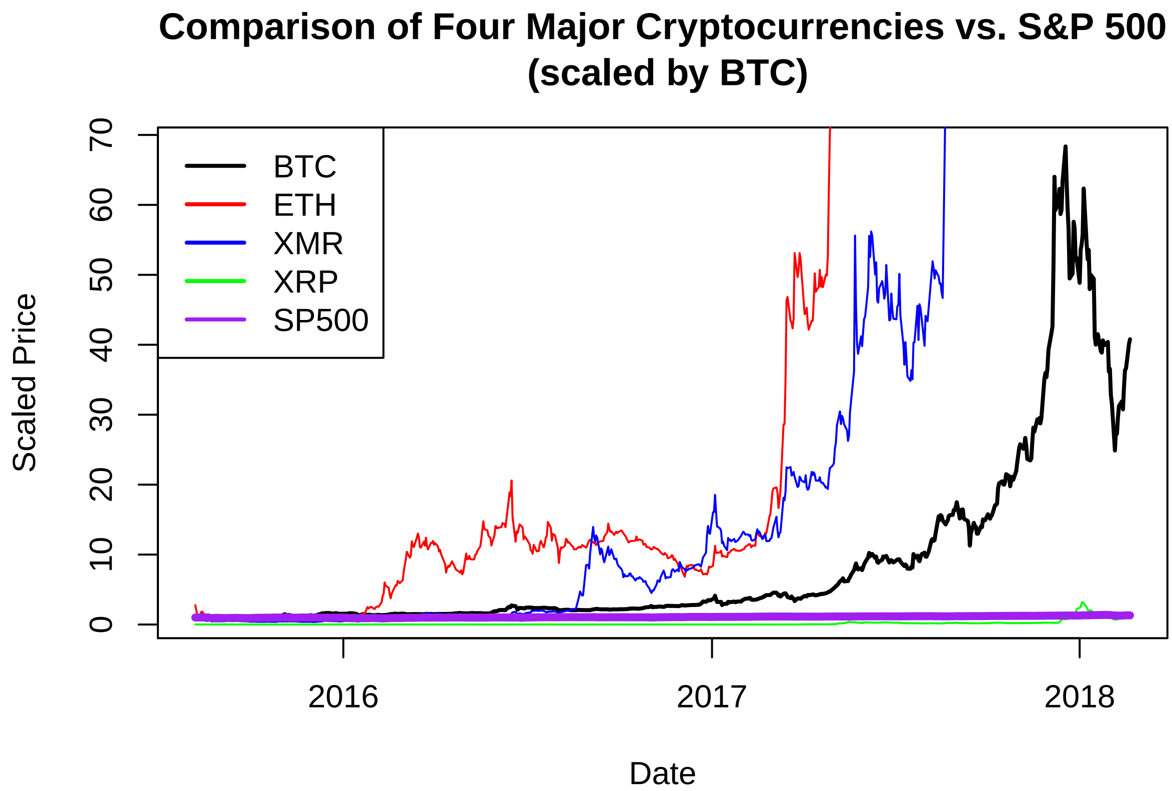


From this comparison, there some commonalities, but certain currencies seem to rise earlier than others.

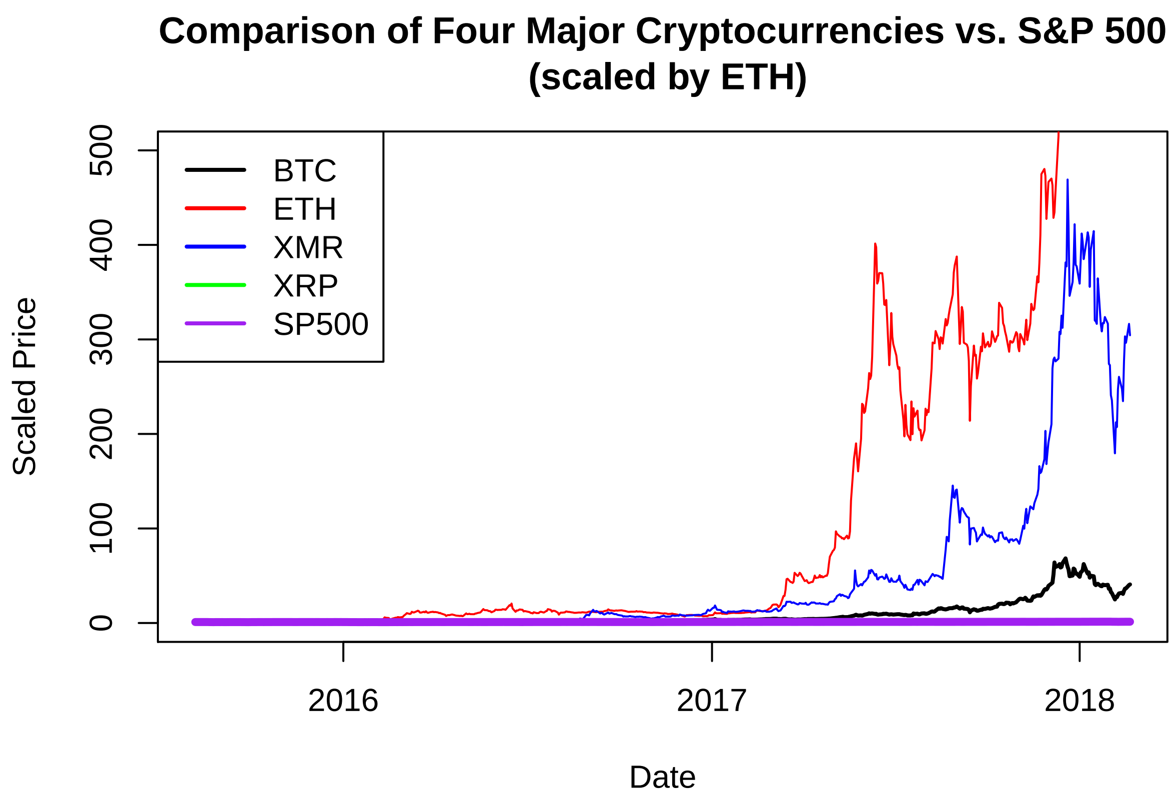
(REQ: A graphical display that is different from those in the textbook or in the class scripts.)

However, if we overlay these change in prices on the same plot, and rescale to a common starting point, the differences become very visible.

The following plot scales to the most popular cryptocurrency: Bitcoin (BTC)



Now, if we scale by the currency that rose most aggressively, the story appears different.



The S&P 500 looks like there was no return at all, even though this period was one of the best bull markets in history. Clearly, the cryptocurrency dynamics in 2017 into early 2018 are significantly different from traditional price movements. Let's now quantify more precisely what these differences are and see which probability distributions appropriately model this behavior.

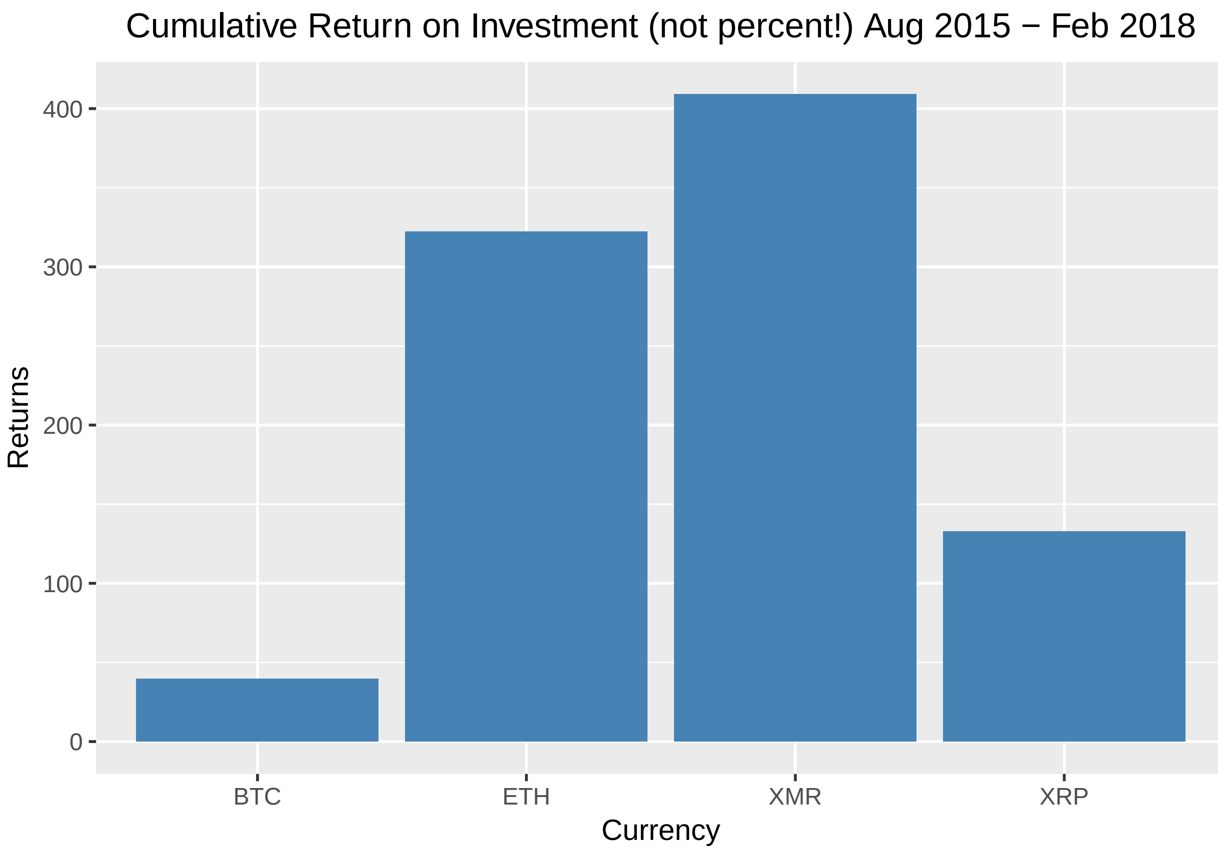
ROI for Cryptocurrencies

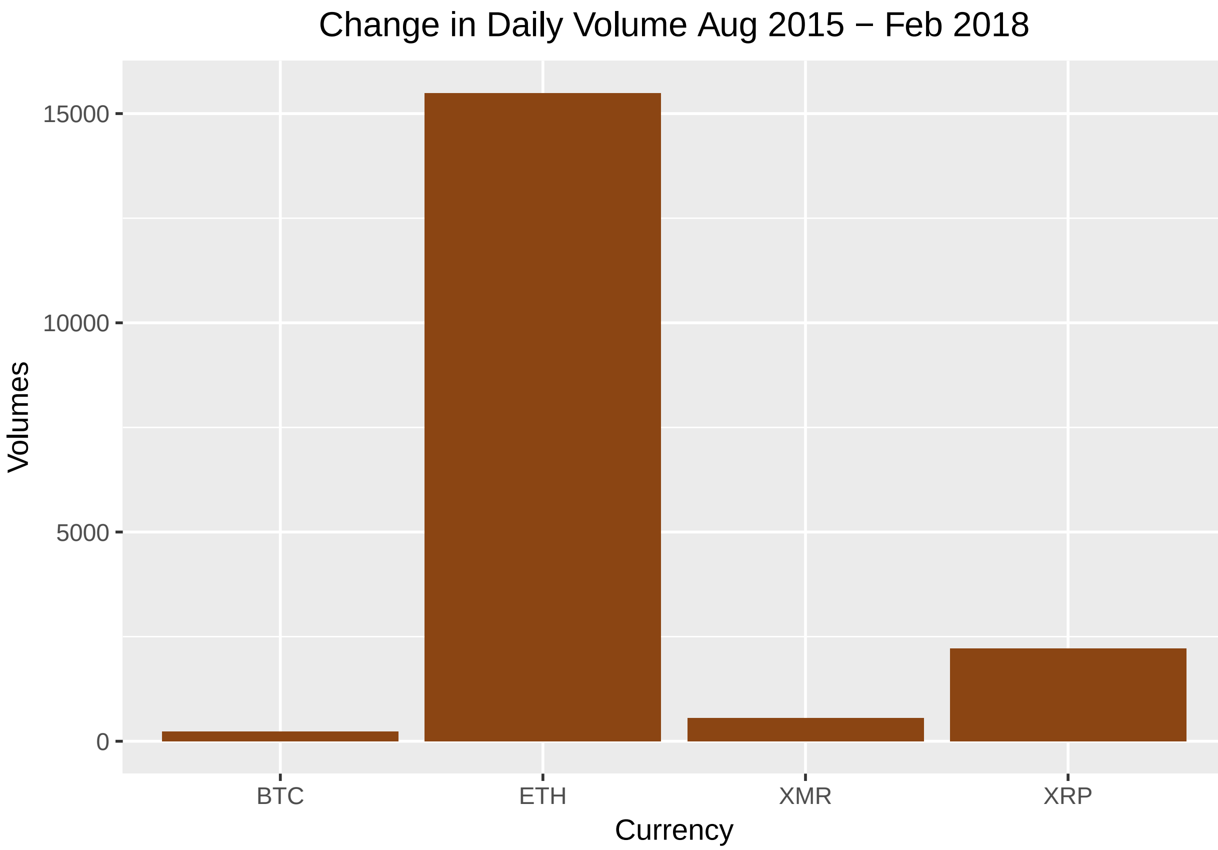
So what is the actual ROI for cryptocurrencies in this period:

Table 1: Cryptocurrency ROI

Currency	Returns	Volumes
BTC	39.79	232.65
ETH	322.24	15487.81
XMR	409.08	551.93
XRP	132.71	2217.31

With numbers like these, there is no wonder that there was rampant speculation! It was like buying lottery tickets! Here is a graphical representation of these returns:





(REQ: barplot)

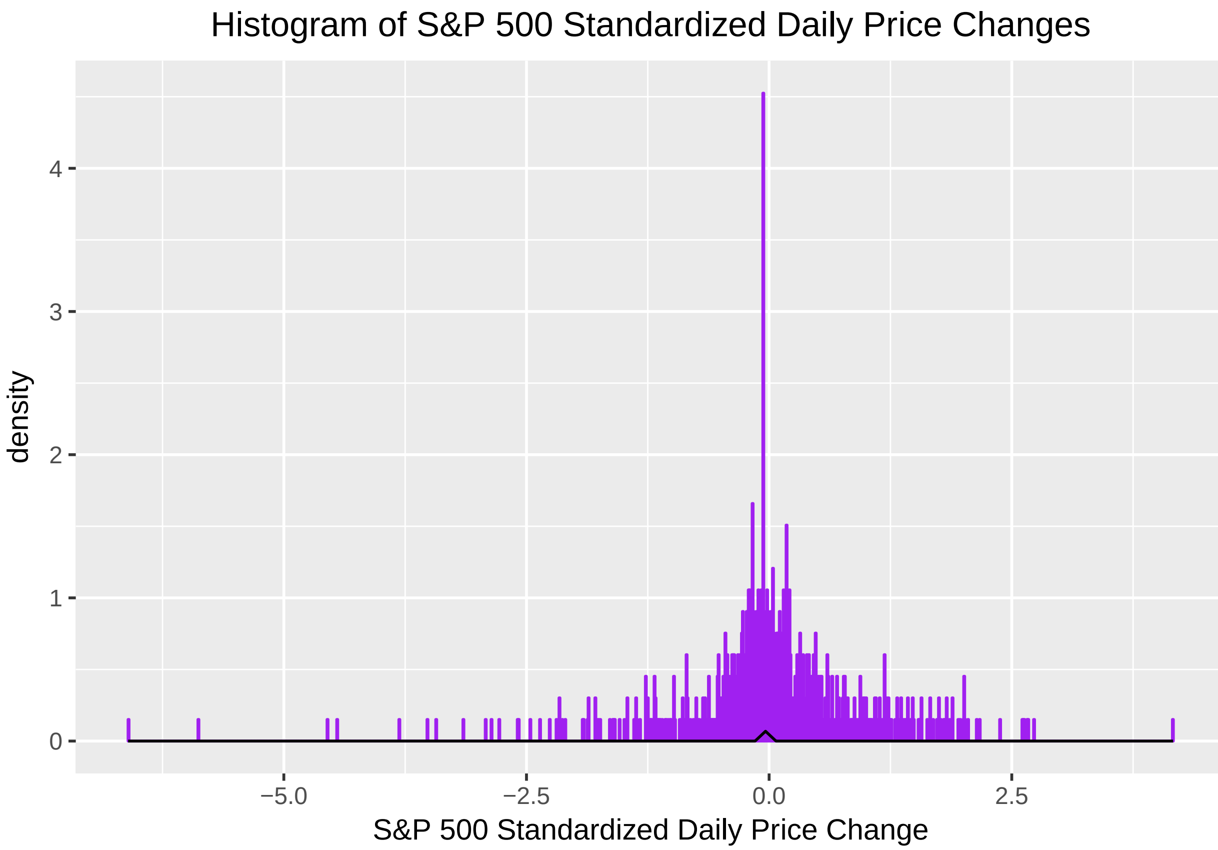
Topic 1: Distribution of Relative Price Changes

Distribution of S&P 500 Price Changes

The S&P 500 index is a broad index representing 500 large companies having stock listed on major US Stock Exchanges. It is generally considered a reasonably accurate proxy for the US Stock Market.

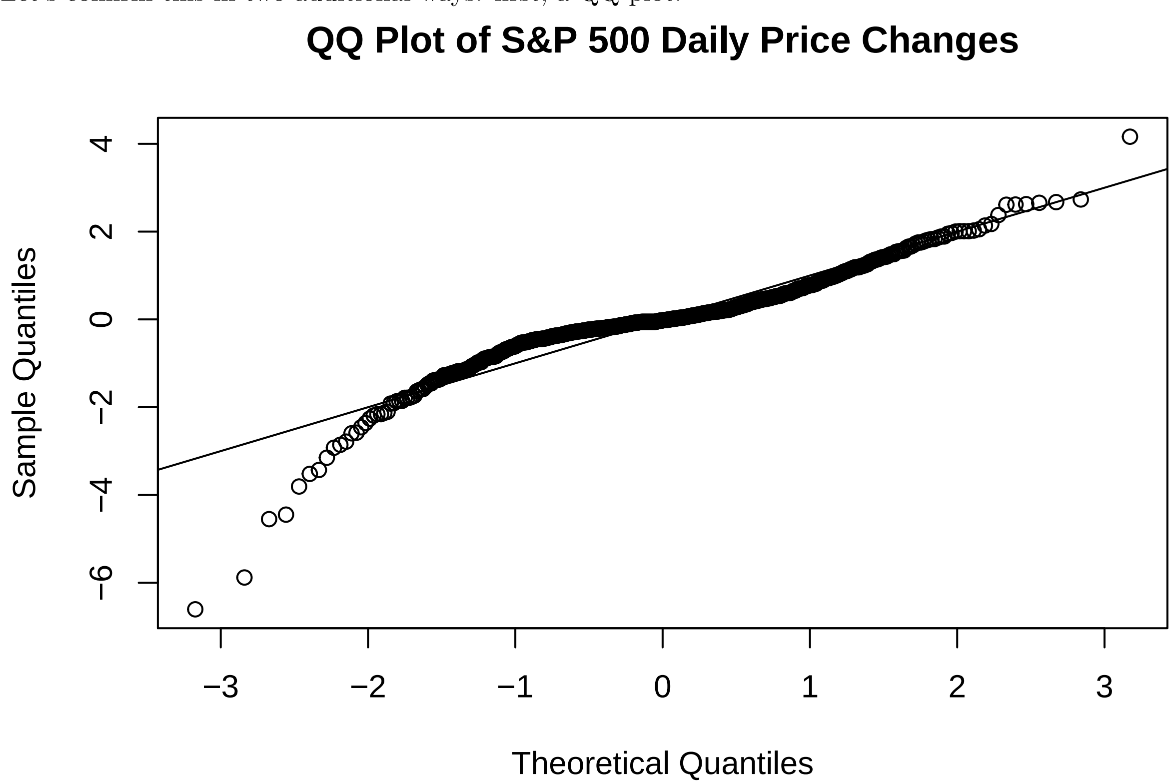
Let's first examine the distribution of relative price changes for the S&P 500. Classic economic theory, notably by A Random Walk Down Wall Street claims that price movements are random and should follow a Gaussian distribution. However, as shown in the empirical plot, the theory is incorrect.

The daily price changes have been standardized, and yet the price fluctuations for this period are vastly different than the predicted stable prices shown in the little bump of the overlaid Gaussian distribution for $\mu = 0$, $\sigma = 0.0083192$. **Clearly this is a not a Gaussian Distribution**. This has been confirmed by other sources as well.



(REQ: A probability density graph overlaid on a histogram)

Let's confirm this in two additional ways: first, a QQ-plot.



This can also be confirmed analytically via the Shapiro-Wilk Normality test.

```
##
##  Shapiro-Wilk normality test
##
## data:  multidf$SP500deltaStd
## W = 0.90285, p-value < 2.2e-16
```

The $p - value < 0.05$ confirms that the distribution of price changes is not normally distributed.

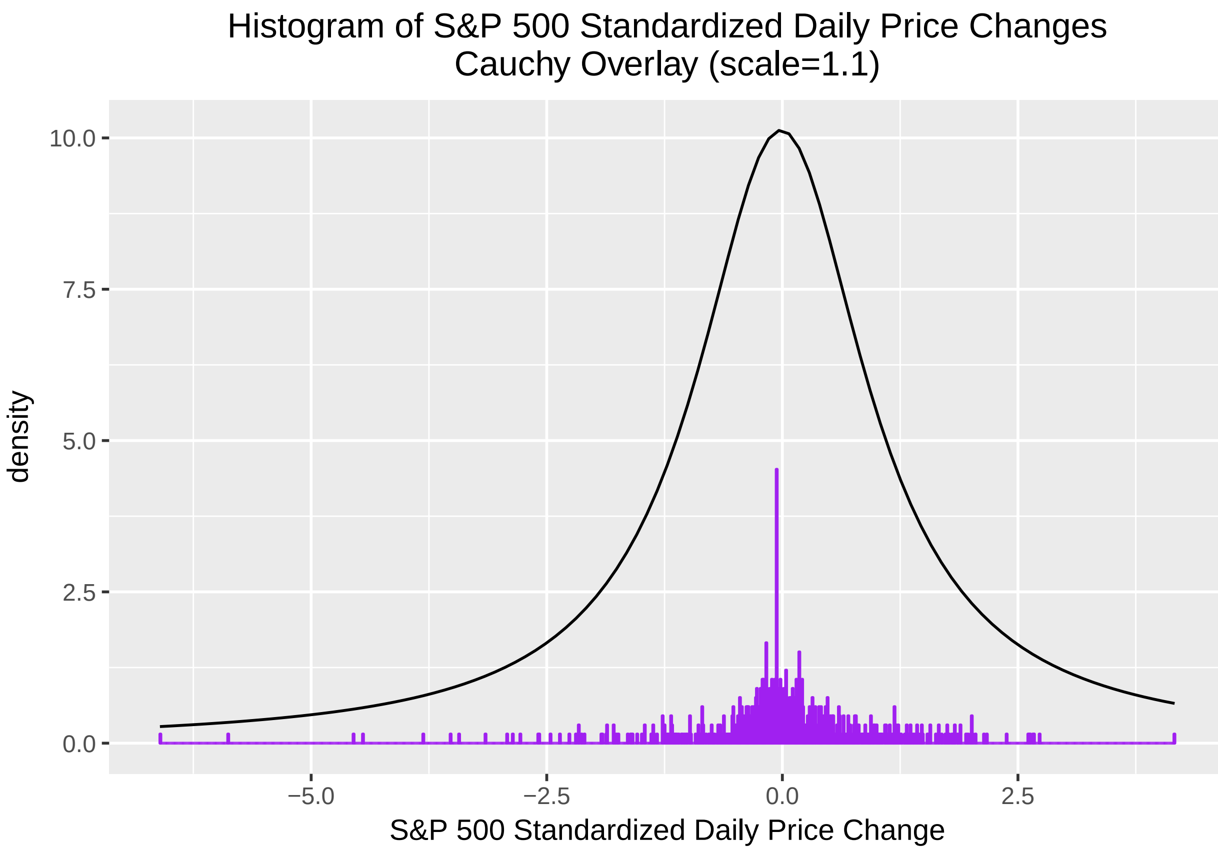
So by three different methods, it is clear that the Random Walk Down Wall St. is empirically wrong for time periods of at least 2.5 years, which is a long time to be wrong in the market.

(REQ: A p-value or other statistic based on a distribution function)

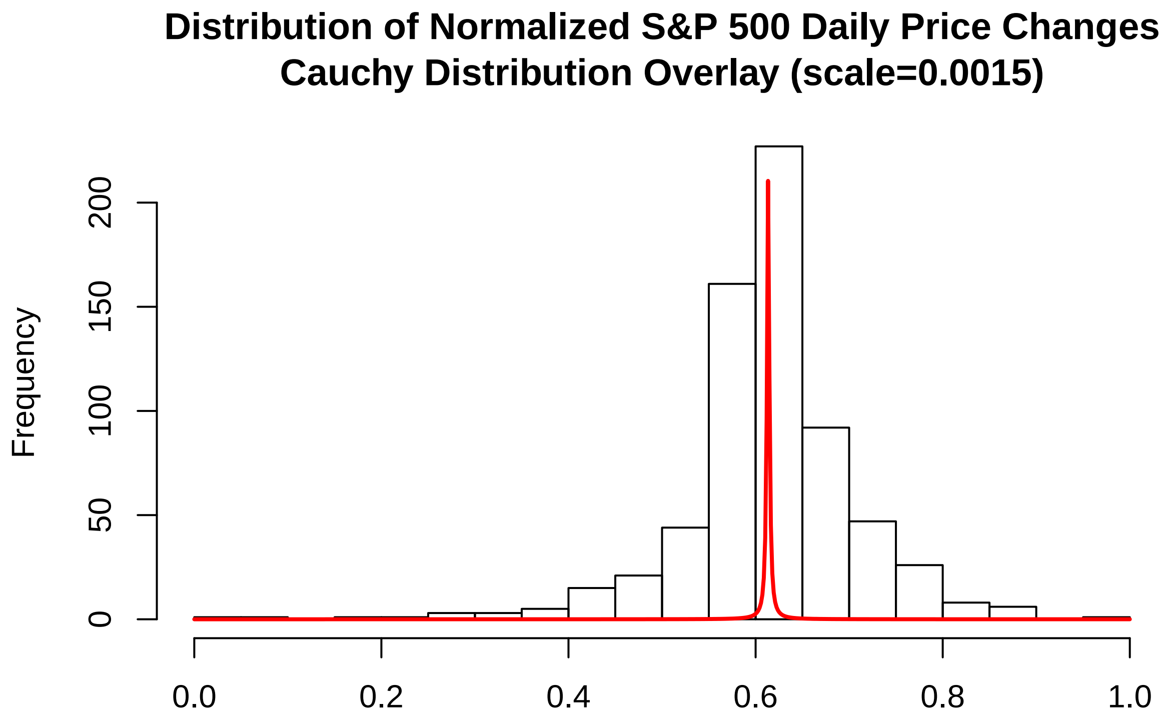
So what is the appropriate distribution for the market?

We attempted several different distributions to find the best fit: beta, cauchy, inverse gamma.

We used the R MASS package, with the `fitdistr` to find the optimal parameters (source)



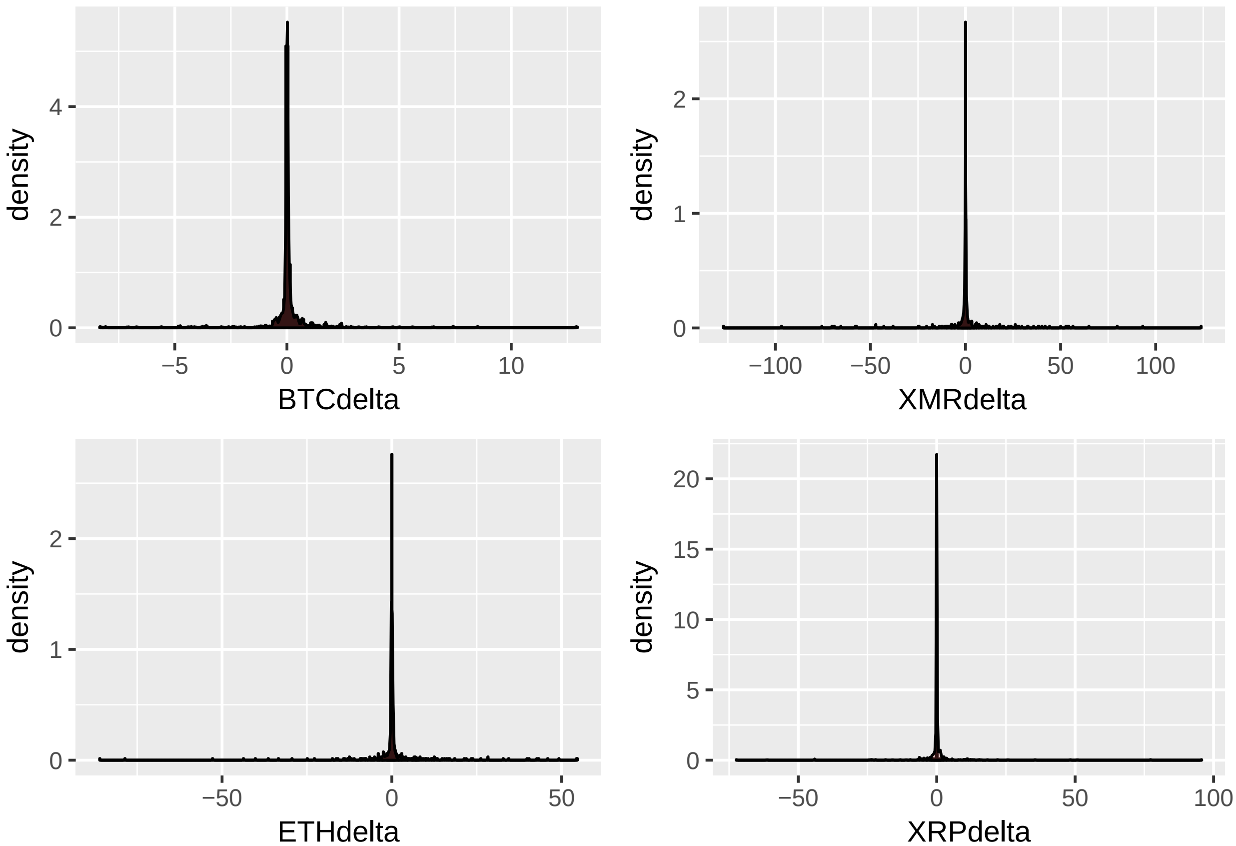
After many attempts, no single distribution was entirely satisfactory. The closest approximation was the Cauchy distribution over the normalized (rescaled to [0,1]) price change values as shown below.

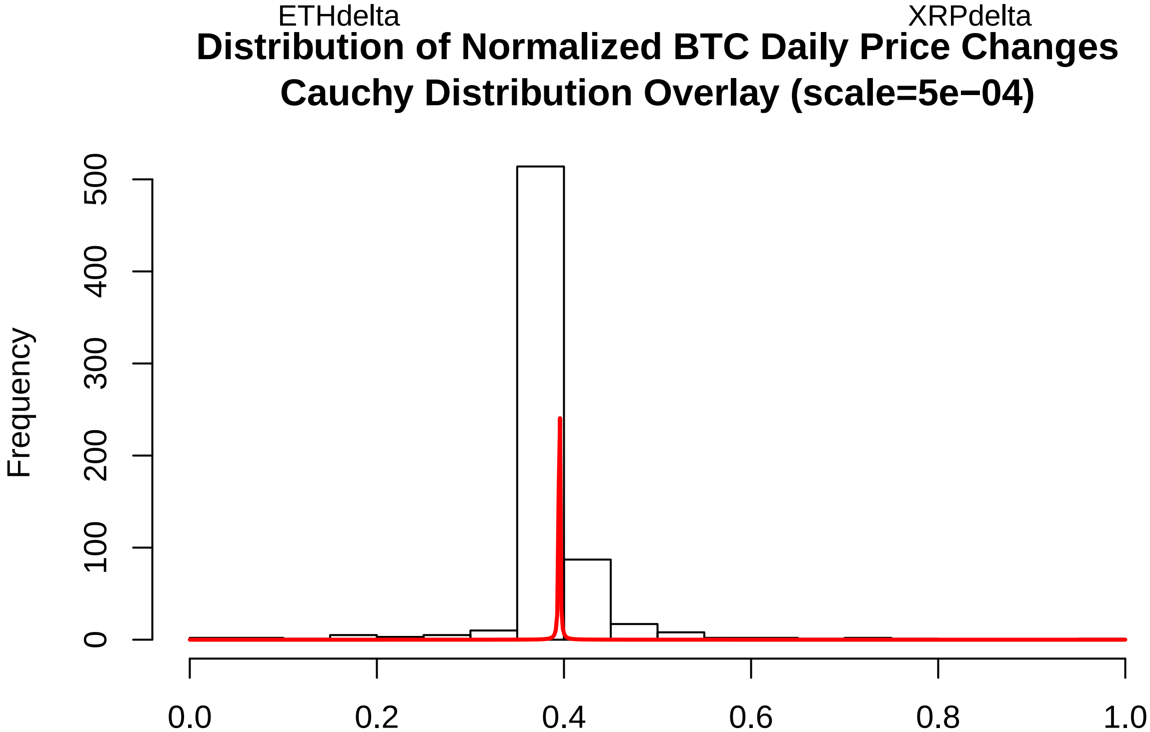


(REQ: Appropriate use of R functions for a probability distribution other than binomial, normal, or chi-square)

Distribution of Cryptocurrency Price Changes

We then attempted to see if the Cauchy distribution would adequately fit the higher variance cryptocurrency distributions. It came closer than any other distribution but was still not entirely satisfactory.





It is interesting to note that the Cauchy probability distribution has infinite variance. This probably reflects the uncertainty of the cryptocurrency market!

Topic 2 - Correlation between Cryptocurrencies

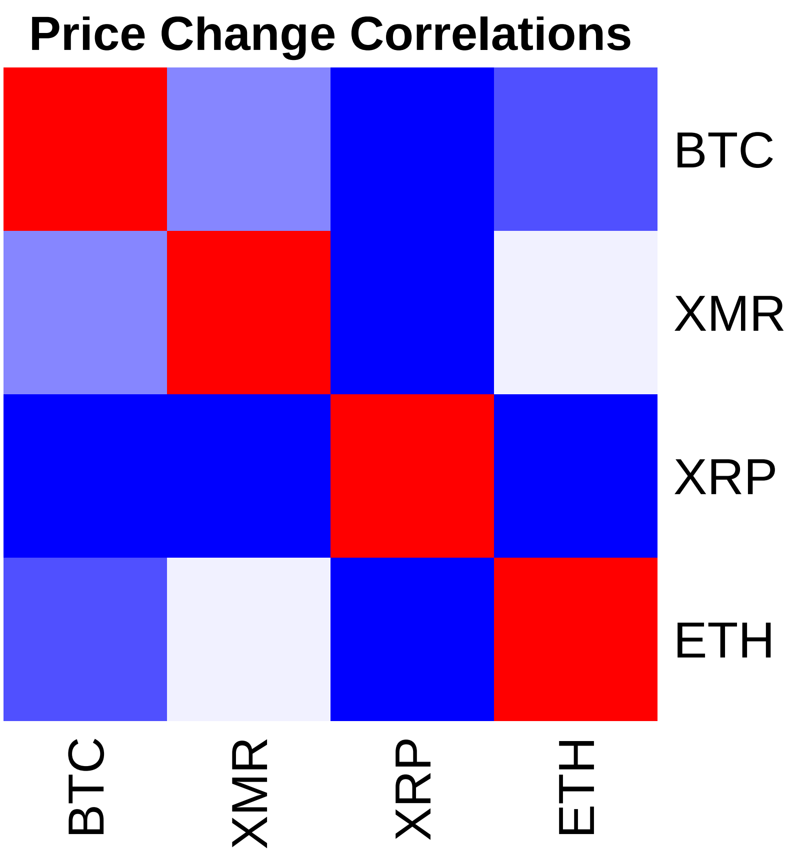
Correlation of Cryptocurrency Prices

In the Exploratory Data Analysis, we saw that cryptocurrencies had the same general trend and shape, but the timing could be different by months. For example, Ethereum (ETH) spiked months before Monero (XMR) which in turn spiked before Bitcoin (BTC). We'll now examine these correlations in more depth.

Table 2: Correlations of Cryptocurrency Closing Prices

Currency	Correlations
BTCXMR	0.97
ETHXMR	0.95
BTCETH	0.91
XMRXRP	0.88
ETHXRP	0.88
BTCXRP	0.80

A heatmap is an effective visualization of these correlations. To provide a slightly different perspective, we'll visualize the correlations of *standardized price changes* rather than just closing price.

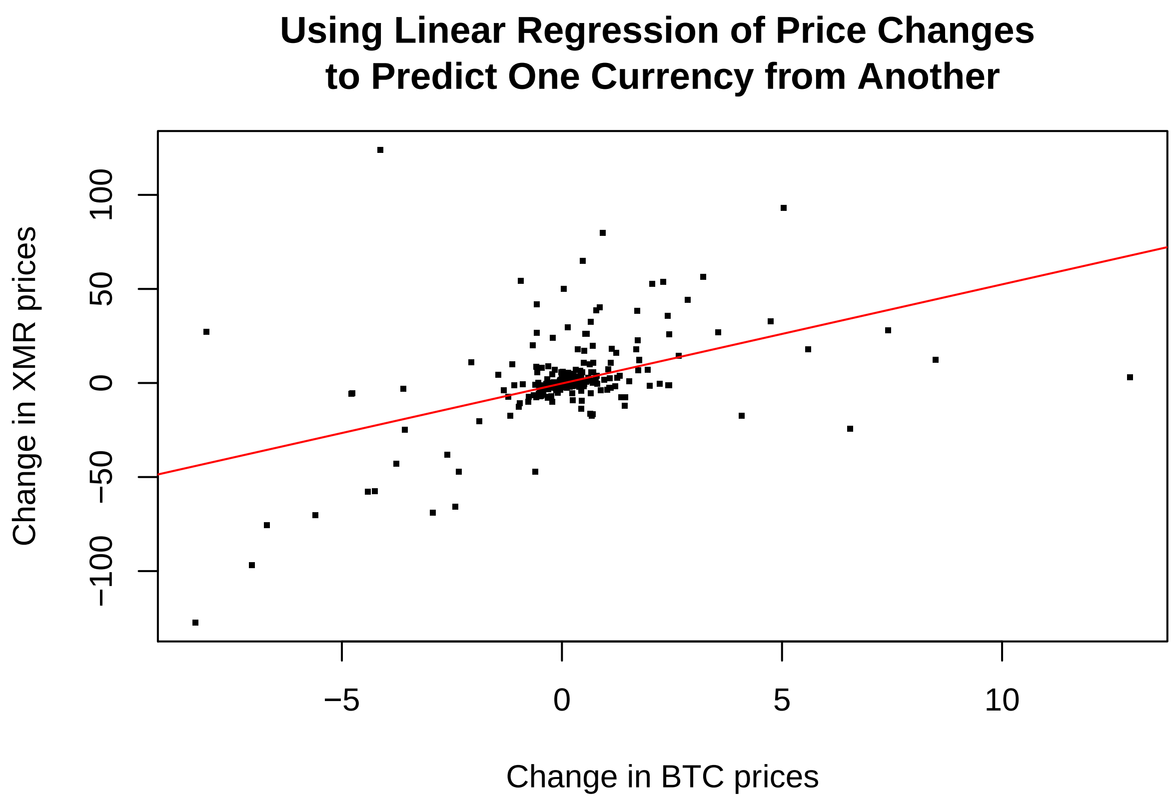


Interestingly, by correlation alone, we see a high degree of similarity. However, as we saw in the previous line charts of prices over time, *these correlations do not occur at the same time* leading investment opportunities if certain currencies are consistently leading indicators for the other currencies.

(REQ: appropriate use of correlation)

Using Linear Regression to Predict Prices Changes

For the most highly correlated currency pair (BTCXMR), let's examine if change in prices in one can be used to predict change in prices in the other via linear regression:



As the above plot shows, due to the wide spread away from the line of best fit, linear regression would not be a good means of predicting price changes even for the most highly correlated currency pair.

Running a linear regression in R yields:

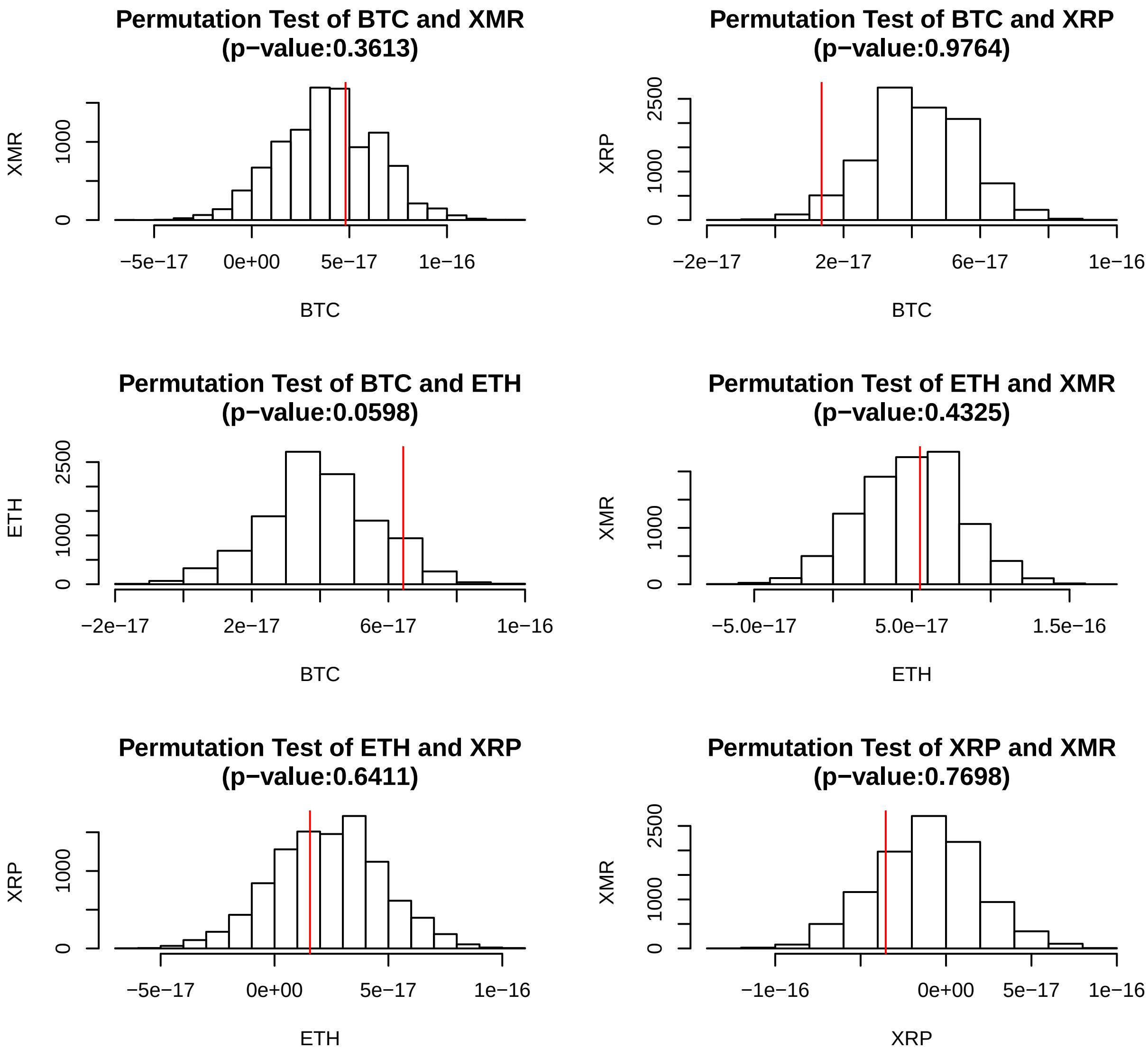
```
##
## Call:
## lm(formula = XMRdelta ~ BTCdelta, data = multidf)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -83.757  -0.864  -0.335   0.028 145.295
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.3008     0.4990   0.603    0.547
## BTCdelta      5.2683     0.3994  13.189   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12.83 on 661 degrees of freedom
## Multiple R-squared:  0.2083, Adjusted R-squared:  0.2071
## F-statistic:   174 on 1 and 661 DF,  p-value: < 2.2e-16
```

An R^2 of 0.21 shows a weakly predictive model. Clearly, correlation does not tell a sufficient story about the relationship between two currencies.

(REQ: use of linear regression)

Permutation Test

We will now use a permutation test as another means of evaluating correlation between price movements of the currencies.



In all cases, $p - value > 0.05$, we fail to reject the null hypothesis. So the permutation tests, like correlation, show that the distributions of price changes are very similar even though the time periods for these correlations are significantly different. So here we have two examples where the aggregate statistics (correlation, permutation test) do not tell the complete story of multiple variables that change over time.

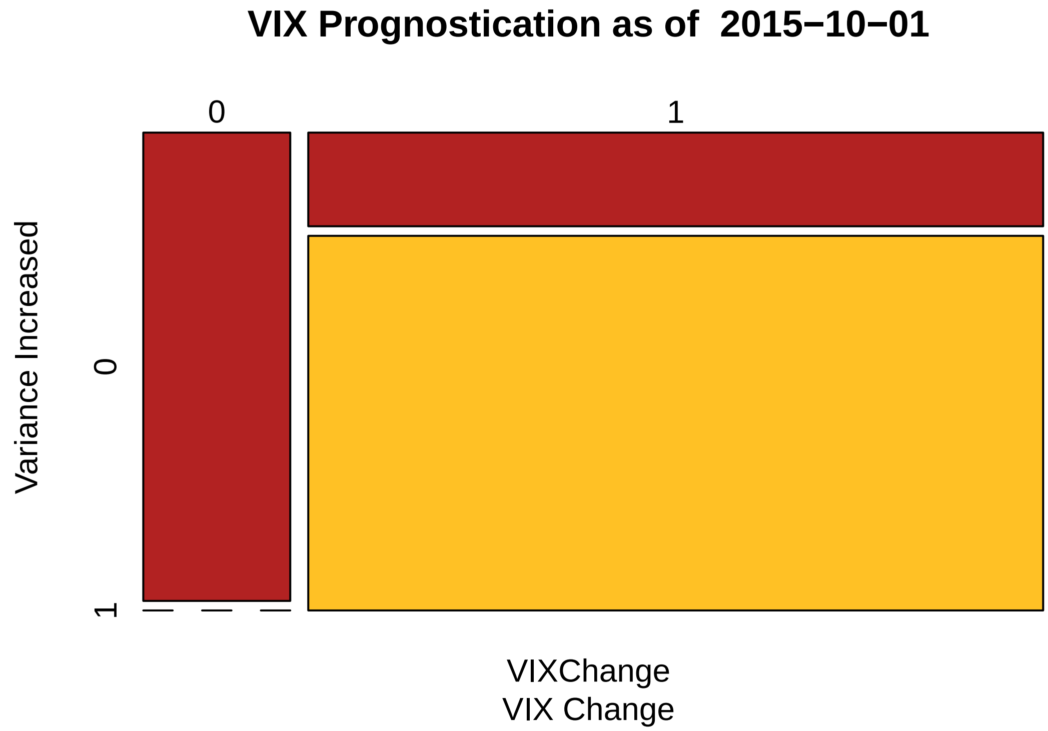
(REQ: a permutation test, “A convincing demonstration of a relationship that might have been statistically significant but that turns out not to be so”)

Gauging VIX Accuracy Using a Contingency Table

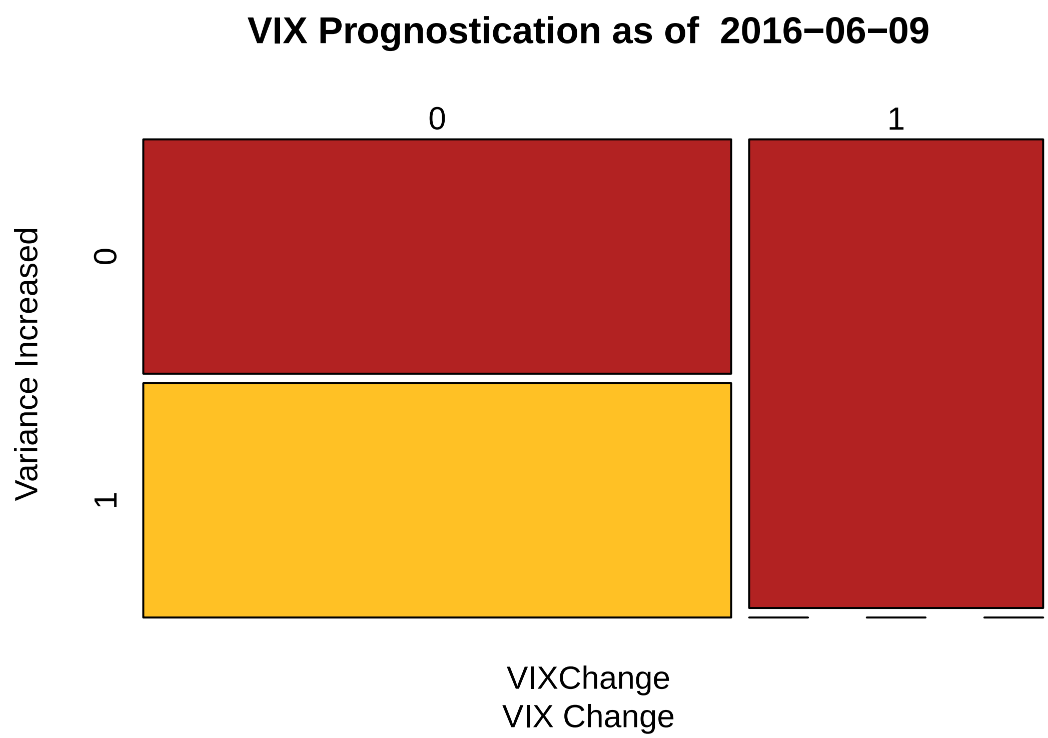
The VIX is a measure of constant 30-day expected volatility of the US Stock market. In market terms, volatility is precisely equivalent to variance in statistical terms.

VIX is primarily based on a perception of the market and has no ground truth on a tangible asset.

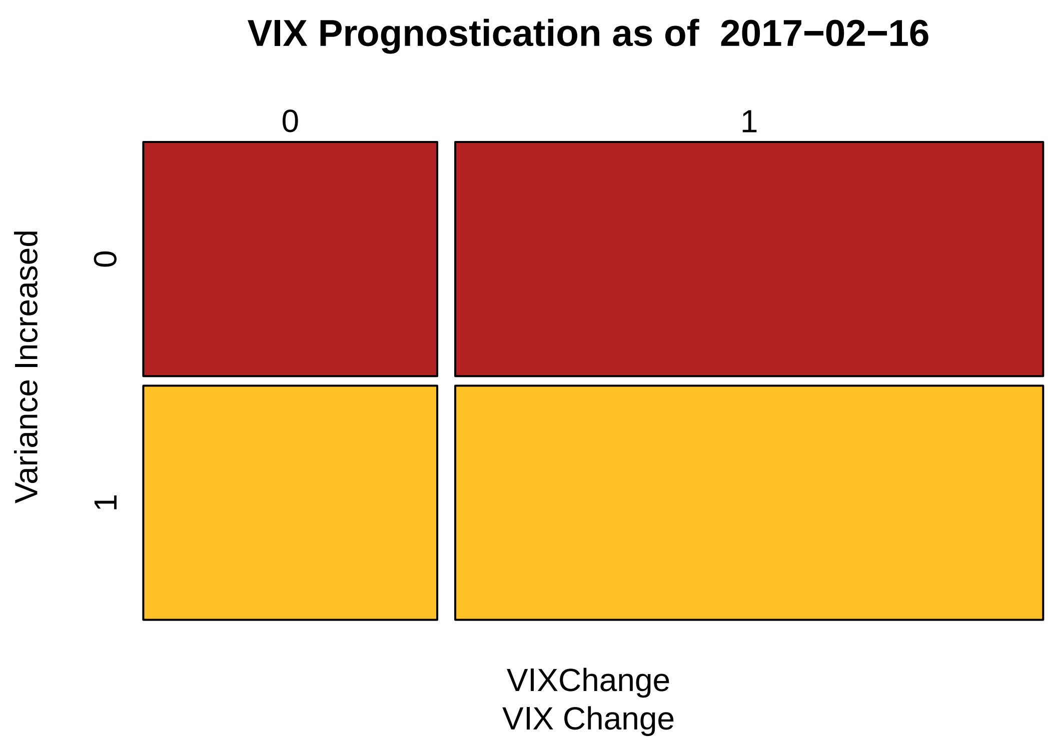
Using the change in VIX, we wanted to determine if it is indeed a reliable prognosticator of change of variance in the overall market. An increase in VIX should manifest as an increase in variance of the S&P500 price changes from one 30 day period to another. In order to empirically validate the value of VIX, we took the change in VIX over a three day period and measured the variance of the market for 30 days before and after this three day period. We evaluated this at three separate time periods.




```
##          hasVarInc
## VIXChange 0 1
##         0 1 0
##         1 1 4
```




```
##           hasVarInc
## VIXChange 0 1
##         0 2 2
##         1 2 0
```




```
##           hasVarInc
## VIXChange 0 1
##         0 1 1
```

```
##            1 2 2
```

Over the three time periods, VIX was not significantly better than random at forecasting change in volatility. Since VIX was not useful at forecasting volatility of the market for which it was designed, we did not attempt to see if it could be further used to forecast volatility of the more unstable cryptocurrency market.

(REQ: Contingency table and related analysis)

Summary of Project Requirements

Required dataset standards

- [x] A dataframe
- [x] At least two categorical or logical columns
- [x] At least two numeric columns
- [x] At least 20 rows, preferably more, but real-world data may be limited

Required graphical displays (all graphs must be colored and nicely labeled)

- [x] A barplot
- [x] A histogram
- [x] A probability density graph overlaid on a histogram
- [x] A contingency table

Required Analysis

- [x] A permutation test
- [x] A p-value or other statistic based on a distribution function (Shapiro-Wilk)
- [x] Analysis of a contingency table
- [x] Comparison of analysis by classical methods (chi-square, CLT) and simulation methods

Required submission uploads

- [x] A .csv with the dataset
- [x] A long, well-commented script that loads the dataset, explores it, and does all the analysis.
- [x] A shorter .Rmd with compiled .pdf or .html file that presents highlights in ten minutes.
- [x] A one-page handout that explains the dataset and summarizes the analysis.

Additional points for creativity or complexity (up to 10 points)

- 1. A data set with lots of columns, allowing for comparison of many different variables (20 columns)
- 2. A data set that is so large that it can be used as a population from which samples are taken (over 600 rows)
- 4. A one-page document that discusses ethical issues raised by conclusions reached from analysis of the data. (see separate document)
- 5. A graphical display that is different from those in the textbook or in the class scripts. (many!)
- 6. Appropriate use of R functions for a probability distribution other than binomial, normal, or chi-square (Cauchy, others attempted)
- 9. A convincing demonstration of a relationship that might have been statistically significant but that turns out not to be so. (See Correlation Discussion, Permutation Test)
- 10. Professional-looking software engineering (e.g defining and using your own functions). (many!)

- 11. Nicely labeled graphics using ggplot, with good use of color, line styles, etc., that tell a convincing story. (many)
- 13. Appropriate use of novel statistics (e.g. trimmed mean, maximum or minimum, skewness, ratios). (Shapiro-Wilk)
- 14. Use of linear regression. (see Correlation Discussion)
- 16. Appropriate use of covariance or correlation. (See Correlation Discussion)
- 22. Team consists of exactly two members (otherwise, 1 or 3 is a possibility). (Just the two of us, standing on the shoulders of giants)

Future Work

We did not perform a spectral analysis, but this would another interesting mechanism to determine if patterns in the data can be modeled. We ended up using only a few of the other non-cryptocurrency metrics as there was already plenty of analysis to do. We would like to explore these techniques with the other metrics as well both for cryptocurrency and traditional market insights.

Conclusion

There is only one possible conclusion: immediately disband the search committee and hire us!

Acknowledgements

We'd like to thank Paul Bamberg, Joe Palin, Rachel Gologorsky, Mihika Shilpi and the other Math 23C staff for a challenging, interesting and very educational class.

References

The full version of this project can be found at https://github.com/wihl/math23c-project

Berlinger et al. *Mastering R for Quantitative Finance*, Packt Publishing, 2015

Clifford S. Ang, *Analyzing Financial Data and Implementing Financial Models Using R*, Springer, 2015

Mao Xin, "The VIX Volatility Index", Project Report 2011, url: https://uu.diva-portal.org/smash/get/diva2:417612/FULLTEXT01.pdf

Ngai Hang Chan, *Time Series - Applications to Finance with R and S-Plus*, Wiley 2010

Appendix A - Long Script

This is the "long script" used to prepare these reports:

```r
knitr::opts_chunk$set(echo = TRUE)
# Exteral libraries
library(xts)
library(quantmod)
library(ggplot2)
library(knitr)
# Save parameters for later reset
op <- par(no.readonly = TRUE)
# Utility function to standardize a vector
standarize <- function(v) {
  mu_v = mean(v)
  sd_v = sd(v)
  return (v - mu_v) / sd_v
}

# Read one price history file per currency
BTCdf = read.csv("data/bitcoin_price.csv", stringsAsFactors = F)
ETHdf = read.csv("data/ethereum_price.csv", stringsAsFactors = F)
XMRdf = read.csv("data/monero_price.csv", stringsAsFactors = F)
XRPdf = read.csv("data/ripple_price.csv", stringsAsFactors = F)
# Fix rest of data:
# 1- Make dates native format
# 2- Convert Volume and market cap:
#     a) From string ("123,456") to numeric (123456).
#     b) Convert "-" to 0.
# 3- Sort chronologically
fixVolCap = function(df) {
  df$Date = as.Date(df$Date,"%b %d, %Y")
  df$Volume = as.numeric(gsub("-","0",gsub(",","",df$Volume)))
  df$Market.Cap = as.numeric(gsub("-","0",gsub(",","",df$Market.Cap)))
  return (df[order(df$Date),])
}
BTCdf = fixVolCap(BTCdf)
ETHdf = fixVolCap(ETHdf)
XMRdf = fixVolCap(XMRdf)
XRPdf = fixVolCap(XRPdf)

# Ensure that all data start from the same date
earliestCommonDate =  max(min(BTCdf$Date),
                          min(ETHdf$Date),
                          min(XMRdf$Date),
                          min(XRPdf$Date))
BTCdf = BTCdf[BTCdf$Date>=earliestCommonDate,]
ETHdf = ETHdf[ETHdf$Date>=earliestCommonDate,]
XMRdf = XMRdf[XMRdf$Date>=earliestCommonDate,]
XRPdf = XRPdf[XRPdf$Date>=earliestCommonDate,]


# Read in traditional, noncryptocurrency data
noncrypto = read.csv("data/noncrypto.csv",stringsAsFactors = FALSE)
```

```r
noncrypto$DATE = as.Date(noncrypto$DATE,"%Y-%m-%d")
# Plot price and volume data over time
plotSeries = function(df,label){
  dfdata = xts(df[,2:7],order.by = df[,1])
  OHLC = as.quantmod.OHLC(dfdata)
  chartSeries(OHLC,name=label,layout=NULL, TA=NULL)
}
layout(matrix(1:4,nrow=4))
plotSeries(BTCdf, "Bitcoin")
plotSeries(ETHdf, "Ethereum")
plotSeries(XMRdf, "Monero")
plotSeries(XRPdf, "Ripple")

# Weekly Comparison - not used
# wk = BTCdata
# data.wk = to.weekly(wk)
# plot(data.wk)
# OHLC = as.quantmod.OHLC(BTCdata)
# chartSeries(OHLC)


# Extract closing prices
BTCdata = xts(BTCdf[,2:7],order.by = BTCdf[,1])
close.prices = BTCdata$Close
close.prices = cbind(close.prices,ETHdf$Close,XMRdf$Close,XRPdf$Close)

# Create one new dataframe to compare closing prices of both crypto and non-crypto securities
multidf = cbind(index(close.prices), data.frame(close.prices))
names(multidf) = paste(c("Date","BTC","ETH","XMR","XRP"))
# Merge in the non-crypto metrics
multidf=merge(multidf,noncrypto,by.x="Date", by.y="DATE")

# Create a new column with the relative price changes (scaled to the starting price)
multidf$BTC.idx = multidf$BTC / multidf$BTC[1]
multidf$ETH.idx = multidf$ETH / multidf$ETH[1]
multidf$XMR.idx = multidf$XMR / multidf$XMR[1]
multidf$XRP.idx = multidf$XRP / multidf$XRP[1]
multidf$SP500.idx = multidf$SP500 / multidf$SP500[1]
multidf$GOLDAMGBD228NLBM.idx = multidf$GOLDAMGBD228NLBM / multidf$GOLDAMGBD228NLBM[1]
# Plot changes in pricing over time
# 1- default y scale
par(op)
plot(x = multidf$Date,y=multidf$BTC.idx,type="l",xlab="Date",col="black",lty=1,lwd=2,
     main="Comparison of Four Major Cryptocurrencies vs. S&P 500\n (scaled by BTC)",
     ylab="Scaled Price")
lines(x=multidf$Date,y=multidf$ETH,col="red")
lines(x=multidf$Date,y=multidf$XMR,col="blue")
lines(x=multidf$Date,y=multidf$XRP,col="green")
lines(x=multidf$Date,y=multidf$SP500.idx,col="purple",lwd=4)
legend("topleft",c("BTC","ETH","XMR","XRP","SP500"),col=c("black","red","blue","green","purple"),
       lty=c(1,1,1,1,1),
       lwd=c(2,2,2,2,2))
# 2- y scale from 0-500
```

```r
plot(x = multidf$Date,y=multidf$BTC.idx,type="l",xlab="Date",col="black",lty=1,lwd=2,ylim=c(0,500),
     main="Comparison of Four Major Cryptocurrencies vs. S&P 500\n (scaled by ETH)",
     ylab="Scaled Price")
lines(x=multidf$Date,y=multidf$ETH,col="red")
lines(x=multidf$Date,y=multidf$XMR,col="blue")
lines(x=multidf$Date,y=multidf$XRP,col="green")
lines(x=multidf$Date,y=multidf$SP500.idx,col="purple",lwd=4)
legend("topleft",c("BTC","ETH","XMR","XRP","SP500"),col=c("black","red","blue","green","purple"),
       lty=c(1,1,1,1,1),
       lwd=c(2,2,2,2,2))
# Display bar plots showing Overall Return and Change in Daily Volume
overallReturn = function(df){
  return ((df$Close[nrow(df)] - df$Close[1]) / df$Close[1])
}

volIncrease = function(df){
  return ((df$Volume[nrow(df)] - df$Volume[1]) / df$Volume[1])
}

returns = c(overallReturn(BTCdf),overallReturn(ETHdf),overallReturn(XMRdf),overallReturn(XRPdf))
volumes = c(volIncrease(BTCdf),volIncrease(ETHdf),volIncrease(XMRdf),volIncrease(XRPdf))

barData = data.frame(Currency=c("BTC","ETH","XMR","XRP"), Returns=returns,Volumes=volumes)
kable(barData, caption="Cryptocurrency ROI",
      row.names = F,digits=2)
# REQ: barplot
ggplot(data=barData, aes(x=Currency, y=Returns)) +
  geom_bar(stat="identity",fill="steelblue") +
  ggtitle("Cumulative Return on Investment (not percent!) Aug 2015 - Feb 2018") +
  theme(plot.title = element_text(hjust = 0.5))

ggplot(data=barData, aes(x=Currency, y=Volumes)) +
  geom_bar(stat="identity",fill="chocolate4") +
  ggtitle("Change in Daily Volume Aug 2015 - Feb 2018") +
  theme(plot.title = element_text(hjust = 0.5))

# distribution of relative price changes
multidf$BTCdelta = c(0,diff(multidf$BTC.idx))
multidf$ETHdelta = c(0,diff(multidf$ETH.idx))
multidf$XMRdelta = c(0,diff(multidf$XMR.idx))
multidf$XRPdelta = c(0,diff(multidf$XRP.idx))
multidf$SP500delta = c(0,diff(multidf$SP500.idx))

# REQ: display a histogram
par(op)
mu_SP500delta = mean(multidf$SP500delta)
sd_SP500delta = sd(multidf$SP500delta)
multidf$SP500deltaStd = (multidf$SP500delta - mu_SP500delta) / sd_SP500delta
# Plot histogram of standardized daily price changes in S&P500, overlaid with equivalent
# Gaussian distribution.
ggplot(multidf, aes(x=SP500deltaStd)) +
  geom_histogram(aes(y=..density..),
                 binwidth=0.01,colour="purple",
```

```r
                 fill="white") +
  #geom_density(alpha=0.2,fill="#FF6666") +
  stat_function(
    fun = function(x,mean,sd,n){
      n*dnorm(x=x, mean=mean, sd=sd)
    },
    args=with(multidf, c(mean=mu_SP500delta, sd=sd_SP500delta, n=length(multidf)))
  ) +
  ggtitle("Histogram of S&P 500 Standardized Daily Price Changes") +
  theme(plot.title = element_text(hjust = 0.5)) +
  xlab("S&P 500 Standardized Daily Price Change")

# QQplot to see if a distribution is normal
qqnorm(multidf$SP500deltaStd,main="QQ Plot of S&P 500 Daily Price Changes")
abline(0,1)
shapiro.test(multidf$SP500deltaStd)
# Find a better distribution than Gaussian.
# Plot histogram of standardized daily price changes in S&P500, overlaid with equivalent
# Cauchy distribution.
ggplot(multidf, aes(x=SP500deltaStd)) +
  geom_histogram(aes(y=..density..),
                 binwidth=0.01,colour="purple",
                 fill="white") +
  #geom_density(alpha=0.2,fill="#FF6666") +
  stat_function(
    fun = function(x,mean,sd,n){
      n*dcauchy(x=x, location=-0.014, scale=1.1)
    },
    args=with(multidf, c(mean=mu_SP500delta, sd=sd_SP500delta, n=length(multidf)))
  ) +
  ggtitle("Histogram of S&P 500 Standardized Daily Price Changes\nCauchy Overlay (scale=1.1)") +
  theme(plot.title = element_text(hjust = 0.5)) +
  xlab("S&P 500 Standardized Daily Price Change")

# Overlay Cauchy
overlayCauchy = function(v,label,scale){
  # this will rescale vector v to [0,1]
  v_norm = (v - min(v)) / (max(v) - min(v))
  hist(v_norm,main=paste0(label,"\nCauchy Distribution Overlay (scale=",scale,")"),
       xlab="",breaks=20)
  xfit = seq(0, 1, length=length(v))
  yfit = dcauchy(x=xfit, location=mean(v_norm), scale=scale)
  lines(xfit,yfit,col="red", lwd=2)
}
overlayCauchy(multidf$SP500delta,"Distribution of Normalized S&P 500 Daily Price Changes",0.0015)


# From http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/
# Multiple plot function
#
# ggplot objects can be passed in ..., or to plotlist (as a list of ggplot objects)
# - cols:   Number of columns in layout
# - layout: A matrix specifying the layout. If present, 'cols' is ignored.
```

```r
#
# If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE),
# then plot 1 will go in the upper left, 2 will go in the upper right, and
# 3 will go all the way across the bottom.
#
multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
  library(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }

 if (numPlots==1) {
    print(plots[[1]])

  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                      layout.pos.col = matchidx$col))
    }
  }
}

p1 = ggplot(multidf, aes(x=BTCdelta)) +
  geom_histogram(aes(y=..density..),
                 binwidth=0.1,colour="black",
                 fill="white") +
  geom_density(alpha=0.2,fill="#FF6666")

p2 = ggplot(multidf, aes(x=ETHdelta)) +
  geom_histogram(aes(y=..density..),
                 binwidth=0.1,colour="black",
                 fill="white") +
  geom_density(alpha=0.2,fill="#FF6666")
```

```
p3 = ggplot(multidf, aes(x=XMRdelta)) +
  geom_histogram(aes(y=..density..),
                 binwidth=0.1,colour="black",
                 fill="white") +
  geom_density(alpha=0.2,fill="#FF6666")

p4 = ggplot(multidf, aes(x=XRPdelta)) +
  geom_histogram(aes(y=..density..),
                 binwidth=0.1,colour="black",
                 fill="white") +
  geom_density(alpha=0.2,fill="#FF6666")

multiplot(p1,p2,p3,p4,cols=2)
par(op)
overlayCauchy(multidf$BTCdelta,"Distribution of Normalized BTC Daily Price Changes",0.0005)


# Categorical Variables

# Converting VIX into a categorical
multidf$VIXCLS.idx = multidf$VIXCLS / multidf$VIXCLS[1]
multidf$VIXCLSdelta = c(0,diff(multidf$VIXCLS.idx))
multidf$VIXCLSsgn = ifelse(multidf$VIXCLSdelta>=0,1,-1)
multidf$BTCsgn = ifelse(multidf$BTCdelta>=0,1,-1)
# Not used
# overlayGaussian = function(v,label){
#   mu_v = mean(v)
#   sd_v = sd(v)
#   v_std = (v - mu_v) / sd_v
#   hist(v_std,main=paste("Distribution of Standardized",label))
#   xfit = seq(min(v_std), max(v_std), length=length(v_std))
#   yfit = dnorm(xfit, mean=mu_v, sd = sd_v)
#   lines(xfit,yfit,col="red", lwd=2)
#   qqnorm(v_std,main=paste("QQ Plot of",label))
#   abline(0,1)
#
# }
#overlayGaussian(multidf$BTCdelta,"BTC Daily Price Changes")
#overlayGaussian(multidf$SP500delta,"S&P 500 Daily Price Changes")

# Correlation of cryptocurrency prices
corData = data.frame(Currency=c("BTCETH","BTCXMR","BTCXRP","ETHXMR","ETHXRP","XMRXRP"),
                     Correlations=c(
                        cor(BTCdf$Close, ETHdf$Close),
                        cor(BTCdf$Close, XMRdf$Close),
                        cor(BTCdf$Close, XRPdf$Close),
                        cor(ETHdf$Close, XMRdf$Close),
                        cor(ETHdf$Close, XRPdf$Close),
                        cor(XMRdf$Close, XRPdf$Close)) )
kable(corData[order(-corData$Correlations),], caption="Correlations of Cryptocurrency Closing Prices",
      row.names = F,digits=2)
# Visualize correlations using a heatmap
fourCur <- data.frame(cbind(standarize(multidf$BTCdelta),
```

```
                              standarize(multidf$XMRdelta),
                              standarize(multidf$XRPdelta),
                              standarize(multidf$ETHdelta)))
colnames(fourCur) = c("BTC","XMR","XRP","ETH")
cormat<-signif(cor(fourCur),2)
col<- colorRampPalette(c("blue","white", "red"))(20)
heatmap(cormat, col=col, symm=TRUE,Rowv=NA, main="Price Change Correlations")
#Largest correlation between BTC and XMR
# Linear regression - using change in BTC prices to predict change in XMR prices
plot(multidf$BTCdelta,multidf$XMRdelta,pch = ".",cex = 3,
     main="Using Linear Regression of Price Changes \nto Predict One Currency from Another",
     xlab="Change in BTC prices",
     ylab="Change in XMR prices")
#b is slope
b <- cov(multidf$BTCdelta,multidf$XMRdelta)/var(multidf$BTCdelta)
#a is intercept
a <- mean(multidf$BTCdelta) - b*mean(multidf$BTCdelta)
#We can add this regression line to the plot of the data
abline(a, b, col = "red")

# Check using R's linear regression function
lm = lm(XMRdelta~BTCdelta,data=multidf)
summary(lm)
# Permutation Tests
#Define function so we can repeat the process easily
permTest <- function(v1, v2, label1,label2) {

  Obs <- mean(v1 - v2); Obs
  N <- 10000; diff <- numeric(N)
  for (i in 1:N) {
    scramble <- sample(v1,length(v1))
    diff[i] <- mean(scramble - v2)
  }
  pval = mean(diff > Obs)
  hist(diff,main=paste0("Permutation Test of ",label1," and ",label2,"\n(p-value:",pval,")"),
       xlab=label1, ylab=label2)
  abline(v=Obs, col = "red")
  return(pval)
}
par(mfrow=c(3,2))
permTest(standarize(multidf$BTCdelta), standarize(multidf$XMRdelta),"BTC","XMR")
permTest(standarize(multidf$BTCdelta), standarize(multidf$XRPdelta),"BTC","XRP")
permTest(standarize(multidf$BTCdelta), standarize(multidf$ETHdelta),"BTC","ETH")
permTest(standarize(multidf$ETHdelta), standarize(multidf$XMRdelta),"ETH","XMR")
permTest(standarize(multidf$ETHdelta), standarize(multidf$XRPdelta),"ETH","XRP")
permTest(standarize(multidf$XRPdelta), standarize(multidf$XMRdelta),"XRP","XMR")
par(op)
#Contingency Table
cntgTableCustom <- function(firstIndex, periods, periodlength, vixrange) {
  hasVarInc <- numeric(periods)
  VIXChange <- numeric(periods)
  for(i in 1:periods) {
    sIdx <- firstIndex + (periodlength*i)
```

```r
    #Get the vix change in the first 3 days of the period
    VIXChange[i] <- (multidf[sIdx+vixrange,]$VIXCLS - multidf[sIdx,]$VIXCLS) >= 0
    #Get the variance of the price changes for this month and last month
    thisMonthVar <- var(multidf$SP500delta[c(sIdx:(sIdx+periodlength))]);
    lastMonthVar <- var(multidf$SP500delta[c((sIdx-periodlength):(sIdx-1))]);
    hasVarInc[i] <- thisMonthVar >= lastMonthVar;
  }
  table <- table(VIXChange, hasVarInc); table
  #Used https://rstudio-pubs-static.s3.amazonaws.com/158214_3e5cc0d244f942f2a2dc33fecdf87764.html
  #For the mosaicplot example
  mosaicplot( table, col = c("firebrick", "goldenrod1"), cex.axis = 1, sub = "VIX Change", ylab = "Vari
  table
}

#Start from 2015 October 1st with 6 periods of 30-days and using the change in VIXCLS for first 3 days
periods = 6
periodlength = 30
firstIndex <- which(multidf$Date==as.Date("2015-10-01"))
cntgTableCustom(firstIndex, periods, periodlength, 3)

#Do it again with the next 6 month period
firstIndex <- firstIndex + periods*periodlength
cntgTableCustom(firstIndex, periods, periodlength, 3)

#Do it again with the next 6 month period
firstIndex <- firstIndex + periods*periodlength
cntgTableCustom(firstIndex, periods, periodlength, 3)
##
```