Math23C Spring 2018 Final Project

Patrick Watts, David Wihl

May 6, 2018

```
library(xts)
library(quantmod)
library(ggplot2)
```

Abstract

Cryptocurrencies such as Bitcoin, Ethereum, etc. generated significant attention in 2017. Cryptocurrencies have significant volalility as there is rampant speculation. Given the high variance in prices, can data science methods explored in this class be used to model the market dynamics?

Data Source

We obtained a history cryptocurrency price data from https://www.kaggle.com/sudalairajkumar/cryptocurrencypricehistory/data for four different cryptocurrencies (BTC, ETH, XMR, XRP)

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

Data Ingestion, clean up and normalization

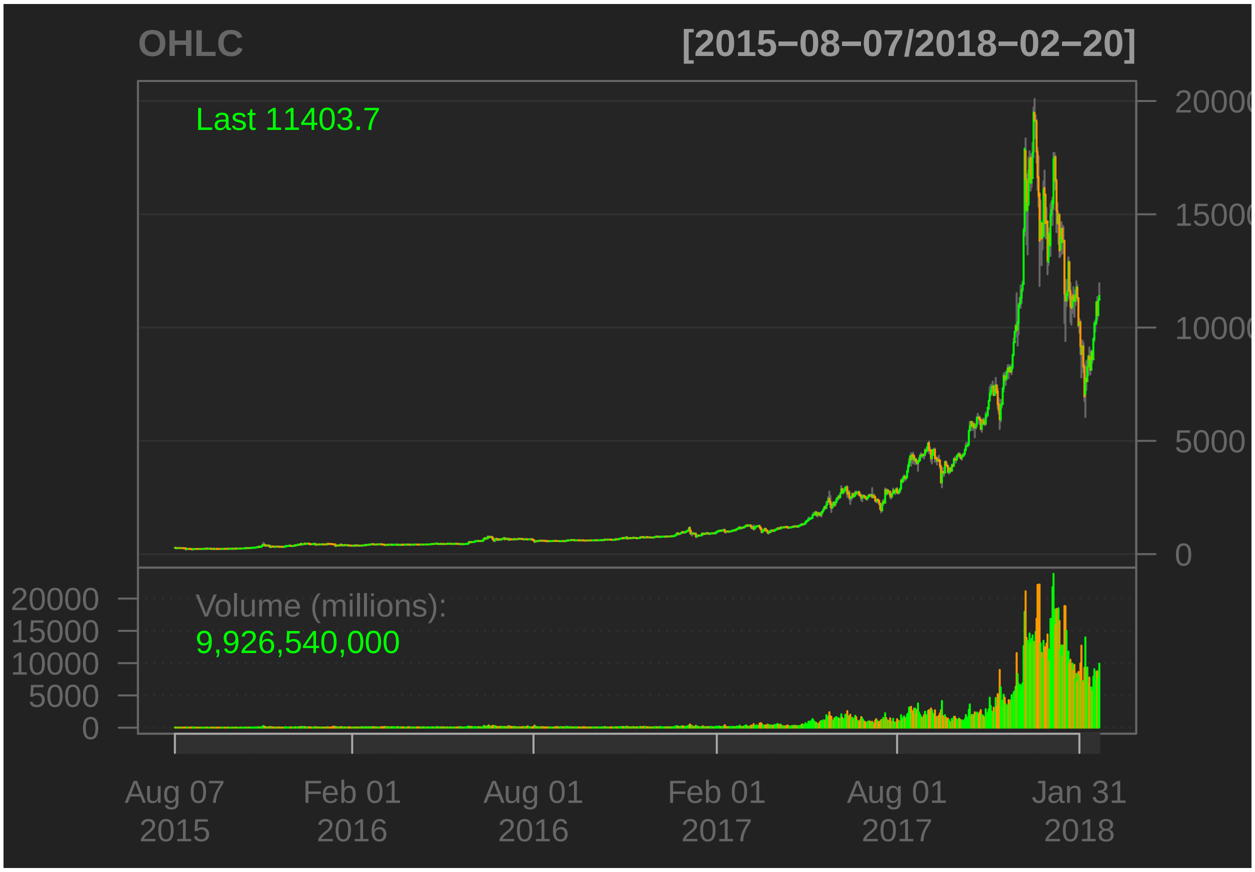
```
# Read one price history file per currency
BTCdf = read.csv("data/bitcoin_price.csv", stringsAsFactors = F)
ETHdf = read.csv("data/ethereum_price.csv", stringsAsFactors = F)
XMRdf = read.csv("data/monero_price.csv", stringsAsFactors = F)
XRPdf = read.csv("data/ripple_price.csv", stringsAsFactors = 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,]
```

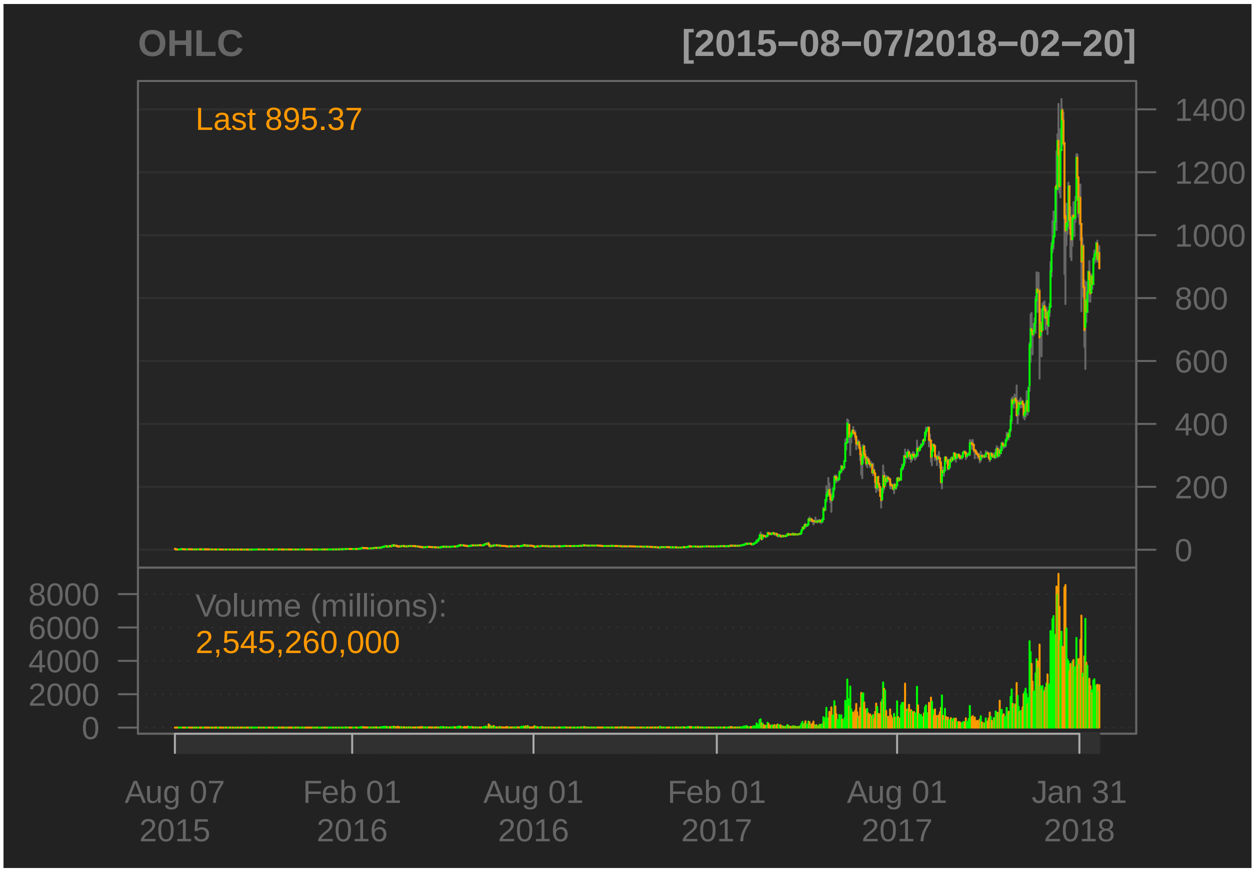
```
# Read in traditional, noncryptocurrency data
noncrypto = read.csv("data/noncrypto.csv",stringsAsFactors = FALSE)
noncrypto$DATE = as.Date(noncrypto$DATE,"%Y-%m-%d")
```

Exploratory Data Analysis

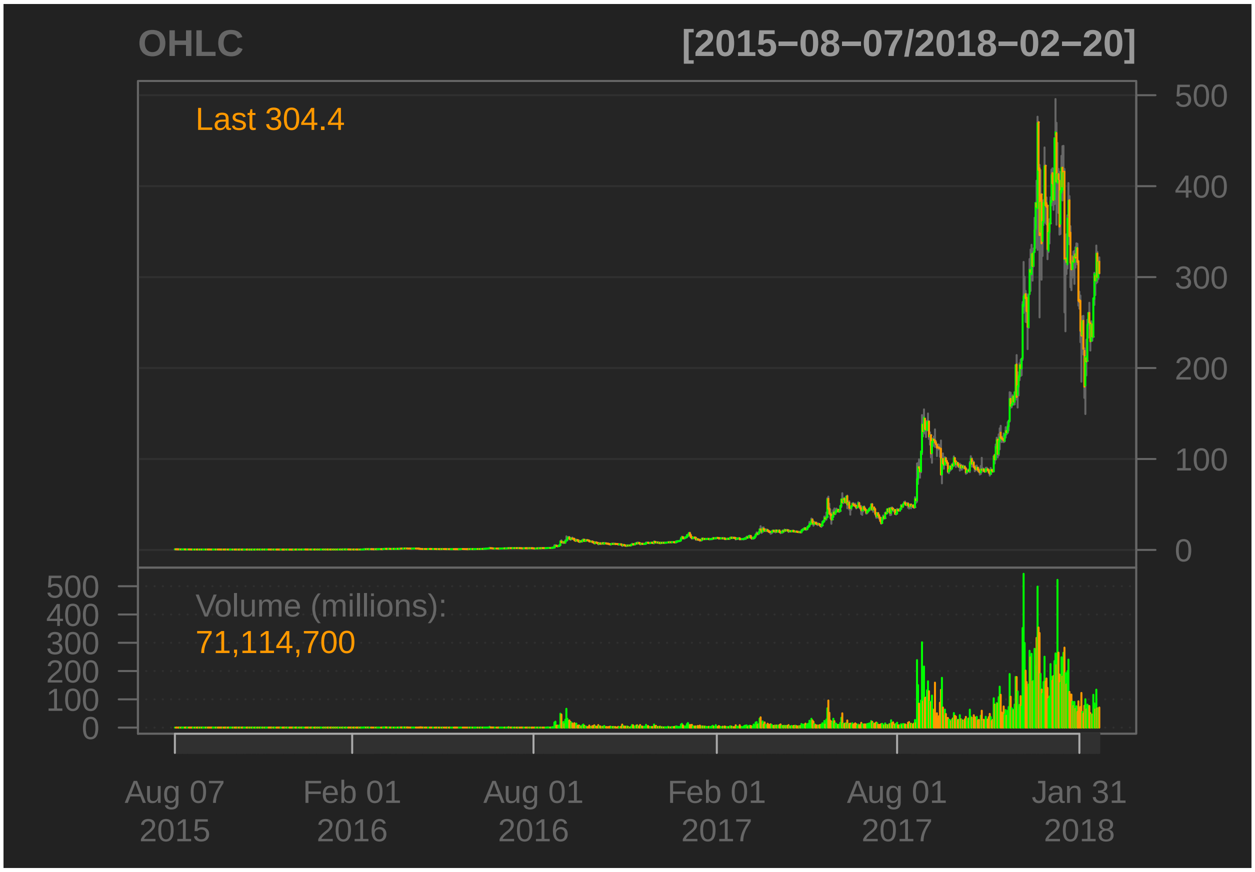
```
plotSeries = function(df){
  dfdata = xts(df[,2:7],order.by = df[,1])
  # TODO fix weekly plot
  #wk = dfdata
  #data.wk = to.weekly(wk)
  #plot(data.wk)
  #plot.new()
  OHLC = as.quantmod.OHLC(dfdata)
  chartSeries(OHLC)
}
plotSeries(BTCdf)
```



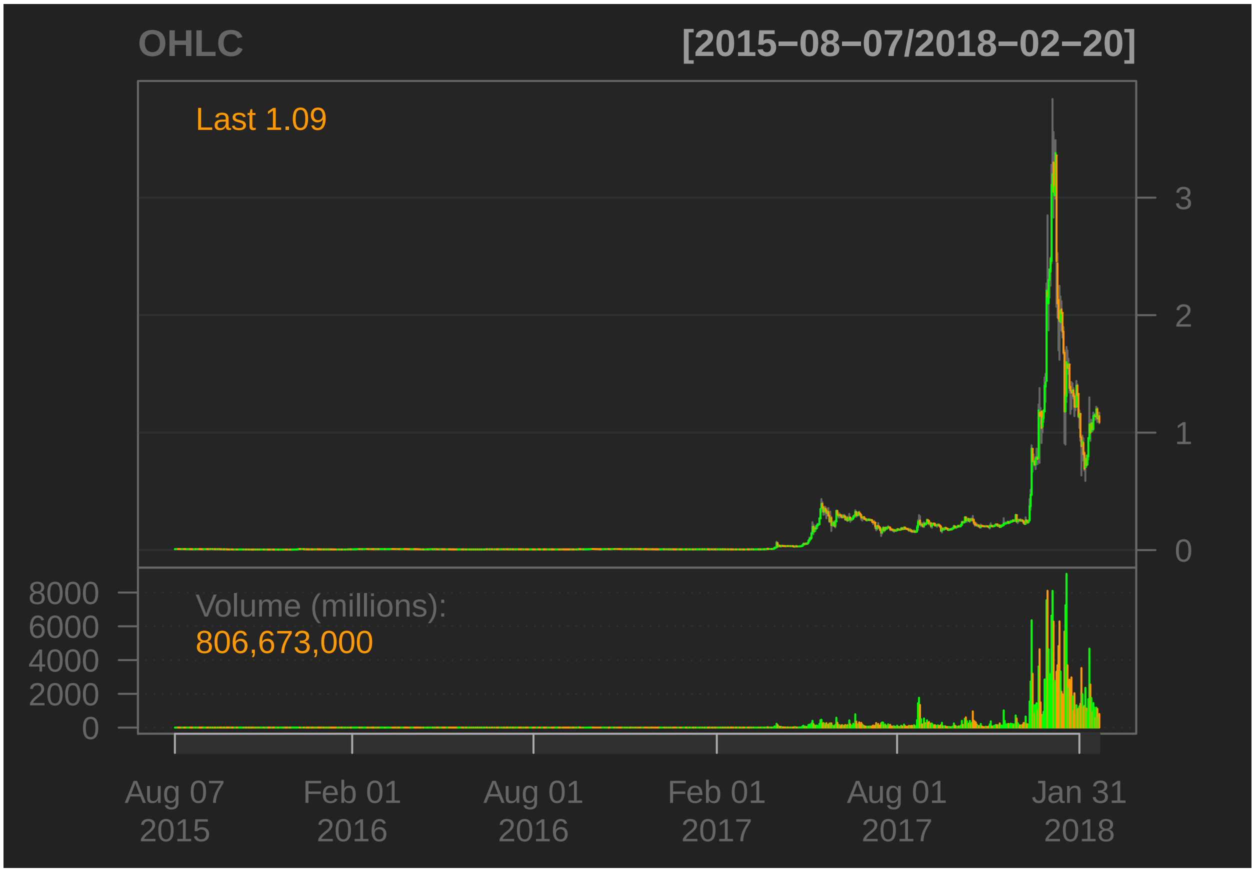
```
plotSeries(ETHdf)
```



```
plotSeries(XMRdf)
```



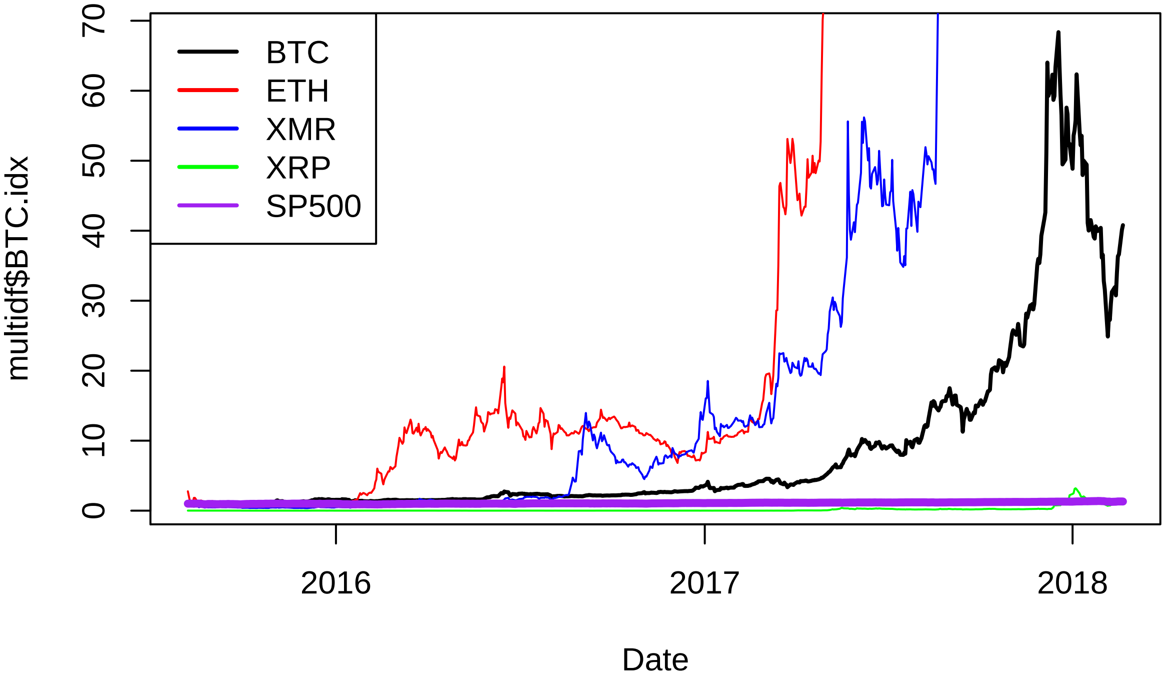
```r
plotSeries(XRPdf)
```



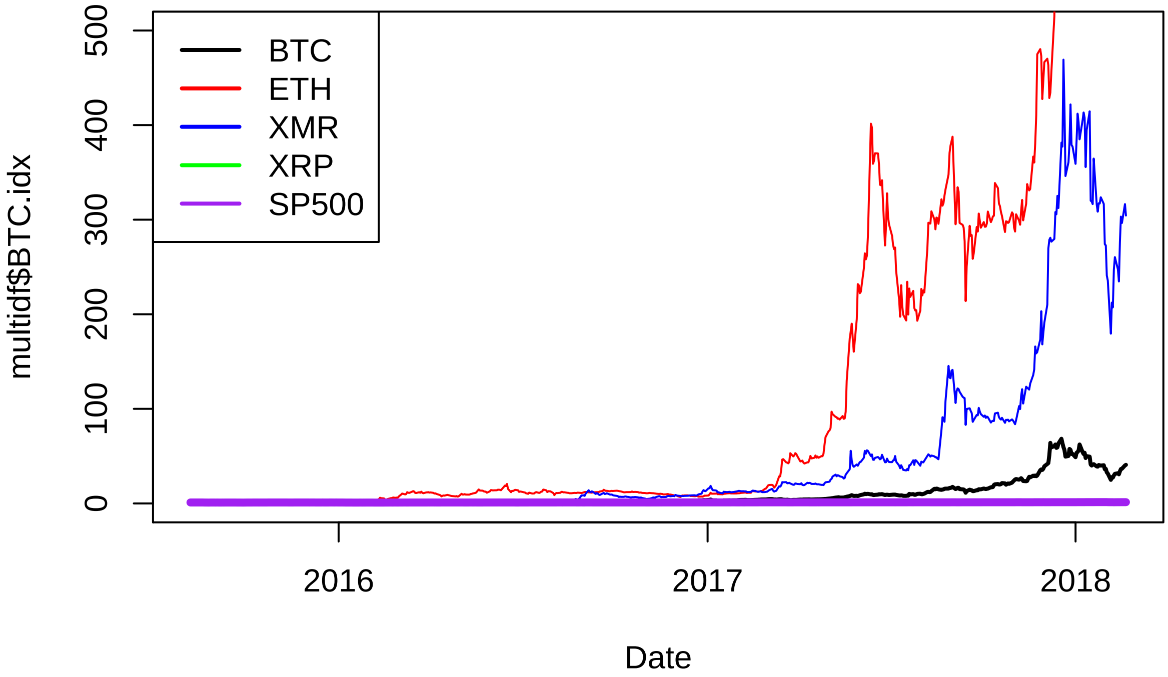
```
BTCdata = xts(BTCdf[,2:7],order.by = BTCdf[,1])
# wk = BTCdata
# data.wk = to.weekly(wk)
# plot(data.wk)
# OHLC = as.quantmod.OHLC(BTCdata)
#chartSeries(OHLC)
```

```
close.prices = BTCdata$Close
close.prices = cbind(close.prices,ETHdf$Close,XMRdf$Close,XRPdf$Close)

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")

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]
```

```
# default y scale
plot(x = multidf$Date,y=multidf$BTC.idx,type="l",xlab="Date",col="black",lty=1,lwd=2)
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))
```

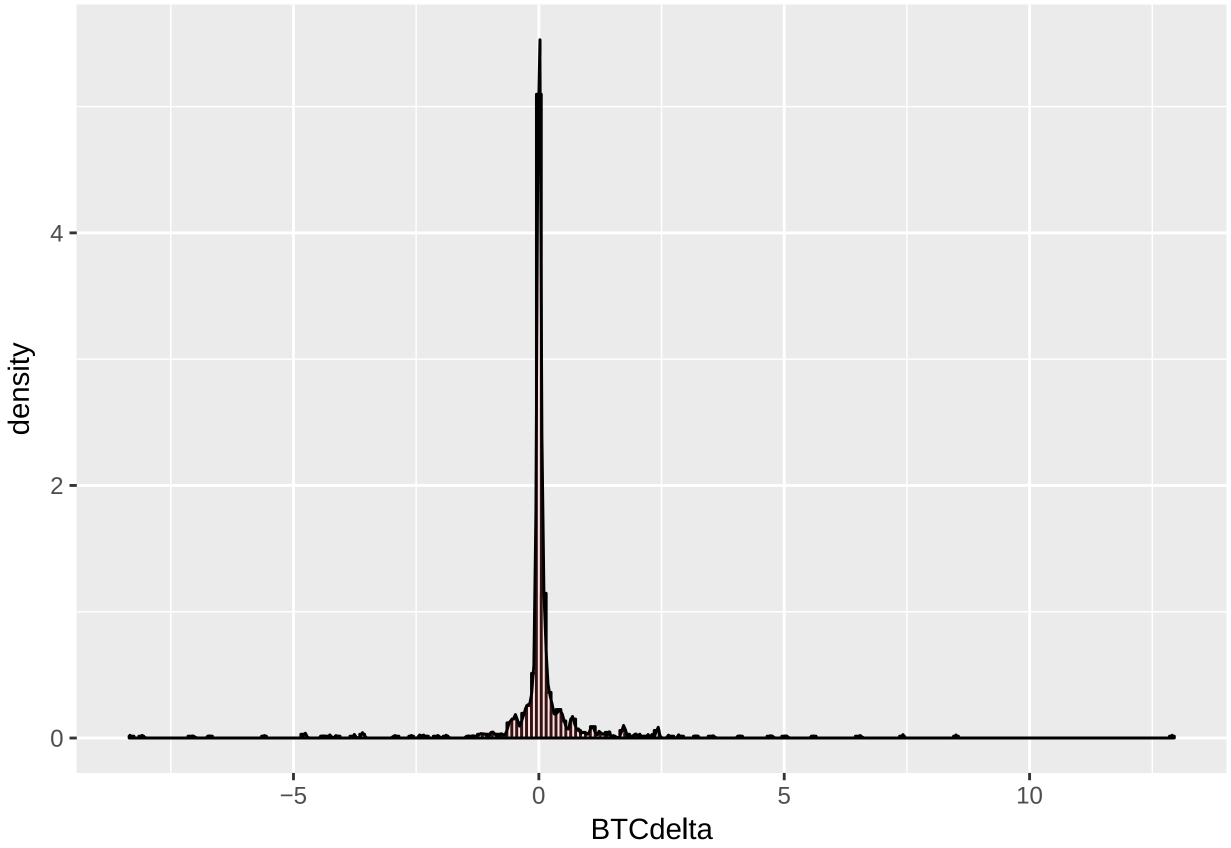


```
# y scale from 0-500
plot(x = multidf$Date,y=multidf$BTC.idx,type="l",xlab="Date",col="black",lty=1,lwd=2,ylim=c(0,500))
lines(x=multidf$Date,y=multidf$ETH,col="red")
lines(x=multidf$Date,y=multidf$XMR,col="blue")
lines(x=multidf$Date,y=multidf$XRP,col="green")
lines(x=multidf$Date,y=multidf$SP500.idx,col="purple",lwd=4)
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))
```

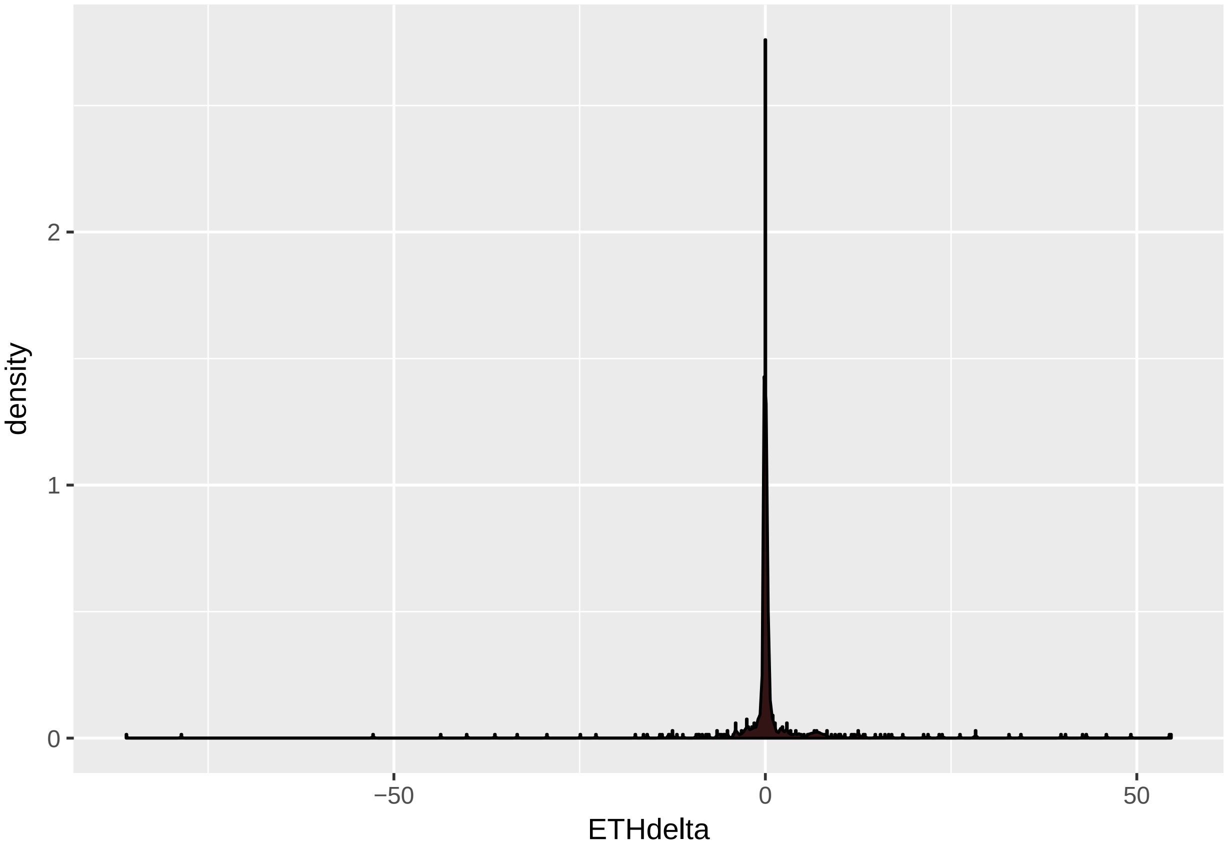


```
# distribution of relative price changes
multidf$BTCdelta = c(0,diff(multidf$BTC.idx))
multidf$ETHdelta = c(0,diff(multidf$ETH.idx))
multidf$XMRdelta = c(0,diff(multidf$XMR.idx))
multidf$XRPdelta = c(0,diff(multidf$XRP.idx))
multidf$SP500delta = c(0,diff(multidf$SP500.idx))
```

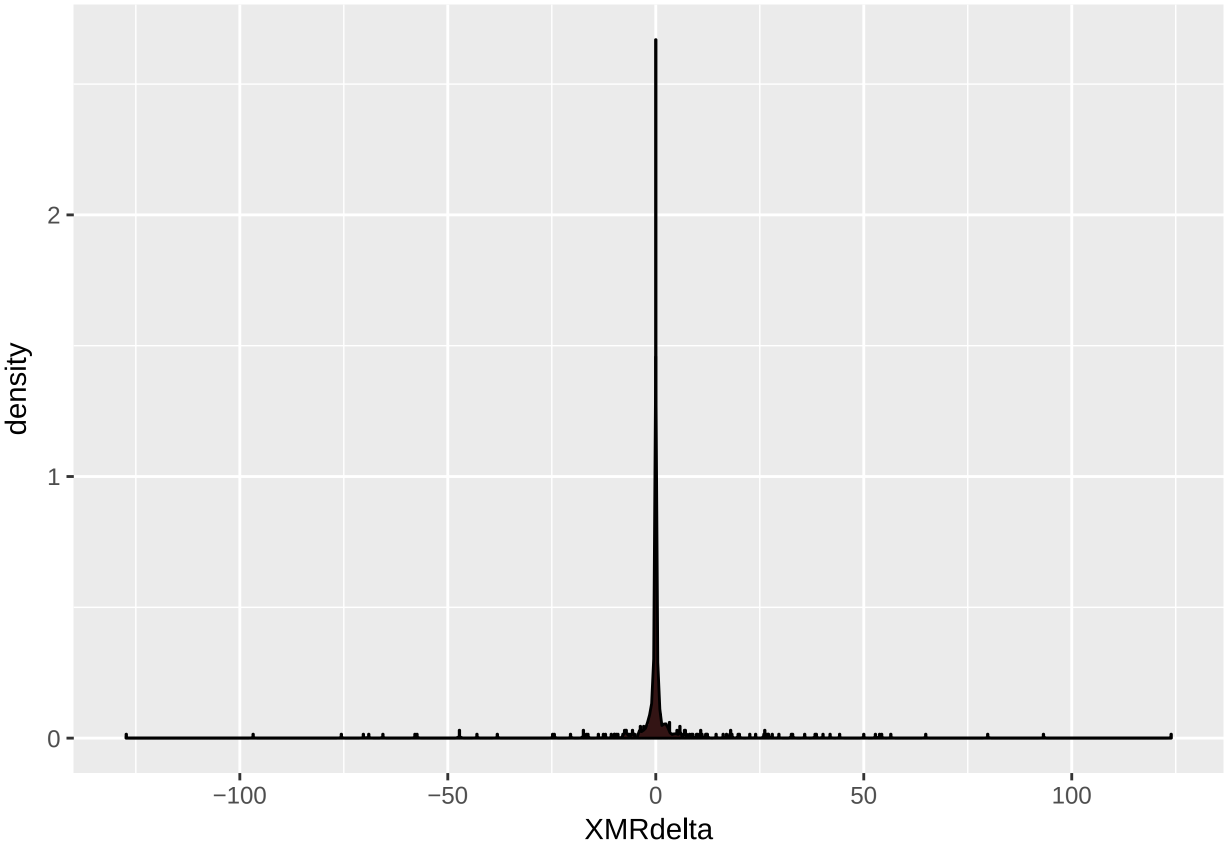
```
# REQ: display a histogram
ggplot(multidf, aes(x=BTCdelta)) +
  geom_histogram(aes(y=..density..),
                 binwidth=0.1,colour="black",
                 fill="white") +
  geom_density(alpha=0.2,fill="#FF6666")
```



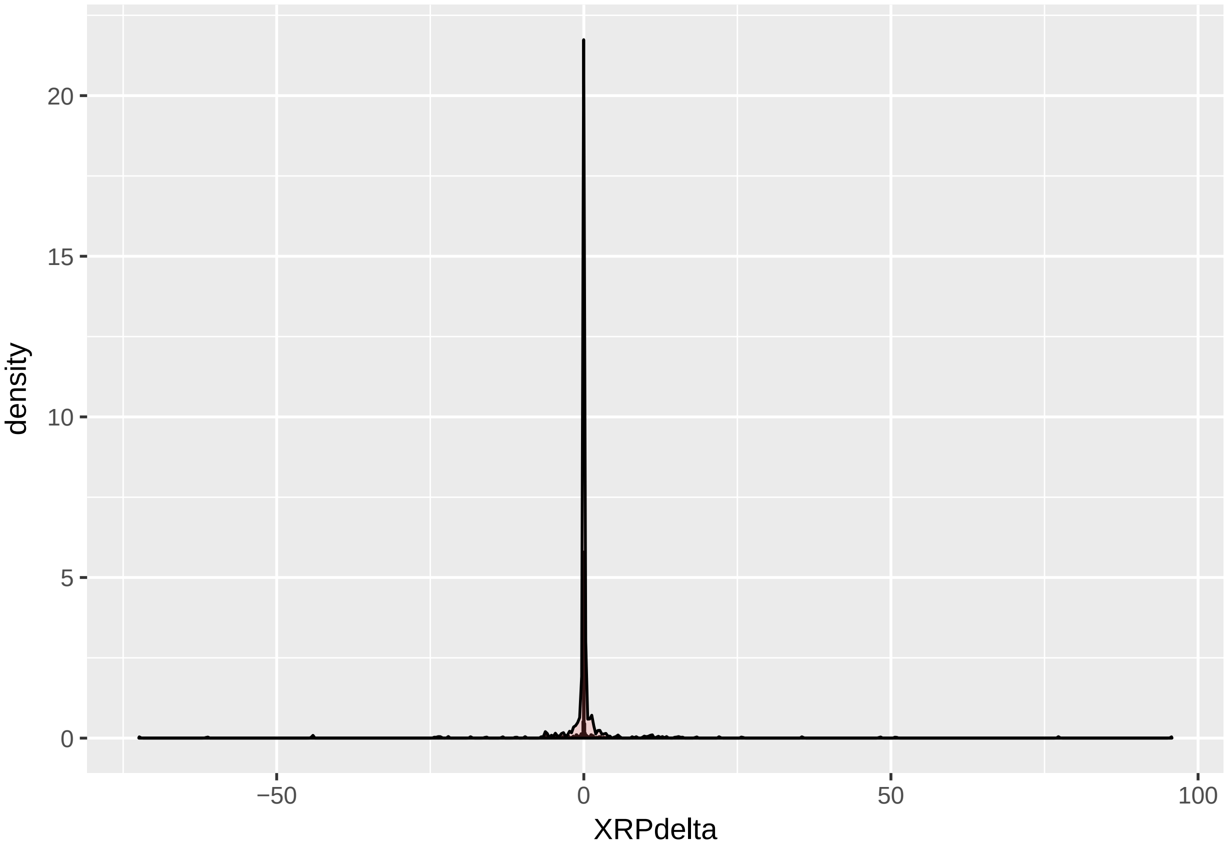
```
ggplot(multidf, aes(x=ETHdelta)) +
  geom_histogram(aes(y=..density..),
                 binwidth=0.1,colour="black",
                 fill="white") +
  geom_density(alpha=0.2,fill="#FF6666")
```



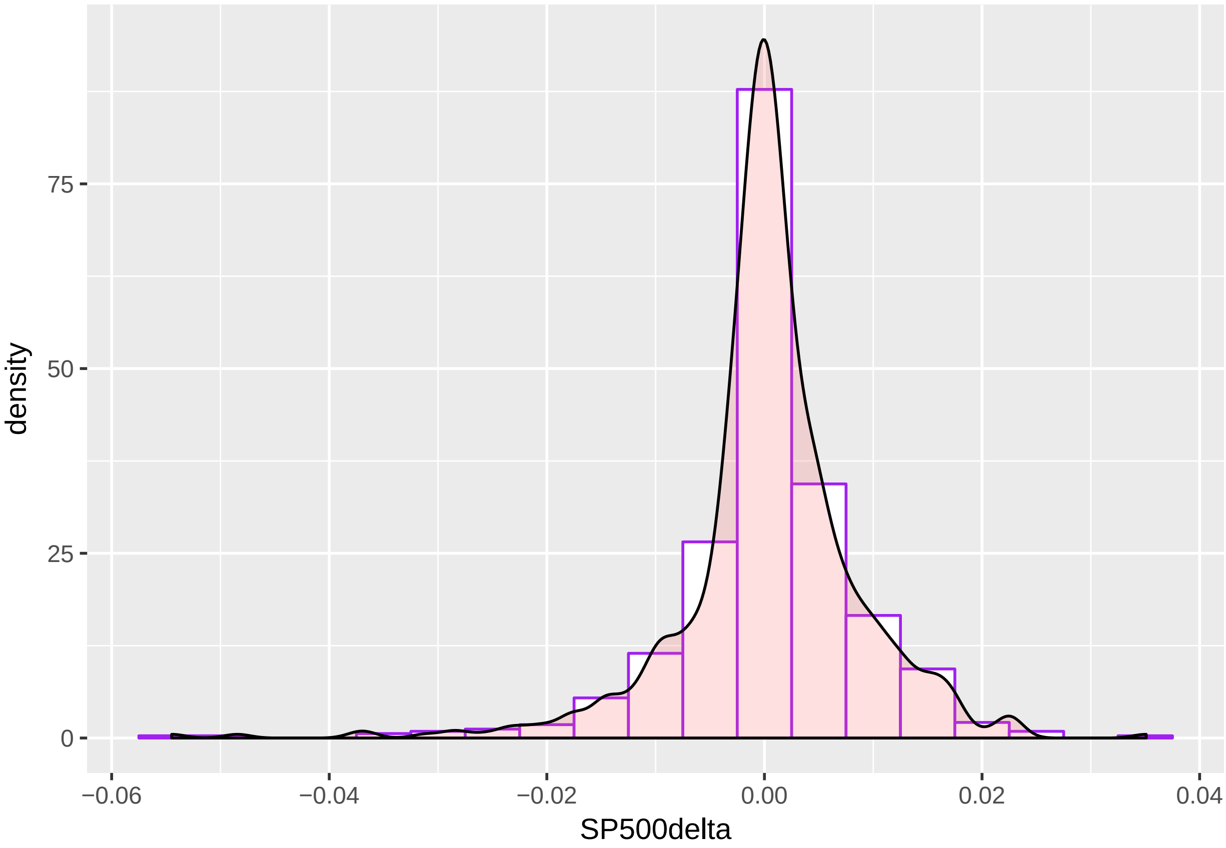
```
ggplot(multidf, aes(x=XMRdelta)) +
  geom_histogram(aes(y=..density..),
                 binwidth=0.1,colour="black",
                 fill="white") +
  geom_density(alpha=0.2,fill="#FF6666")
```



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



```
ggplot(multidf, aes(x=SP500delta)) +
  geom_histogram(aes(y=..density..),
                 binwidth=0.005,colour="purple",
                 fill="white") +
  geom_density(alpha=0.2,fill="#FF6666")
```

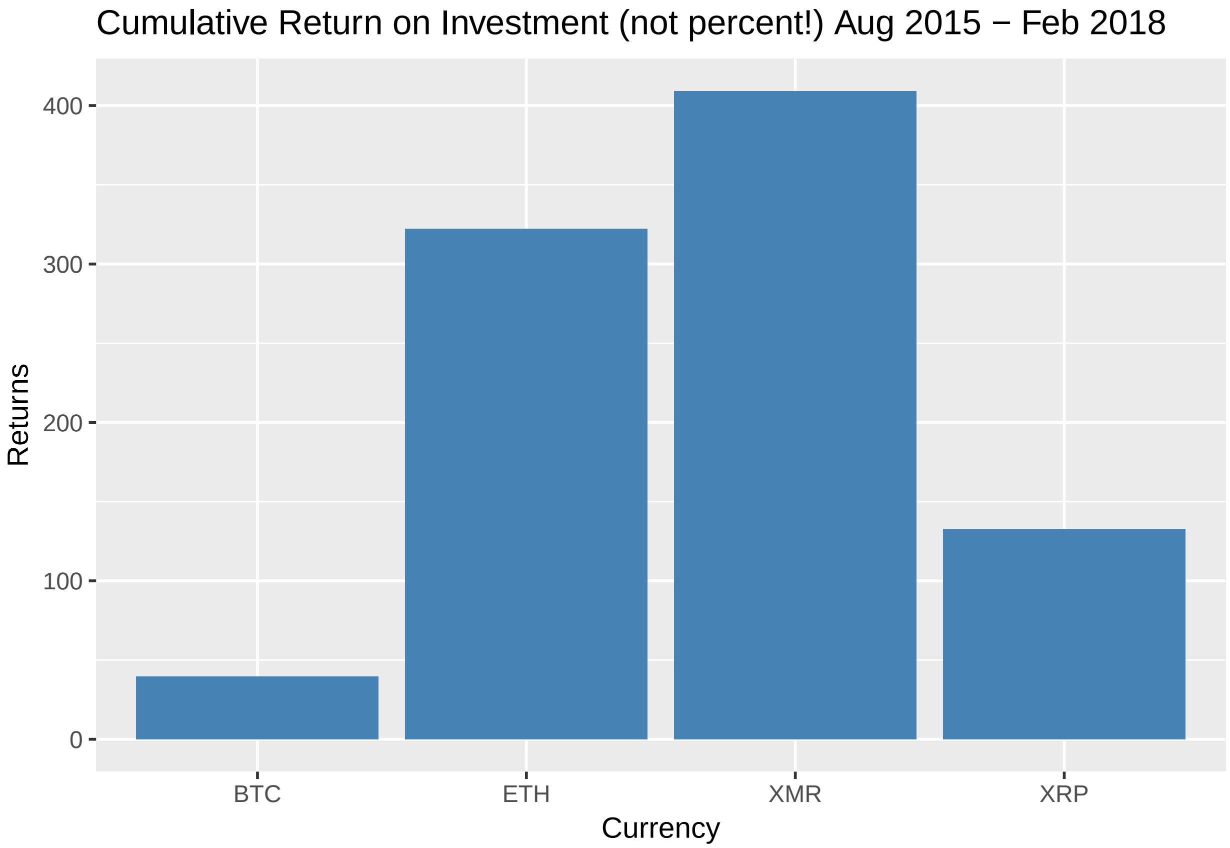


```
# TODO: overlay S&P500 over any cryptocurrency

# TODO: normalize values and overlay Gaussian
```

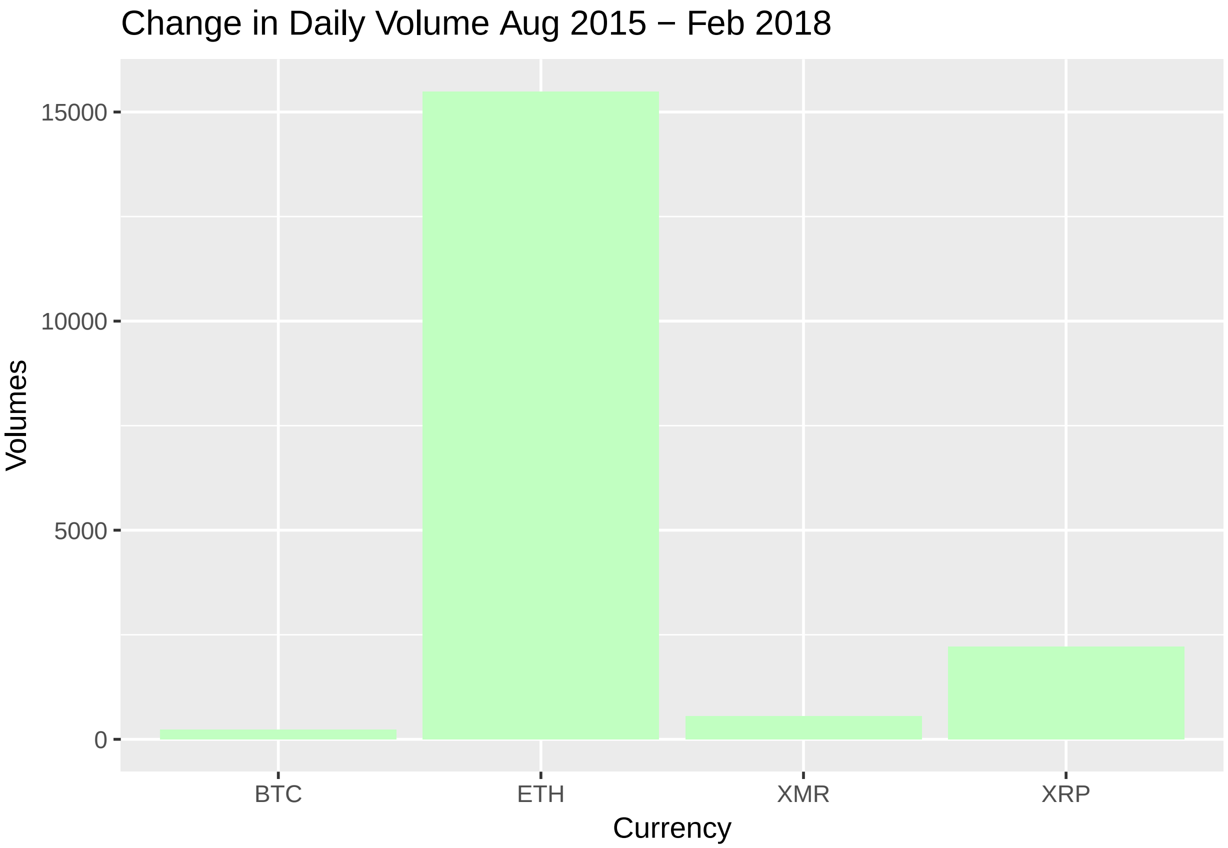
```
# Display bar plots showing Overall Return and Change in Daily Volume
overallReturn = function(df){
  return ((df$Close[nrow(df)] - df$Close[1]) / df$Close[1])
}

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)
barData
```

```
##   Currency   Returns    Volumes
## 1      BTC  39.78868   232.6492
## 2      ETH 322.23827 15487.8060
## 3      XMR 409.08121   551.9270
## 4      XRP 132.70952  2217.3103
```

```
# 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")
```

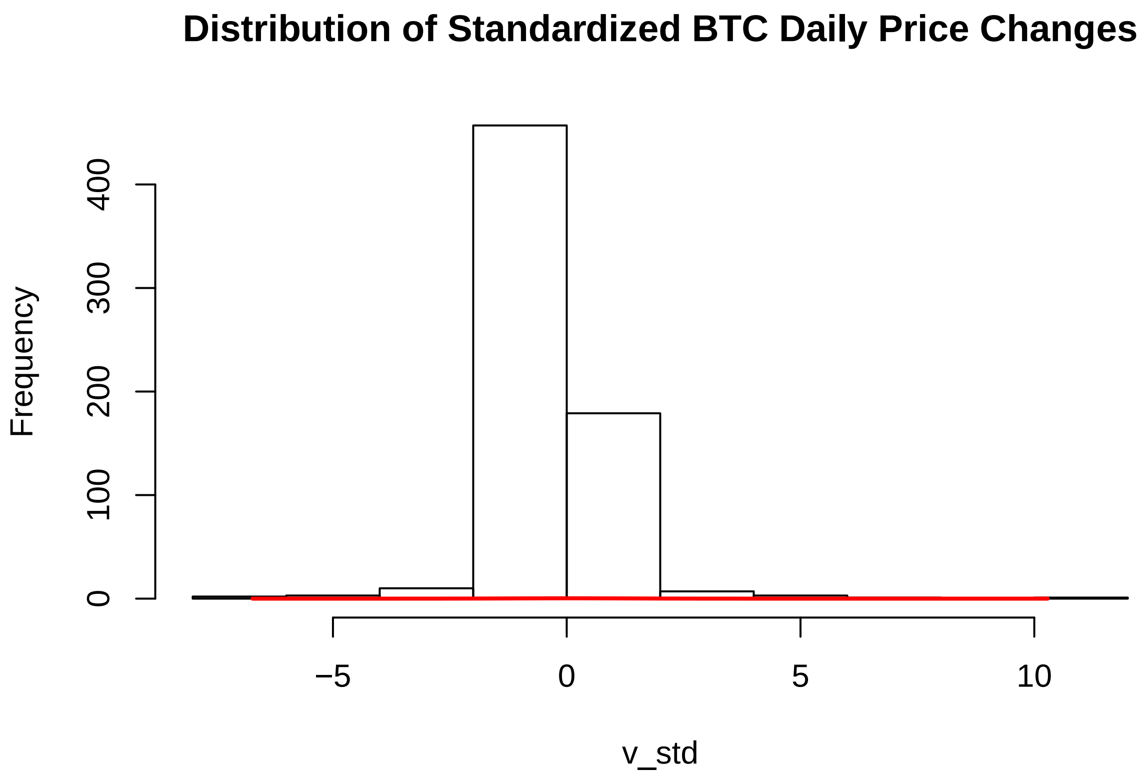


```
ggplot(data=barData, aes(x=Currency, y=Volumes)) +
  geom_bar(stat="identity",fill="darkseagreen1") +
  ggtitle("Change in Daily Volume Aug 2015 - Feb 2018")
```

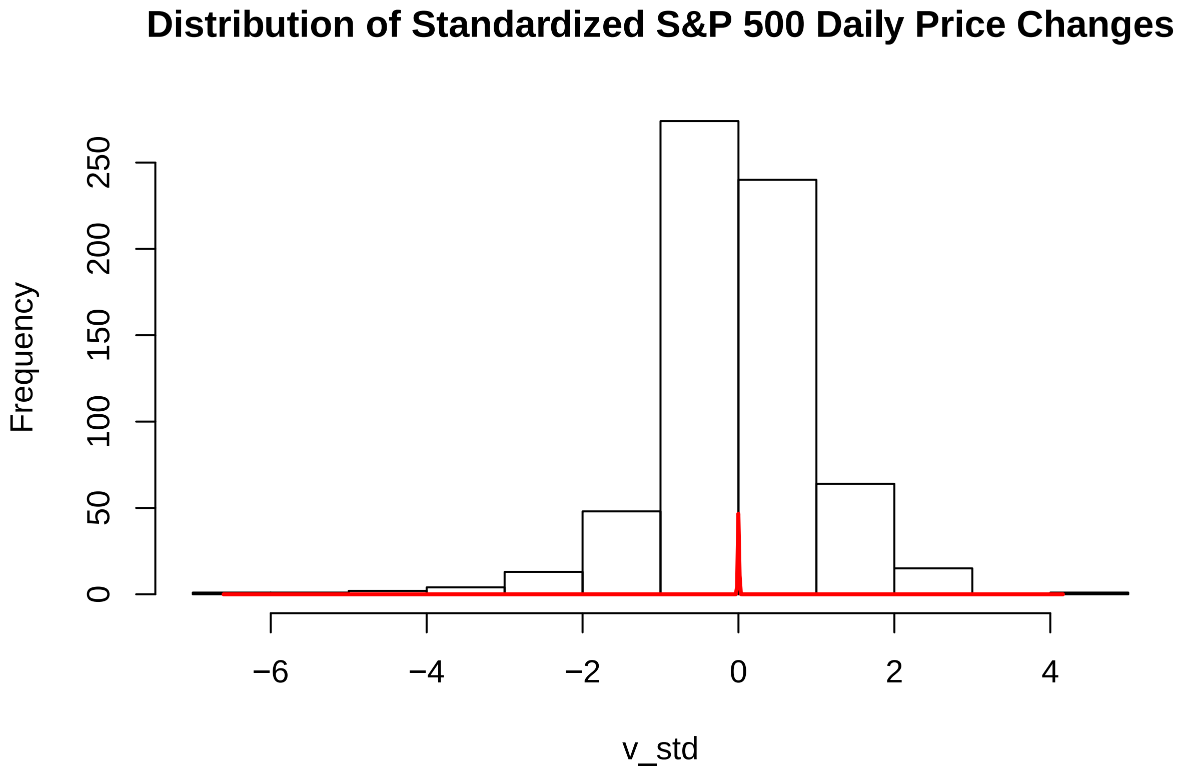


```
# Categorical Variables

# Converting VIX into a categorical
#TODO

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)
```

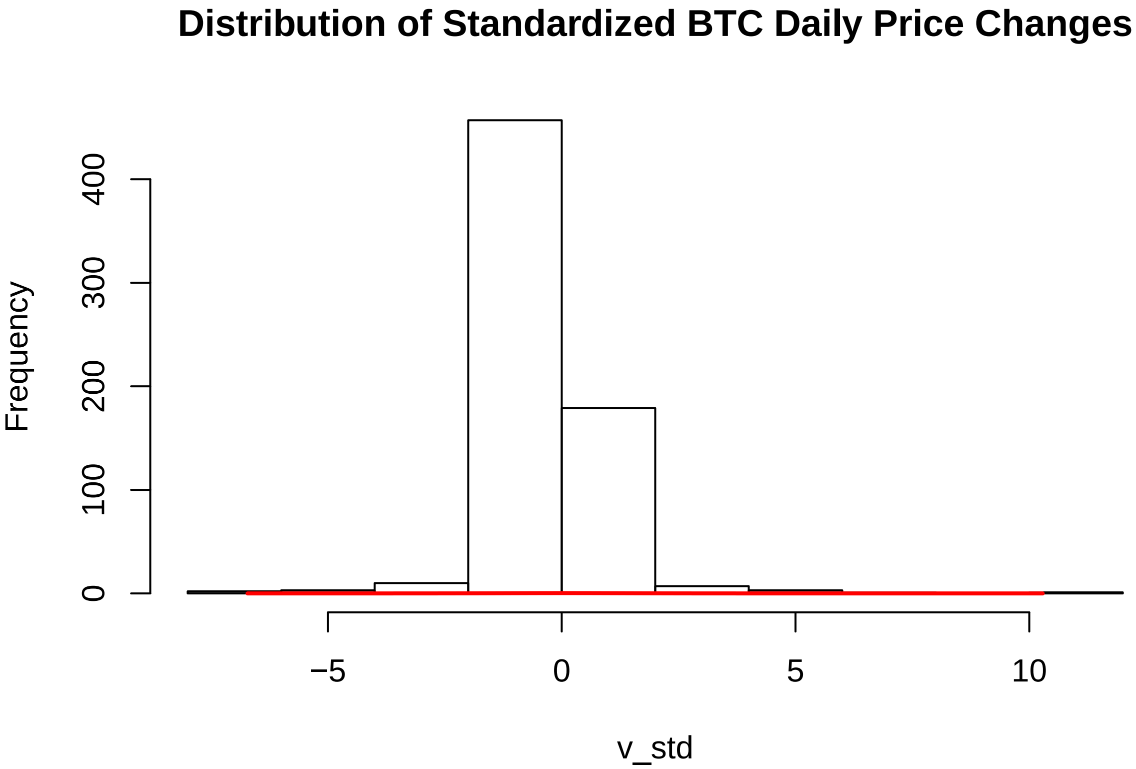
```
# TODO how many standard deviations are exceeded

overlayGaussian = function(v,label){
  mu_v = mean(v)
  sd_v = sd(v)
  v_std = (v - mu_v) / sd_v
  hist(v_std,main=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)
}
overlayGaussian(multidf$BTCdelta,"Distribution of Standardized BTC Daily Price Changes")
```



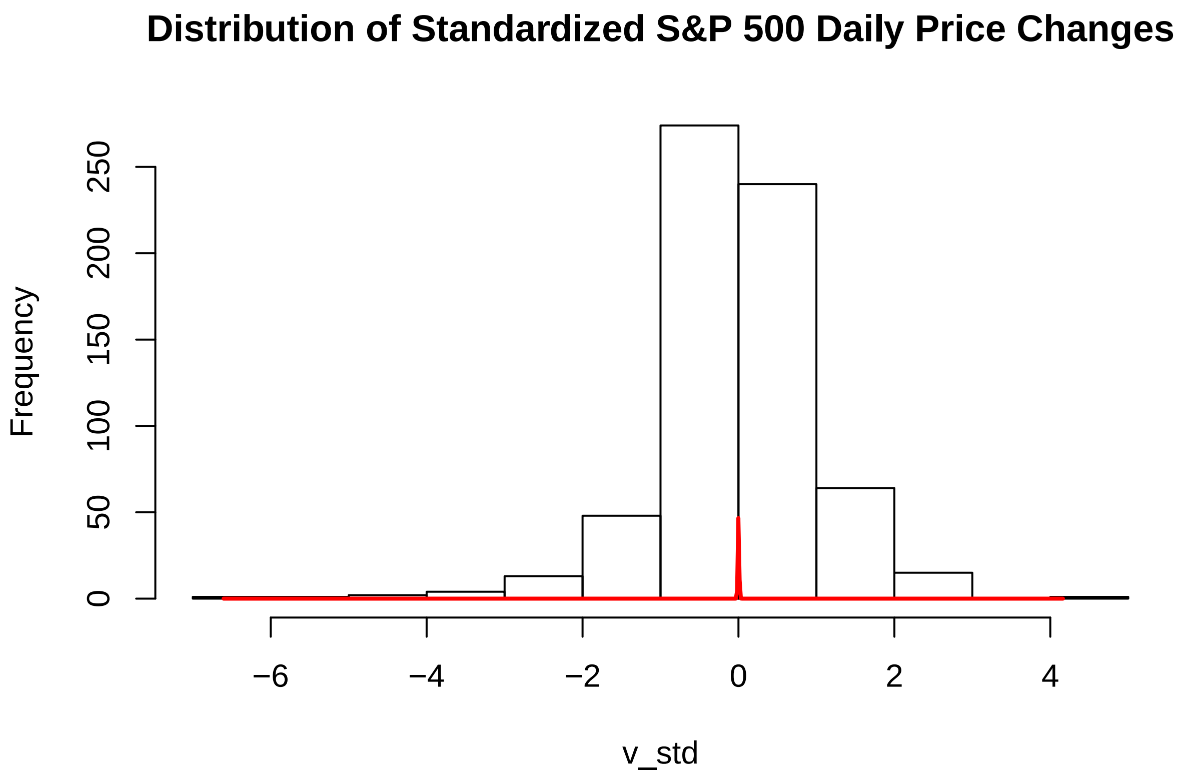
```
overlayGaussian(multidf$SP500delta,"Distribution of Standardized S&P 500 Daily Price Changes")
```



```
# Overlay Beta
overlayBeta = function(v,label){
  # this will rescale vector v to [0,1]
  v_norm = (v - min(v)) / (max(v) - min(v))
  hist(v_norm,main=label)
  xfit = seq(min(v), max(v), length=length(v))
  yfit = dnorm(xfit, mean=mu_v, sd = sd_v)
  lines(xfit,yfit,col="red", lwd=2)
}
overlayGaussian(multidf$BTCdelta,"Distribution of Standardized BTC Daily Price Changes")
```



```
overlayGaussian(multidf$SP500delta,"Distribution of Standardized S&P 500 Daily Price Changes")
```



##

Topic 1 - Volalitility Exploration

Topic 2 -

Topic 3 - Correlation between Cryptocurrencies

```
cor(BTCdf$Close, ETHdf$Close)
```

```
## [1] 0.9060949
```

```
cor(BTCdf$Close, XMRdf$Close)
```

```
## [1] 0.9691732
```

```
cor(BTCdf$Close, XRPdf$Close)
```

```
## [1] 0.8049156
```

```
cor(ETHdf$Close, XMRdf$Close)
```

```
## [1] 0.9525516
```

```
cor(ETHdf$Close, XRPdf$Close)
```

```
## [1] 0.8798746
```

```
cor(XMRdf$Close, XRPdf$Close)
```

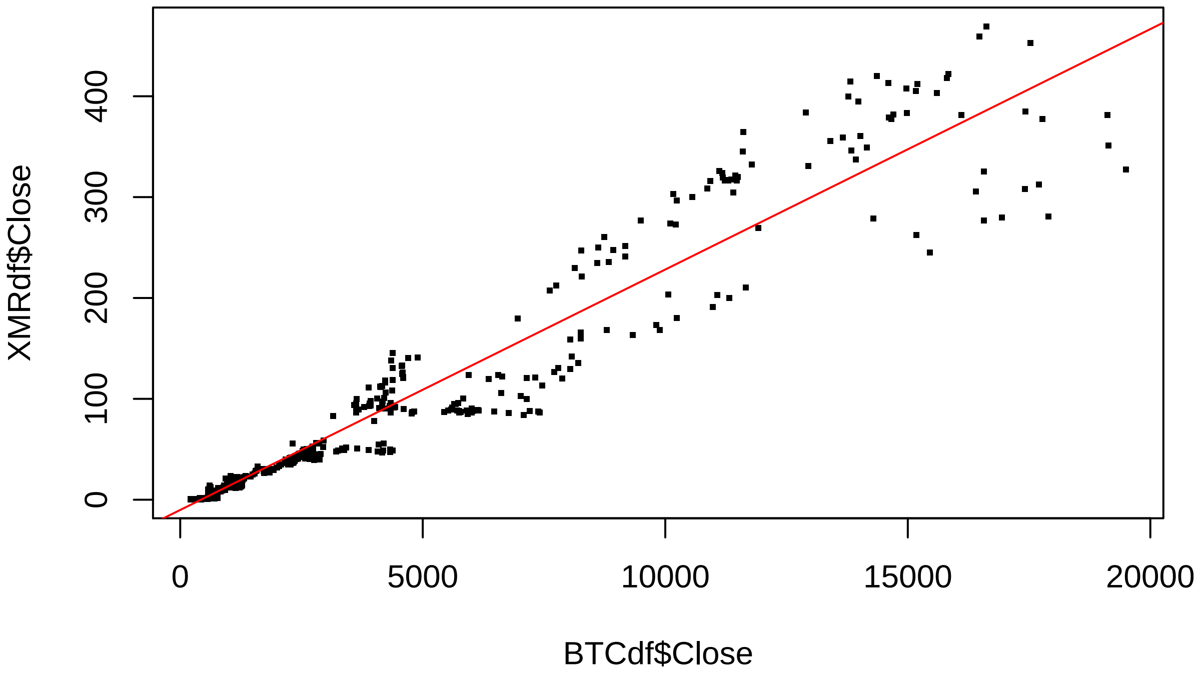
```
## [1] 0.8847865
```

```
#Largest correlation between BTC and XMR
plot(BTCdf$Close,XMRdf$Close,pch = ".",cex = 3)
#b is slope
```

```
b <- cov(BTCdf$Close,XMRdf$Close)/var(BTCdf$Close)
#a is intercept
a <- mean(XMRdf$Close) - b*mean(BTCdf$Close);a
```

```
## [1] -10.16858
```

```
#We can add this regression line to the plot of the data
abline(a, b, col = "red")
```



##
Summary of Project Requirements

Required dataset standards

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

Required Analysis

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

Required submission uploads

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

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

References

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

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