Forecasting Crypto Data: Project Report

Roger Garcia
Data Science with R., Spring 2019
Dr. J. Edward Swan II
Mississippi State University

Fri Apr 26 2019

Contents

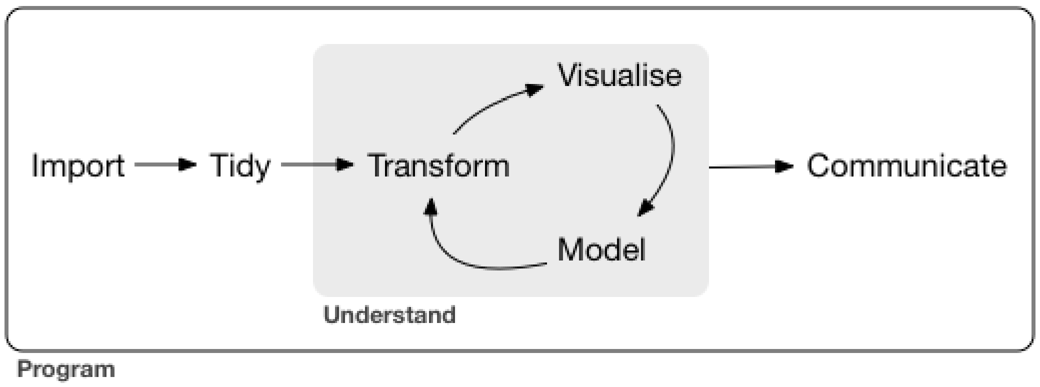


Figure 1: Data Science Workflow

1 Introduction

1.1 Course description

The goal of this report is to share my results as part of my coursework in *Science Data with R* during Spring 2019. The course (Professor: Dr. J. Edward Swan II) focuses on introducing the fundamental elements of data science: visual analysis, data manipulation and modeling. The objective of this semester project was to find a dataset and go through all of the stages of the *Data Science Workflow* (see Figure 1 or R for Data Science), present findings to classmates and submit a presentation and written report on findings.

1.2 Motivation and background

Investing, markets, financial institutions and trading are not new concepts, but in fact have been around for decades and while investing was only something the very wealthy were allowed to do, now a days individuals have a variety of options to invest.

Over the past several years, further attention has been given to the use of blockchain technology in education, finance, healthcare and for social good. Despite, the fundamental concepts surrounding this technology not being new, its ever-expanding interest can be trace back to the popularity of Bitcoin (see Nakamoto 2008).

Having invested in Bitcoin in *July 29, 2017* (*2017-07-29*) for the first time and along with a favorite John Tukey quote of mine, “The greatest value of a picture is when it forces us to notice what we never expected to see”, I wanted to force myself to self-reflect upon the brutal truth of my own investing/trading decisions.

For this reason, during this project I’ve analyze crypto market data in order to build discussion on what’s occur in the *crypto* space, in particularly with Bitcoin and top ten other cryptocurrencies or alts. Endeavored to improve the quality of my own investing decisions I apply techniques used in class to build a higher level abstraction from such a dataset in the form of simple stories or visualizations and let this project be meaningful to me (and hopefully valuable to others) beyond this course.

2 Importing Data

Note: *Along with this work, I have provided* ***cryptodata.RData*** *that contains saved* ***R*** *objects that can be used to reproduce results, in this case there is no need to create an account, unless you want up to date market data*

It is possible to retrieve current and historical information on cryptocurrency prices with the use of `crypto` package by Jesse Vent (see Reference Manual: crypto.pdf).

2.1 Load required packages

```
knitr::opts_chunk$set(echo=TRUE, cache=TRUE, fig.asp=0.65, fig.width = 6.0)

require("crypto")
```

```
## Loading required package: crypto

## Loading required package: rvest

## Loading required package: xml2
```

```
require("tidyverse")
```

```
## Loading required package: tidyverse

## -- Attaching packages --------------------------------------------------------------- tidyverse 1.2.

## v ggplot2 3.1.0     v purrr   0.2.5
## v tibble  2.0.0     v dplyr   0.7.8
## v tidyr   0.8.2     v stringr 1.3.1
## v readr   1.3.1     v forcats 0.3.0

## -- Conflicts ------------------------------------------------------------------ tidyverse_conflicts(
## x dplyr::filter()         masks stats::filter()
## x readr::guess_encoding() masks rvest::guess_encoding()
## x dplyr::lag()            masks stats::lag()
## x purrr::pluck()          masks rvest::pluck()
```

```
require("lubridate") # makes it easier to work with dates and times
```

```
## Loading required package: lubridate

##
## Attaching package: 'lubridate'

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

```
require("tseries")   # Dickey-Fuller Test
```

```
## Loading required package: tseries
```

```
require("forecast")
```

```
## Loading required package: forecast
```

2.2 Get crypto market data

There are few things to do before getting started in order to retrieve data (if you haven't already):

- `CoinMarketCap API`: create an account, decide on pricing plan (*basic personal use suffice for project*) and get free api key.
- `api key`: You will be prompt to paste your api key:

```
# CoinMarketCap Professional API Call
cmc_api("https://pro-api.coinmarketcap.com/v1/cryptocurrency/listings/latest")
```

- `Get historic market tables`:

```
# get historical market data
# (last request: Mon Aprl 15 2019 5pm)
# (convert to type "tbl_df")

#Bitcoin
btc_hist_prices <- as_tibble(crypto_history(coin = "btc"))

#Top-ten ranked alts
topten_coins <- as_tibble(crypto_history(limit = 10))
```

To avoid making multiple calls to `api`, it is useful to save out our dataset using `save.image()` to then load it for future uses.

```
load("cryptodata.RData")
# btc_hist_prices <- btc_hist_prices
# topten_coins <- topten_coins
```

2.3 Tibble printing

This block of code will run and output the *last* 5 rows of our datasets, OHLC (Opening, High, Low and Closing) market data along with column names of datasets.

```
#Bitcoin (BTC)
tail(btc_hist_prices, n = 5)
```

```
## # A tibble: 5 x 13
##   slug  symbol name  date       ranknow  open  high   low close  volume
##   <chr> <chr>  <chr> <date>       <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 bitc~ BTC    Bitc~ 2019-04-10       1 5204. 5422. 5193. 5325. 1.55e10
## 2 bitc~ BTC    Bitc~ 2019-04-11       1 5325. 5354. 5017. 5064. 1.66e10
## 3 bitc~ BTC    Bitc~ 2019-04-12       1 5061. 5103. 4956. 5090. 1.37e10
## 4 bitc~ BTC    Bitc~ 2019-04-13       1 5089. 5127. 5062. 5097. 1.08e10
## 5 bitc~ BTC    Bitc~ 2019-04-14       1 5096. 5184. 5054. 5168. 1.04e10
## # ... with 3 more variables: market <dbl>, close_ratio <dbl>, spread <dbl>
```

```
colnames(btc_hist_prices)
```

```
##  [1] "slug"        "symbol"      "name"        "date"        "ranknow"
##  [6] "open"        "high"        "low"         "close"       "volume"
## [11] "market"      "close_ratio" "spread"
```

```
#Top-ten ranked cryptocurrencies
tail(topten_coins, n = 5)
```

```
## # A tibble: 5 x 13
##   slug  symbol name  date       ranknow   open   high    low  close volume
##   <chr> <chr>  <chr> <date>       <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 card~ ADA    Card~ 2019-04-10      10 0.0838 0.0925 0.0835 0.0900 1.31e8
## 2 card~ ADA    Card~ 2019-04-11      10 0.0899 0.0907 0.0776 0.0834 1.64e8
## 3 card~ ADA    Card~ 2019-04-12      10 0.0836 0.0854 0.0804 0.0835 1.10e8
## 4 card~ ADA    Card~ 2019-04-13      10 0.0835 0.0855 0.0829 0.0837 7.18e7
## 5 card~ ADA    Card~ 2019-04-14      10 0.0837 0.0855 0.0814 0.0846 8.25e7
## # ... with 3 more variables: market <dbl>, close_ratio <dbl>, spread <dbl>
```

```
  colnames(topten_coins)
```

```
##  [1] "slug"        "symbol"      "name"        "date"        "ranknow"
##  [6] "open"        "high"        "low"         "close"       "volume"
## [11] "market"      "close_ratio" "spread"
```

3 Data Exploration and Data Cleaning

One may be interested in exploring the correlation between *alts* and *BTC*, for this reason it may be useful to add a new variable `BTC_value` that will contain the ratio of *ALTS/BTC*. For purposes of this project, I only focus on exploring *closing prices* of our market data, since it tends to be the end of trading day which is used to make decisions for the following day (in traditional markets).

Note: *Crypto markets are open 24/7.*

3.1 Dataset options

I will create a new dataset named `topten_coins_meta` containing the additional variable `BTC_value_close` and will make sure we are using an object class `Date` representing calendar dates for our initial provided variable `date`:

```
btc_close_dates <- topten_coins %>%
  filter(symbol == "BTC") %>%
  transmute(date = as.Date(date, format = "%Y-%m-%d"),
            BTC_close = close)

topten_coins_meta <- topten_coins %>%
  left_join(btc_close_dates, by = "date") %>%
  mutate(BTC_value_close = round(close / BTC_close, digits = 8))

head(topten_coins_meta)
```

```
## # A tibble: 6 x 15
##   slug  symbol name  date       ranknow  open  high   low close volume
##   <chr> <chr>  <chr> <date>       <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 bitc~ BTC    Bitc~ 2013-04-28       1  135.  136. 132.  134.       0
## 2 bitc~ BTC    Bitc~ 2013-04-29       1  134.  147. 134   145.       0
## 3 bitc~ BTC    Bitc~ 2013-04-30       1  144   147. 134.  139        0
## 4 bitc~ BTC    Bitc~ 2013-05-01       1  139   140. 108.  117.       0
## 5 bitc~ BTC    Bitc~ 2013-05-02       1  116.  126.  92.3 105.       0
## 6 bitc~ BTC    Bitc~ 2013-05-03       1  106.  108.  79.1  97.8      0
## # ... with 5 more variables: market <dbl>, close_ratio <dbl>,
## #   spread <dbl>, BTC_close <dbl>, BTC_value_close <dbl>
```

let’s make sure our date variable is of class "Date" with desire format:

```
sapply(topten_coins_meta,
       function(x) !all(is.na(as.Date(as.character(x),
                                      format="%Y-%m-%d"))))
```

```
##            slug          symbol            name            date
##           FALSE           FALSE           FALSE            TRUE
##         ranknow            open            high             low
##           FALSE           FALSE           FALSE           FALSE
##           close          volume          market     close_ratio
##           FALSE           FALSE           FALSE           FALSE
##          spread       BTC_close BTC_value_close
##           FALSE           FALSE           FALSE
```

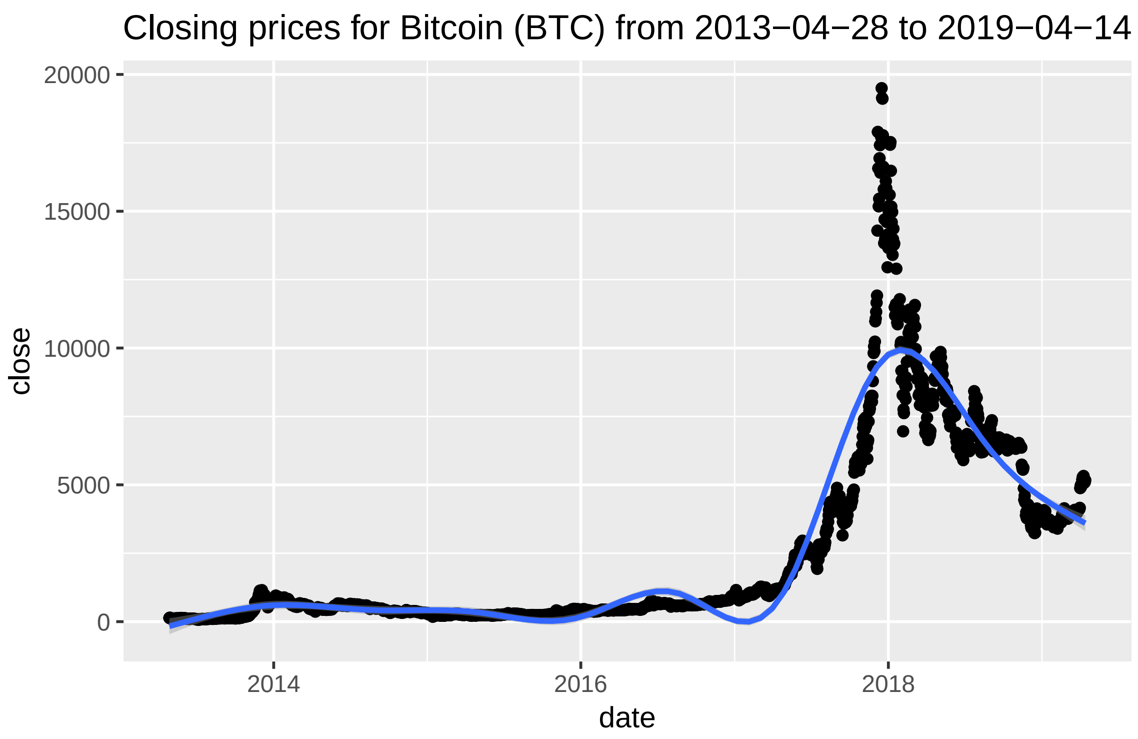
We notice from the resulting output above that date has a TRUE value confirming that we indeed have a Date object with the desire format. In our case, it was not necessary to perform a date conversion since our initial data was in the desire format but it is good practice to verify our date variable in our dataset.

3.2 Visualize

Plot relationship between data and closing price (BTC):

```
btc_hist_prices %>%
  ggplot(mapping = aes(x = date, y = close)) +
  geom_point() +
  geom_smooth() +
  ggtitle(paste("Closing prices for Bitcoin (BTC) from",
            min(btc_hist_prices$date), "to",
            max(btc_hist_prices$date))) +
  theme(plot.title = element_text(hjust = 0.5))
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Awesome !, we can immediately notice a surge of prices for the value of Bitcoin in USD towards the end of 2017 and an immediate decline of prices at the start of 2018.

what about visualizing data since one's first trade?. The following block of code will `filter()` and subset data containing data since one's first trade (we will do the same for top ten):

```
#OHLC since first trade
since_first_trade <- filter(btc_hist_prices, date >= "2017-07-29")
```

let's store the date of our first trade to a variable for future uses:

```
first_trade_date <- "2017-07-29"
```

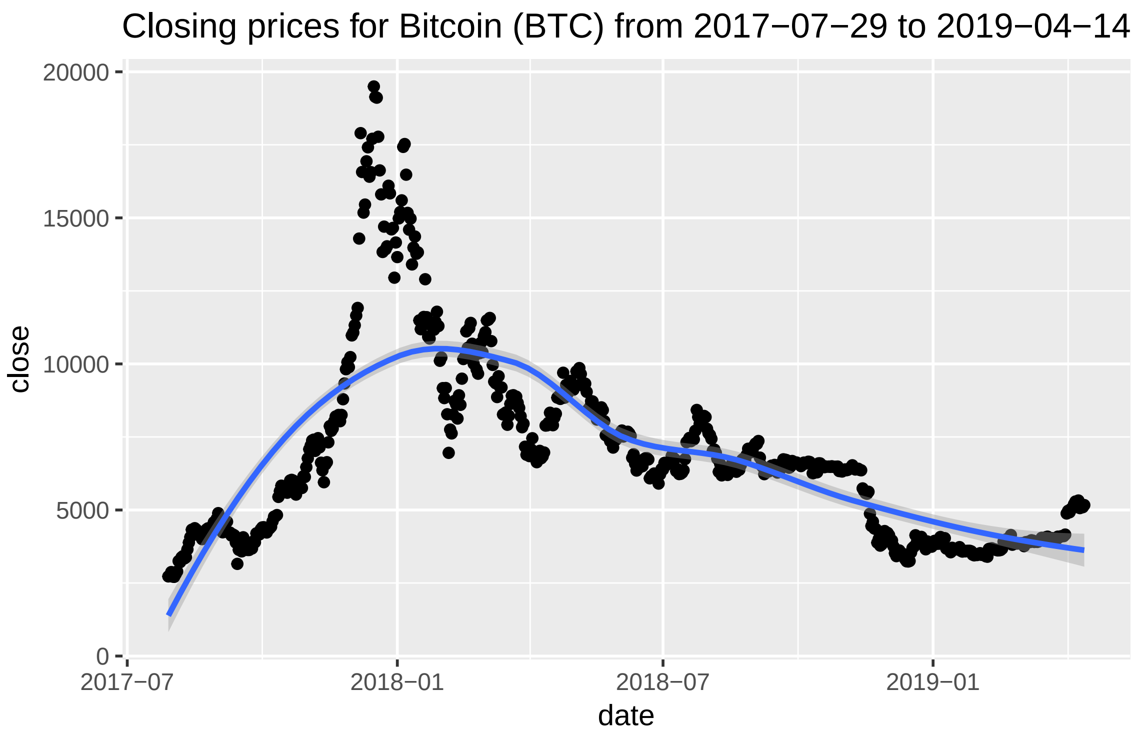
```
#OHLC top ten
topten_since_first_trade <- filter(topten_coins, date >= first_trade_date)

#lets view our dataset
since_first_trade
```

```
## # A tibble: 625 x 13
##    slug  symbol name  date       ranknow  open  high   low close volume
##    <chr> <chr>  <chr> <date>       <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
##  1 bitc~ BTC    Bitc~ 2017-07-29       1 2807. 2809. 2693. 2726. 8.04e8
##  2 bitc~ BTC    Bitc~ 2017-07-30       1 2724. 2759. 2645. 2757. 7.06e8
##  3 bitc~ BTC    Bitc~ 2017-07-31       1 2763. 2890. 2721. 2875. 8.61e8
##  4 bitc~ BTC    Bitc~ 2017-08-01       1 2871. 2921. 2686. 2718. 1.32e9
##  5 bitc~ BTC    Bitc~ 2017-08-02       1 2727. 2763. 2669. 2711. 1.09e9
##  6 bitc~ BTC    Bitc~ 2017-08-03       1 2710. 2813. 2685. 2805. 8.05e8
##  7 bitc~ BTC    Bitc~ 2017-08-04       1 2807. 2899. 2744. 2896. 1.00e9
##  8 bitc~ BTC    Bitc~ 2017-08-05       1 2898. 3290. 2875. 3253. 1.95e9
##  9 bitc~ BTC    Bitc~ 2017-08-06       1 3258. 3293. 3156. 3214. 1.11e9
## 10 bitc~ BTC    Bitc~ 2017-08-07       1 3213. 3398. 3181. 3379. 1.48e9
## # ... with 615 more rows, and 3 more variables: market <dbl>,
## #   close_ratio <dbl>, spread <dbl>
```

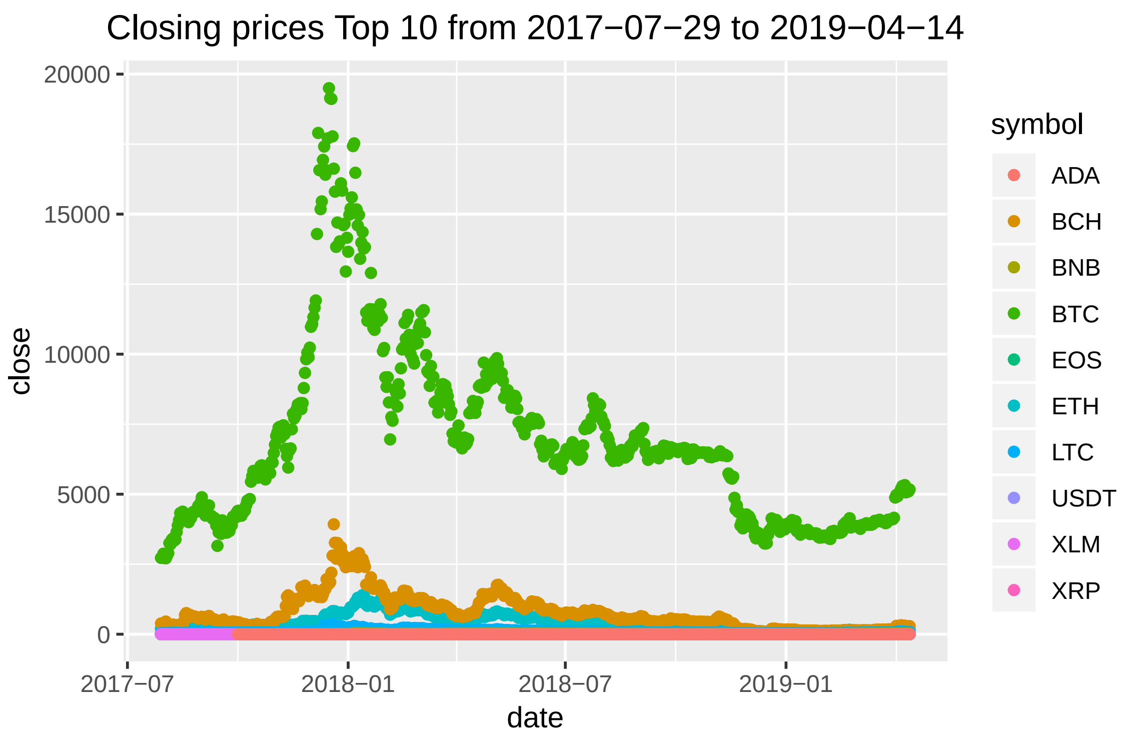
We can now plot our new datasets:

```
since_first_trade %>%
  ggplot(mapping = aes(x = date, y = close)) +
  geom_point() +
  geom_smooth() +
  ggtitle(paste("Closing prices for Bitcoin (BTC) from",
                min(since_first_trade$date), "to",
                max(since_first_trade$date))) +
  theme(plot.title = element_text(hjust = 0.5))
```

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

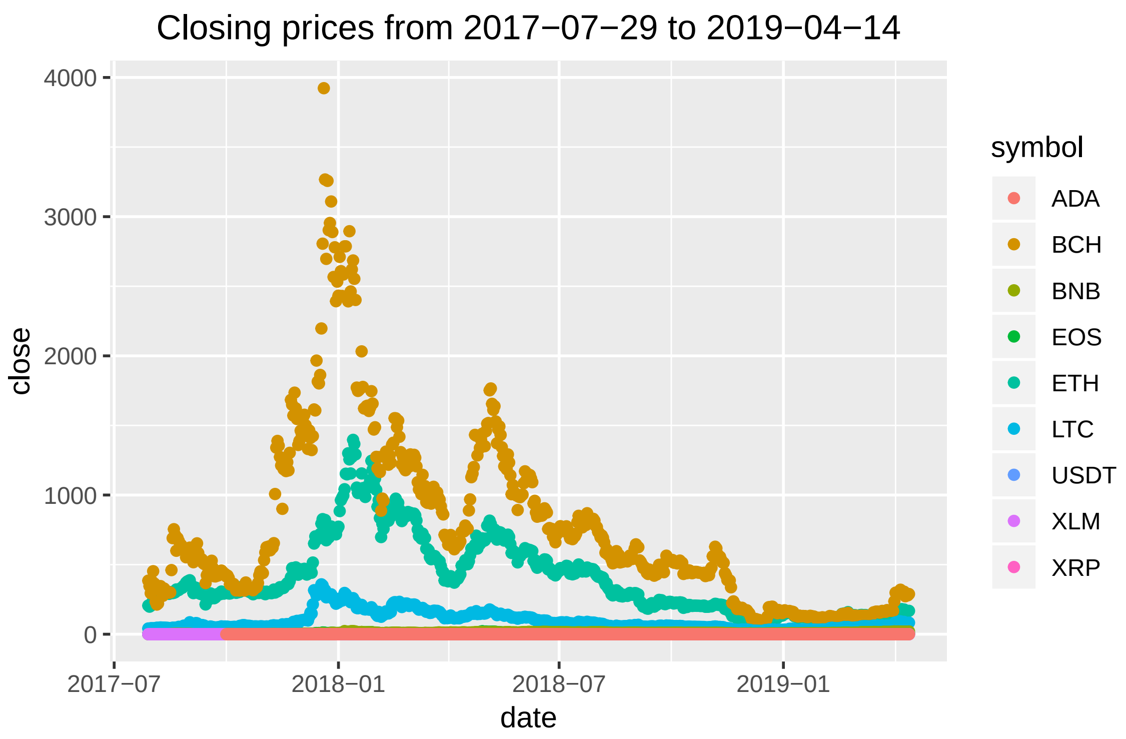


```
# plot relationship between date and closing price since first trade (top ten coins)
topten_since_first_trade %>%
ggplot(mapping = aes(x = date, y = close, color = symbol)) +
  geom_point() +
  ggtitle(paste("Closing prices Top 10 from",
                  min(topten_since_first_trade$date), "to",
                  max(topten_since_first_trade$date))) +
  theme(plot.title = element_text(hjust = 0.5))
```



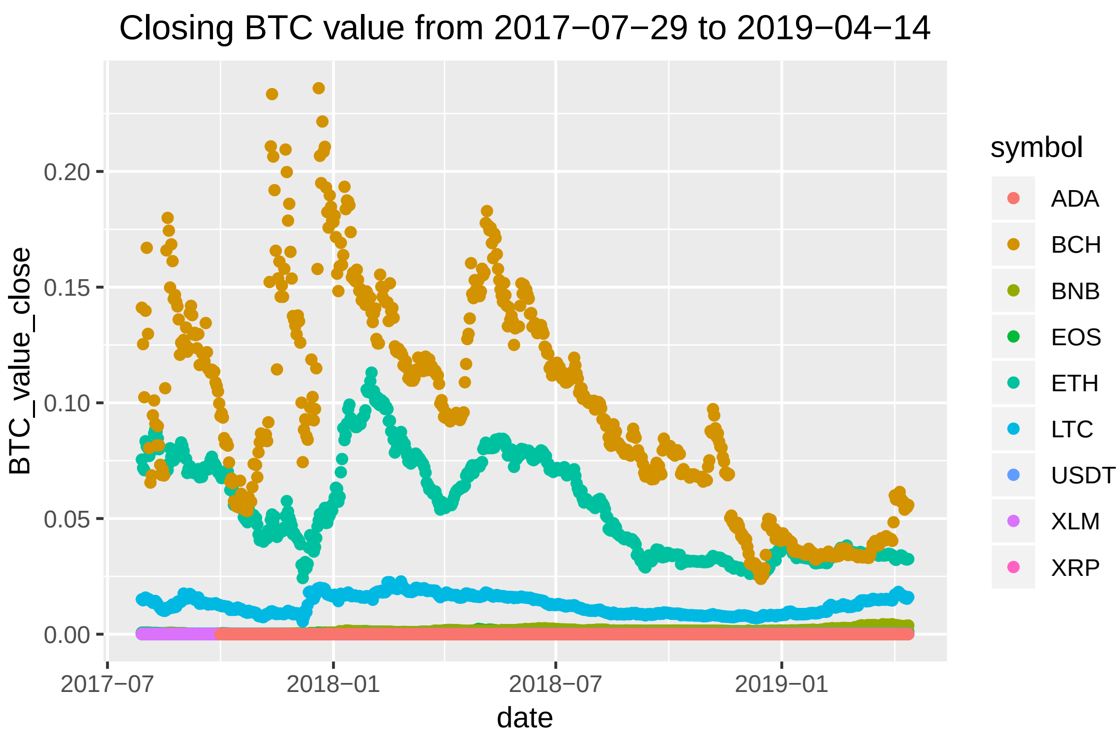
Let's remove BTC:

```
# plot relationship between date and closing price since first trade (top ten coins) no BTC

filter(topten_since_first_trade, symbol != "BTC") %>%
  ggplot( mapping = aes(x = date, y = close, color = symbol)) +
  geom_point() +
  ggtitle(paste("Closing prices from",
                min(topten_since_first_trade$date), "to",
                max(topten_since_first_trade$date))) +
  theme(plot.title = element_text(hjust = 0.5))
```



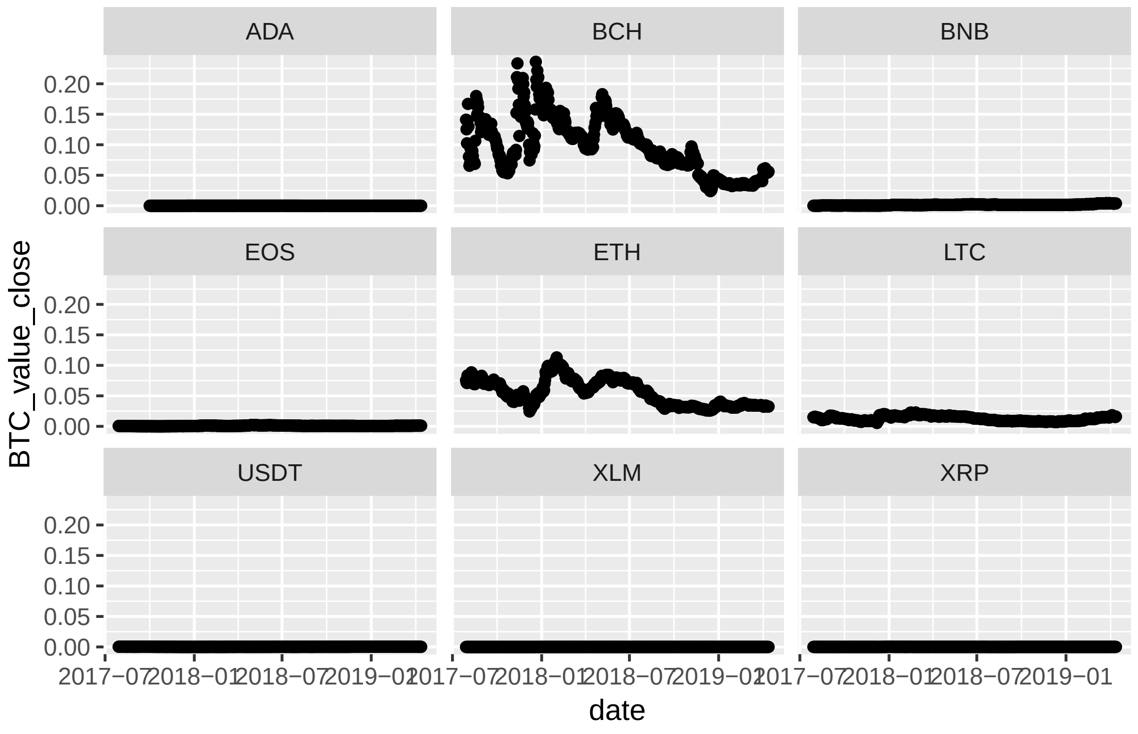
Lets plot same plot but view in terms of Bitcoin value or `BTC_value_close`:

```
# plot relationship between date and closing BTC value since first trade (top ten coins)
# no need to plot BTC (horizontal line at 1.0)

filter(topten_coins_meta, date >= first_trade_date, symbol != "BTC") %>%
  ggplot(mapping = aes(x = date , y = BTC_value_close, color = symbol)) +
  geom_point() +
  ggtitle(paste("Closing BTC value from",
                min(topten_since_first_trade$date), "to",
                max(topten_since_first_trade$date))) +
  theme(plot.title = element_text(hjust = 0.5))
```



Can we plot a different way? let's try using `facet()`:

```r
filter(topten_coins_meta, date >= first_trade_date, symbol != "BTC") %>%
  ggplot(mapping = aes(x = date, y = BTC_value_close)) +
  geom_point() +
  facet_wrap(~ symbol, nrow = 3)
```



4 Model

To begin our modeling stage, we need to have a time series numeric vector or time series of class `ts`:

4.1 Time series data pre-processing

```
startW <- as.numeric(strftime(head(btc_close_dates$date, 1), format = "%W"))
startD <- as.numeric(strftime(head(btc_close_dates$date, 1) + 1, format =" %w"))

btc_close_ts <- ts(btc_close_dates$BTC_close,
                  frequency = 7,
                  start = c(startW, startD)
                  )

btc_close_ts_freq1 <- ts(btc_close_dates$BTC_close,
                  frequency = 1,
                  start = c(startW, startD)
                  )

#spring 2019 classes begin
btc_close_sp19 <- btc_close_dates %>%
  filter(date >= "2019-01-07")


startW_sp19 <- as.numeric(strftime(head(btc_close_sp19$date, 1), format = "%W"))
startD_sp19 <- as.numeric(strftime(head(btc_close_sp19$date, 1) + 1, format =" %w"))

btc_close_sp19_ts <-ts(btc_close_sp19$BTC_close,
                      frequency = 7,
                      start = c(startW_sp19, startD_sp19)
                      )
```

check first and last observation weekdays align with time series objects:

```
# print(btc_close_ts, calendar = T)
head(btc_close_ts, 10)
```

```
## Time Series:
## Start = c(16, 1)
## End = c(17, 3)
## Frequency = 7
##  [1] 134.21 144.54 139.00 116.99 105.21  97.75 112.50 115.91 112.30 111.50
```

```
head(btc_close_dates$date, 1)
```

```
## [1] "2013-04-28"
```

```
weekdays(head(btc_close_dates$date,1))
```

```
## [1] "Sunday"
```

```
tail(btc_close_dates$date, 1)
```

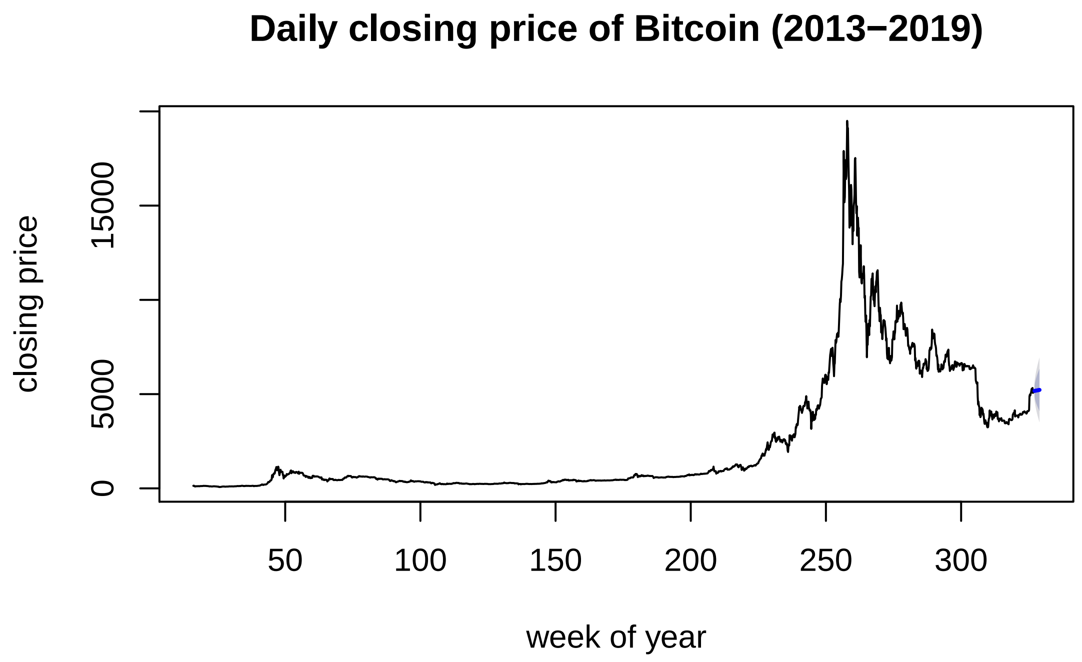
```
## [1] "2019-04-14"
```

```
weekdays(tail(btc_close_dates$date,1))
```

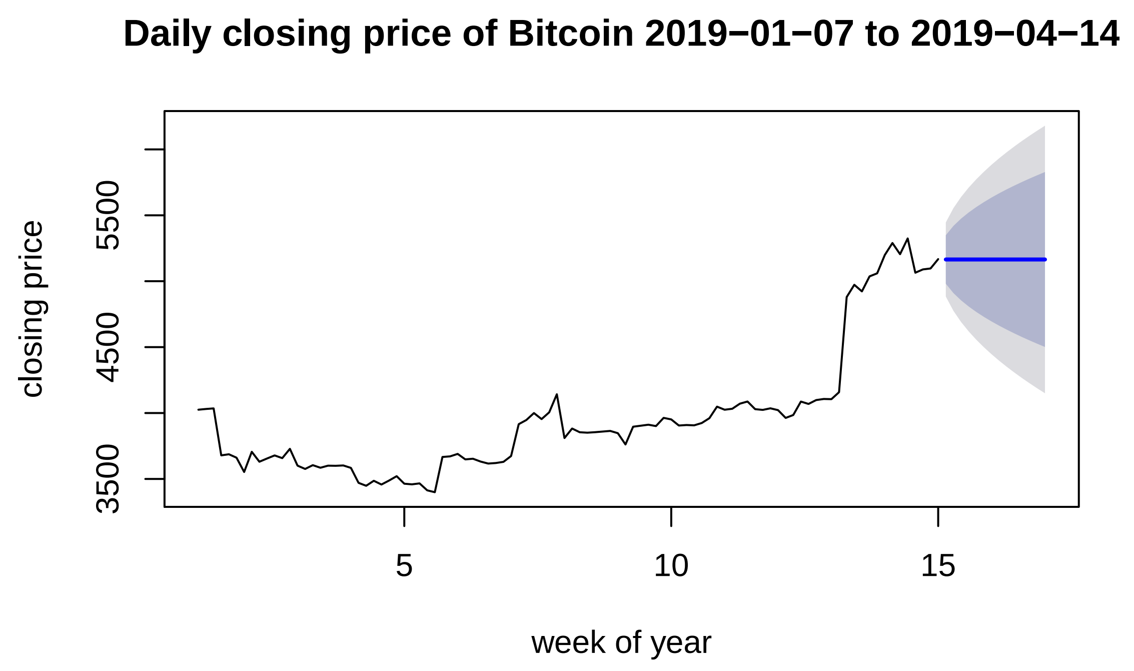
```
## [1] "Sunday"
```

4.2 Exponential smoothing

```
fit1 <- ets(btc_close_ts)

plot(forecast(fit1),
     xlab="week of year",
     ylab="closing price",
     main="Daily closing price of Bitcoin (2013-2019)")
```



```
fit2 <- ets(btc_close_sp19_ts)
plot(forecast(fit2),
     xlab="week of year",
     ylab="closing price",
     main="Daily closing price of Bitcoin 2019-01-07 to 2019-04-14")
```



4.3 Arima modeling

```
fit_opt_btc_close <- auto.arima(btc_close_ts_freq1)

fit_opt_btc_close
```

```
## Series: btc_close_ts_freq1
## ARIMA(1,1,1)
##
## Coefficients:
##           ar1     ma1
##       -0.6626  0.7394
## s.e.   0.0805  0.0716
##
## sigma^2 estimated as 52540:  log likelihood=-14919.3
## AIC=29844.6   AICc=29844.61   BIC=29861.65
```

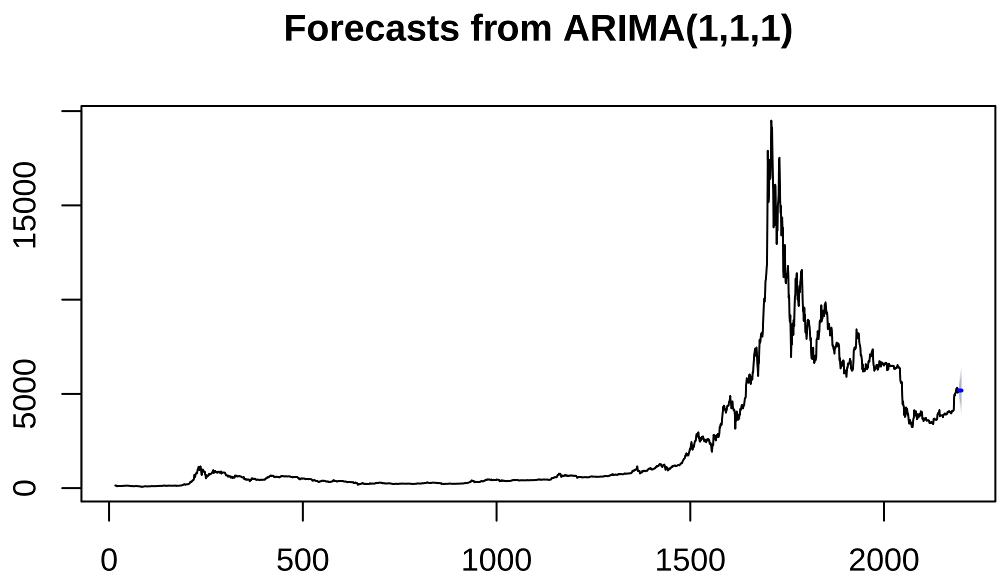
```
coef(fit_opt_btc_close)
```

```
##        ar1        ma1
## -0.6625775  0.7394288
```

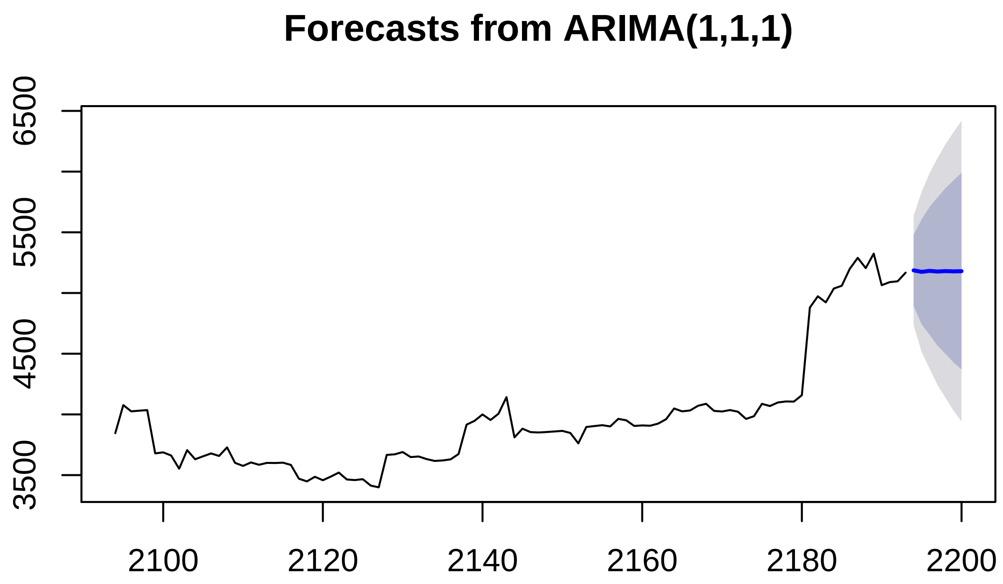
```
#use predictive function
predict(fit_opt_btc_close, n.ahead = 7, se.fit = TRUE)
```

```
## $pred
## Time Series:
## Start = 2194
## End = 2200
## Frequency = 1
## [1] 5186.526 5174.066 5182.322 5176.852 5180.476 5178.075 5179.666
##
## $se
## Time Series:
## Start = 2194
## End = 2200
## Frequency = 1
## [1] 229.2152 336.8457 410.8100 477.2437 533.1936 585.1968 632.0880
```

```
#visualize results
btc_forecast <- forecast(object = fit_opt_btc_close, h = 7)
plot(btc_forecast)
```



```
#You can include = X observations of the original series in your plot with:

plot(forecast(fit_opt_btc_close, h= 7), include = 100)
```



5 Conclusion and Future Work

In this paper, the use of time series data is used to work through all the stages of the *Data Science Workflow*, results of these efforts are reported here. This report, written as part of coursework of *Data Science with R*, is intended as a first step towards understanding the underlying structure and functions involve in working with time series data. For this reason, I look forward to continue working with financial time series data, in particuarly, crypto market data and update results accordingly to github repo.

References

Nakamoto, Satoshi. 2008. "Bitcoin: A Peer-to-Peer Electronic Cash System."