**Goal: Create a program that gets us, for a specified time period, a list of major price fluctuations (i.e. “Events” in which the price fluctuated up/down by a significant amount) and a list of news articles correlated to each “event”.**

Technical/algorithmic overview:

**Part I: Getting raw data**

**News Sources:**

* <https://www.coindesk.com/> (They have an API)
  + <https://www.programmableweb.com/api/coindesk>
* <https://cryptopanic.com/news?filter=important>
  + This is probably the closest to our project. However, it seems like tags of bullish, bearish, important are crowdsourced and not generated by the algo. These are features that we’d like instated to but are secondary to the algorithm generating a score. In our world, the algo tells you how relevant a piece of news is and then after it appears on our feed, users can upvote, downvote or tag it for being important.
  + They even point to the currencies that the article could impact
* <https://github.com/coinpride/CryptoList>
  + Good list, I suggest narrowing down and using the top 3
* <https://coinbuzz.stream/>
  + News aggregator from all over the web -displays using Trello. Not sure if they have an API

**Getting news articles**

1. Get list of “verified”/”reputable” sources from Anand/team
2. Get URLs of each source’s RSS feed
3. Combine all RSS feeds into one master RSS feed (That is constantly being updated with news articles as they get published) (This feed URL is the feed URL we input as feed\_url in the function below)
4. Create function *getArticles* that returns a dataframe with columns: timestamp, article title, publisher, and article URL

|  |
| --- |
| # Pseudocode  *getArticles(feed\_url, topic = “BTC”, startDate, endDate){*  *all\_articles = get dataframe of all articles in feed\_url from startDate to endDate, with columns: article title, article URL, timestamp of publication, name of source*  *topical\_articles = from all\_articles, import into topical\_articles any article (and all its columns) whose title contains the “topic” specified in the parameters (in this case, any article about BTC)*  *return topical\_articles, a dataframe with columns: timestamp, article title, publisher, and article URL for each article*  *}* |

*\*Note - We can ignore the “topic” parameter for now and just use Bitcoin to keep things simpler.*

**Getting asset prices**

1. Identify data source: Coindesk tracks bitcoin closing price by the minute. Link: <https://www.coindesk.com/price/> (*\*Let’s rely on this as our primary source for BTC price for now*)
2. Create function *getPrices* that returns a dataframe with timestamp and asset price

|  |
| --- |
| # Pseudocode  *getPrices(startDate, endDate){*  *prices = dataframe containing export data from* <https://www.coindesk.com/price/> *between startDate and endDate. Two columns: timestamp and price.*  *return prices*  *}* |

**Part II: Processing price data**

Create function *processPriceData*, which takes in a dataframe with columns: timestamp (minute-by-minute pricing) and price (price in USD). Returns a new dataframe with columns: time (the hour in which the price activity happened), meanPrice (the mean price in the hour), priceSTD (the standard deviation of the price within that hour)

|  |
| --- |
| # Pseudocode  *processPriceData(prices){*  *Convert every element in the timestamp column of “prices” into a datetime object*  *hourlyData = dataframe with columns: time, meanPrice and priceSTD*  *Fill hourlyData with rows for each hour within “prices”, i.e. traverse “prices” and find the start time and end time, then create a list of all the 1-hour windows within that period. E.g. If “prices” goes from “2018-02-24 0:00AM” to “2018-02-25 11:59PM”, then hourlyData will have rows: “2018-02-24 0:00AM to 0:59AM”, “2018-02-24 1:00AM to 1:59AM”, “2018-02-24 2:00AM to 2:59AM”, [...], “2018-02-25 10:00PM to 0:59AM”, “2018-02-25 11:00PM to 11:59PM”.*  *For each row in hourlyData (and the corresponding 60 1-minute rows of data in “prices”), calculate mean and stdDev, and fill in the columns in hourlyData*  *return hourlyData*  *}* |

**Part III: Identifying significant price movement**

How do we determine if any given hour block contains/represents a significant (above 2 SDs in magnitude) movement?

Compare to opening price at the start of the time window?

Compare to previous hourly block?

Not sure here.

|  |
| --- |
| # Pseudocode  *getDeviantHours(hourlyData){*  *Go through hourlyData and identify deviant hour blocks (somehow)*  *return deviantHours, a dataframe with columns: time (the hour block within which the deviance occurred), meanPrice and priceSTD*  *}* |

**Part IV: Correlating significant price movement with articles**

Once we have deviantHours and know *when* there was significant price movement, we can turn back to topical\_articles from part I (with the same overall time frame as deviantHours) and find correlated articles.

|  |
| --- |
| # Pseudocode  *getRelevantArticles(hourlyData, topical\_articles){*  *create hourly\_topical\_articles, a dataframe which follows a similar time-slicing algorithm like the one in Part II, containing a list of hourly blocks and then a list of articles within each dataframe*  *for each hourly block in hourlyData, pull the hourly block from hourly\_topical\_articles*  *return a new dataframe relevantArticles that basically puts hourlyData and hourly\_topical\_articles side by side so we can go through the info manually*  *}* |