Economic Sea Urchins: Viewing Market Trends on a Clock

Sam Craig

Oberlin College, Oberlin, Ohio, USA; scraig@oberlin.edu

Abstract

Resembling a sea urchin or the hands of a clock, I provide a program that converts stock data into graphs rotated along a circle. Beyond the aesthetics of this possibly seeming like a sea urchin with jagged spikes, attributes of this hemispherical structure can yield insights into the market that it represents.

Introduction

The premise of economic sea urchins came from a numerical similarity: there are 365 degrees around a circle and 360 days in a year. Noting these numbers as being close, I wondered if there were some time-based data where I could attach one day to a one degree rotation along a circle, drawing a line representing that data out of the origin of the circle. I considered stocks because they are commonly shown as a graph. Because of how small a change in one degree is, I ended up attempting a plan involving one graph for each month of a year. This results in 12 lines that I wanted to rotate along a circle.

This is not an uncommon idea—after all, analog clocks themselves split a circle into twelve regions with lines jutting out of the origin. While occupying an entire circle would allow for 30 degrees of space being occupied for each line area (in the case of 12 lines), allowing more subtle details of the graph to be seen, I opted for only using the upper hemisphere of the circle. This was to fulfill an art goal: producing a creation that looks like a sea urchin but with jagged spikes. However, as we will find in the section "Reading Market Trends from an Urchin," restricting the spacing to half a circle can encourage trends to be realized. Below I present an example urchin of aesthetic value to me.

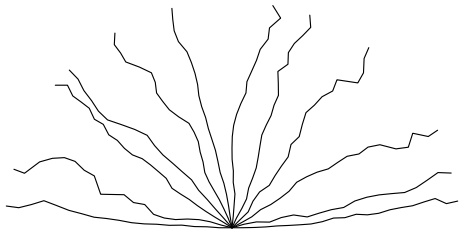

Figure 1: *urchin of the 2018 Dow Jones*

Creating an Urchin

To get the most detail out of this section, the code itself that was used to generate these urchins will be provided. It was all written in Ruby. This section will summarize the process used to create the urchins.

The program is split into three primary components: a stock preparer, a grapher, and a combiner. These components take input from stdin and output from stdout and mostly communicate through pipes. These components are detailed below.

Grapher The primary role of the grapher is to retrieve data and convert it into PostScript code that defines a graph of this data. In order to do this, the grapher scales all retrieved data to a standard viewing frame. Because we are dealing with graphs of stocks, the output paths assume increasing x values and y values that can be either positive or negative.

```
2020-02-28,25270.8301,25494.2402,24681.0098,25409.3594,915990000
2020-02-27,26526.0000,26775.3105,25752.8203,25766.6406,664980000
2020-02-26,27159.4609,27542.7793,26890.9707,26957.5898,472450000
2020-02-25,28037.6504,28149.1992,26997.6191,27081.3594,513270000
2020-02-24,28402.9297,28402.9297,27912.4395,27960.8008,452580000
2020-02-21,29146.5293,29146.5293,28892.6992,28992.4102,311210000
2020-02-20,29296.2500,29368.4492,28959.6504,29219.9805,287780000
2020-02-19,29312.6992,29409.0898,29274.3809,29348.0293,240640000
2020-02-18,29282.7793,29330.1602,29116.8105,29232.1895,256600000
2020-02-14,29440.4707,29463.0391,29283.1797,29398.0801,231000000
2020-02-13,29436.0293,29535.4004,29345.9297,29423.3105,291150000
2020-02-12,29406.7500,29568.5703,29406.7500,29551.4199,309530000
2020-02-11,29390.7109,29415.3906,29210.4707,29276.3398,279540000
2020-02-10,28995.6602,29278.0703,28995.6602,29276.8203,250510000
2020-02-07,29286.9199,29286.9199,29056.9805,29102.5098,252860000
2020-02-06,29388.5801,29408.0508,29246.9297,29379.7695,263700000
2020-02-05,29048.7305,29308.8906,29000.8496,29290.8496,357540000
2020-02-04,28696.7402,28904.8809,28696.7402,28807.6309,332750000
2020-02-03,28319.6504,28630.3906,28319.6504,28399.8105,307910000
```

```
0 0.0
1 377.08980000000156
2 729.0801000000029
3 1068.9297000000006
4 967.2695000000022
5 676.0097999999998
6 1071.0604999999996
7 1087.0996000000014
8 1116.3788999999997
9 1120.820300000003
10 963.1288999999997
11 993.0488000000005
12 976.5996000000014
13 826.8788999999997
14 83.27930000000197
15 -282.0
16 -1160.1895000000004
17 -1793.6503999999986
18 -3048.8202999999994
```

Figure 2: *(l) raw stock data; (r) the corresponding grapher input*

Stock Preparer In order to make the role of the grapher easier, there is an intermediary process between raw stock data (retrieved from AlphaVantage[1]) and input data to the grapher. The reason for this is to make the grapher more versatile in handling more possibilities—what if the goal were to use stock data that did not follow the format of that of AlphaVantage? The preparer takes the stock data and puts it into a friendly format for the grapher. One key element of translation is to define $y = 0$ as the opening stock price of the first chronological entry. Each of the corresponding entries are relative to this first entry in the data provided to the grapher. An example of the translation is shown in Figure 2.

Combiner The combiner facilitates communication between the stock preparer and grapher and produces an actual urchin. The grapher component itself simply produces the PostScript for a graph, but does nothing toward actually rotating the graphs about a point. In order for the combiner to put multiple graphs rotated about a point together, it accepts a file that defines attributes that the stock preparer should use when looking for data. This has been given the term a "combiner file," and an example is included in Figure 3. Each line is a new graph that will be displayed in the final

```
-y 2002 -m 1
-y 2002 -m 2
-y 2002 -m 3
-y 2002 -m 4
-y 2002 -m 5
-y 2002 -m 6
-y 2002 -m 7
-y 2002 -m 8
-y 2002 -m 9
-y 2002 -m 10
-y 2002 -m 11
-y 2002 -m 12
```

Figure 3: *an example combiner file*

urchin. The combiner asks the preparer to prepare stock data given these attributes, and then asks the grapher to generate PostScript code for the data. Then, the combiner puts all of the graphs in the same file, rotated accordingly.

Auxiliary Components In my initial batch of economic sea urchins, I generated both urchins with graphs of the Dow Jones of each month within a year (January, February, . . .) from 2000 through 2019 as well as urchins with graphs of each instance of a month from 2000 through 2019 (April 2000, April 2001, . . .). In order to make this painless, I created a fourth auxiliary component, called the Generator, that allowed me to generate combiner files for all years and sequences of months in my data set quickly.

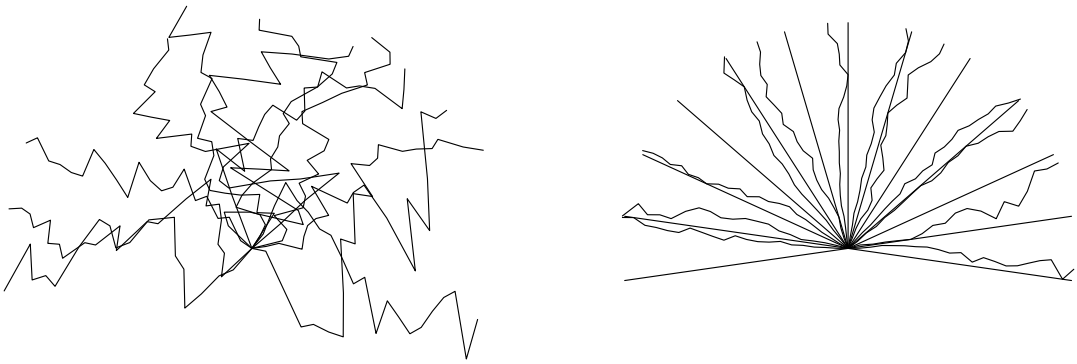

Figure 4: *(l) free urchin (each month of the Dow Jones of 2002); (r) intersection-restricted urchin with zones marked (same dataset)*

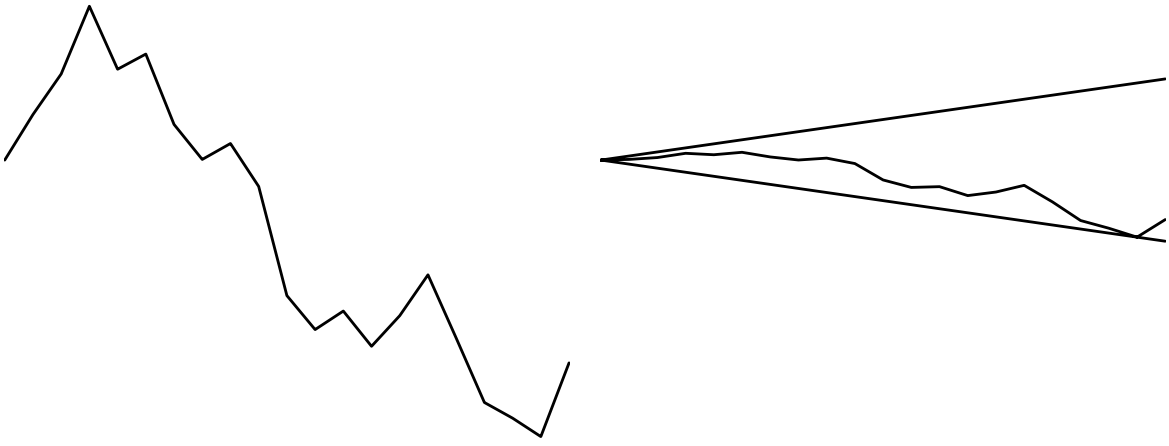

Figure 5: *(l) free graph of the June 2001 Dow Jones; (r) intersection-restricted graph with cone limits displayed (same dataset)*

Restricting Intersections

In my initial batch of economic sea urchins, the urchins were rather chaotic and it was impossible to tell what was going on. However, by restricting each graph to a zone of degrees of the circle ($180/(n-1)$ degrees, where n is the number of graphs to be displayed), it is possible to remove overlaps of lines. This restricting is done by scaling the maximum y value of the graph from the original constant value of 98 (arbitrarily picked to allow for ease of viewing) to the function $y_{\max} = x \tan\theta$, where θ is the rotation between two graphs. This rotation is equal to the number of degrees that defines a zone, as defined above. This equation is derived from a right triangle bounded by a point on a circle, the origin, and the vertical line at x. Figure 4 displays the difference in the 2002 Dow Jones using the intersection restriction or not. While this makes it a lot more clear which individual month is which, this does reduce the total amount of detail that can be displayed in the graph. Since the y limit in the graph scaling is relative to x, it results in very little variation at low x and high variation at high x. Figure 5 displays an example of the loss in detail resulting from restricting intersections. In the end, while this loss of variation is indeed a disappointing loss, the gains in reading overall trends by restricting intersections are helpful, as detailed in the next section.

Reading Market Trends from an Urchin

While the origin of this project derived entirely from aesthetics and with no plan for future analysis of the data employed, after restricting intersections I found some patterns that could be used to glean information about the markets from the urchin.

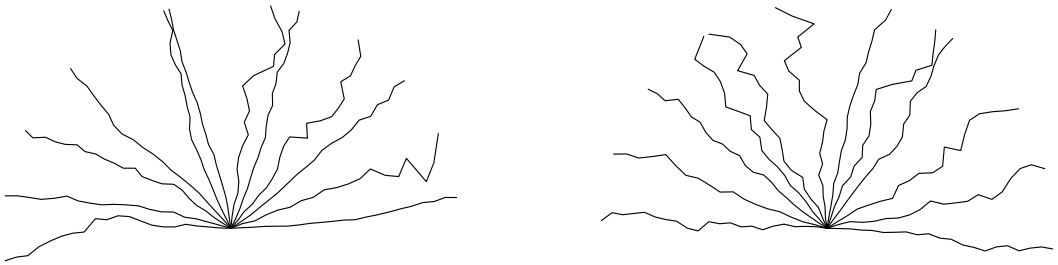

Figure 6: *(l) urchin of the 2013 Dow Jones; (r) urchin of the 2008 Dow Jones*

Tilt of an Urchin Because the spikes coming out of the urchin are rotated along a circle, a positive graph will go upwards, or y positive, at low rotations and downwards, or y negative, at high rotations. Over the course of an entire urchin, this can result in a very slight tilt in one direction. Two examples are included in Figure 6. In the case of 2013, the urchin has slight tilt to the left, and the Dow Jones had an incredible bull year, gaining 26.50% over the course of the year[2]. In the case of 2008, the urchin tilts right slightly, and the Dow Jones had a bear year of 33.84% lost[2]. Both of these tilts are very slight, and human perception of them may be too dependent on the first and last spikes. But, these trends are actually amplified in the free urchins. Figure 7 illustrates massive tilts in these same years.

Spacing Between Spikes Urchins can also give an indication of the general stability of the market. Here, we are defining stability as consistency in either trending upward or downward. This stability can be seen in the spacing between the ends of the spikes coming out of the urchin. If the degree distance between these

Figure 7: *(l) free urchin of the 2013 Dow Jones; (r) free urchin of the 2008 Dow Jones*

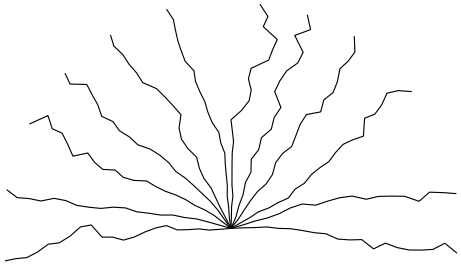

Figure 8: *urchin of the 2009 Dow Jones*

spikes is consistently close to $180/(n-1)$ as defined earlier, the market is doing a similar action (percent rise or fall) each month and is therefore stable and reliable. Meanwhile, if the degree distance is 0 between two spikes and $2(180/(n-1))$ for the next two spikes, the market is wildly fluctuating between strong gains and strong losses and it is hard to truly know what to expect from the market. Figure 8 shows a market with a fairly stable second half of the year—note that the first half of the year is decently stable minus a couple months that deviated from the trend, producing large gaps.

References

[1] AlphaVantage, TIME_SERIES_DAILY (DJI stock data), Retrieved 30 March 2020.

[2] Macrotrends, "Dow Jones - 10 Year Daily Chart", Retrieved 7 April 2020. https://www.macrotrends.net/1358/dow-jones-industrial-average-last-10-years