EXPONENTIALLY WEIGHTED STATISTICS

.

1. Motivation

Very frequently in trading model design, we need to know how a variable compares to recent history. As data points recede into the past, however, their importance to current trading strategies declines. In effect, at time t, we want to include points x_{t-s} in the past with declining weight $W = W(s)$ depending on the age s. We can then get a weighted sum

$$S_t = \int_{s=0}^{\infty} W(s) x_{t-s} ds$$

or its near equivalent weighted average

$$A_t = \frac{\int_{s=0}^{\infty} W(s) x_{t-s} ds}{\int_{s=0}^{\infty} W(s) ds}.$$

Often, especially for the common operation of averaging, we rescale our weights to make life easier

$$w(s) := \frac{W(s)}{\int_{s=0}^{\infty} W(s) ds}.$$

and frequently we deal in discrete sampling where sums take the place of these integrals so that we may say for example

$$A_{t_j} = \sum_{i=0}^{\infty} w_i x_{j-i}.$$

Fans of functional analysis or signal processing will recognize this as a convolution operation.

We could consider many possible functions for such a weighting scheme. The simplest, conceptually, is a moving *boxcar* window where we discard any points older than, say $s = 30$ days. The trouble with boxcar weighting is that, by not smoothly decaying old data, it gives us surprises as we run the business. The reason is easy to see. Imagine we start with a relatively stable set of observations

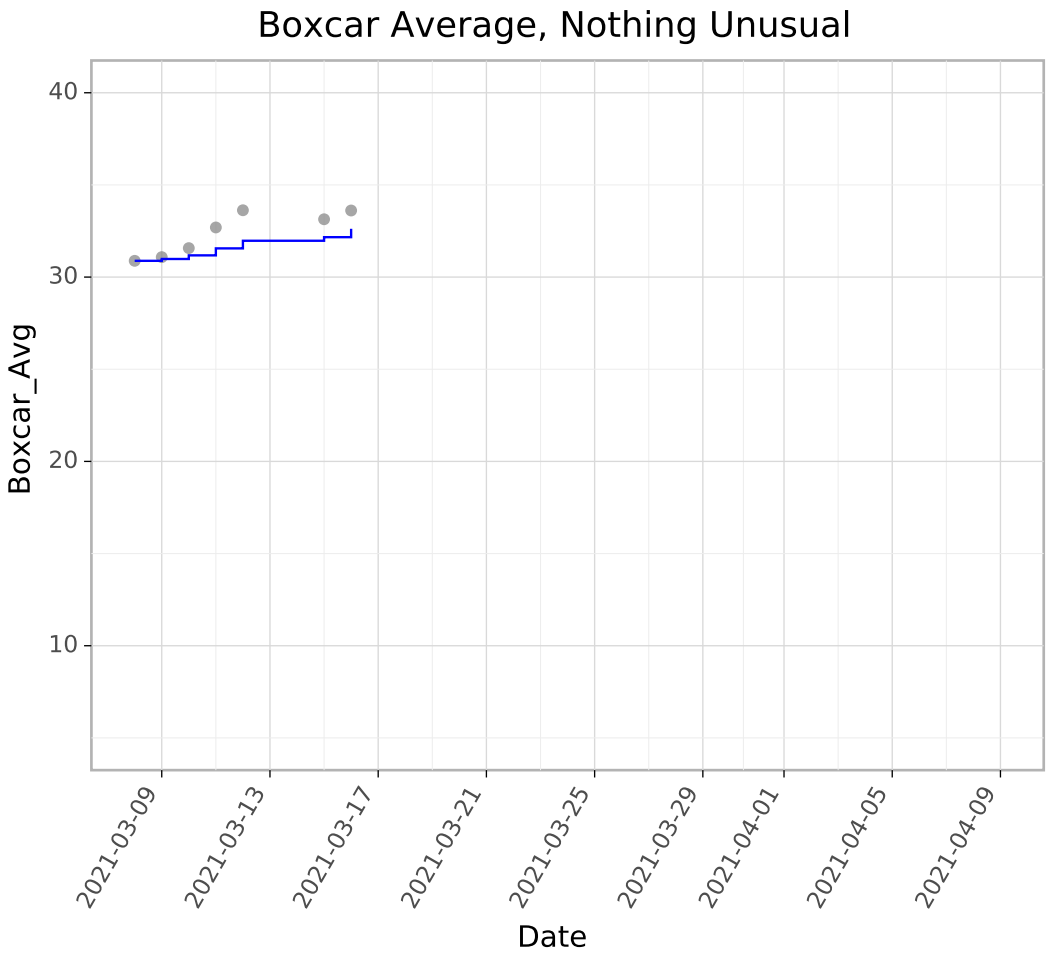

FIGURE 1. A boxcar average doing a nice job

We get one big outlier, which naturally brings down the average in a sizable jump

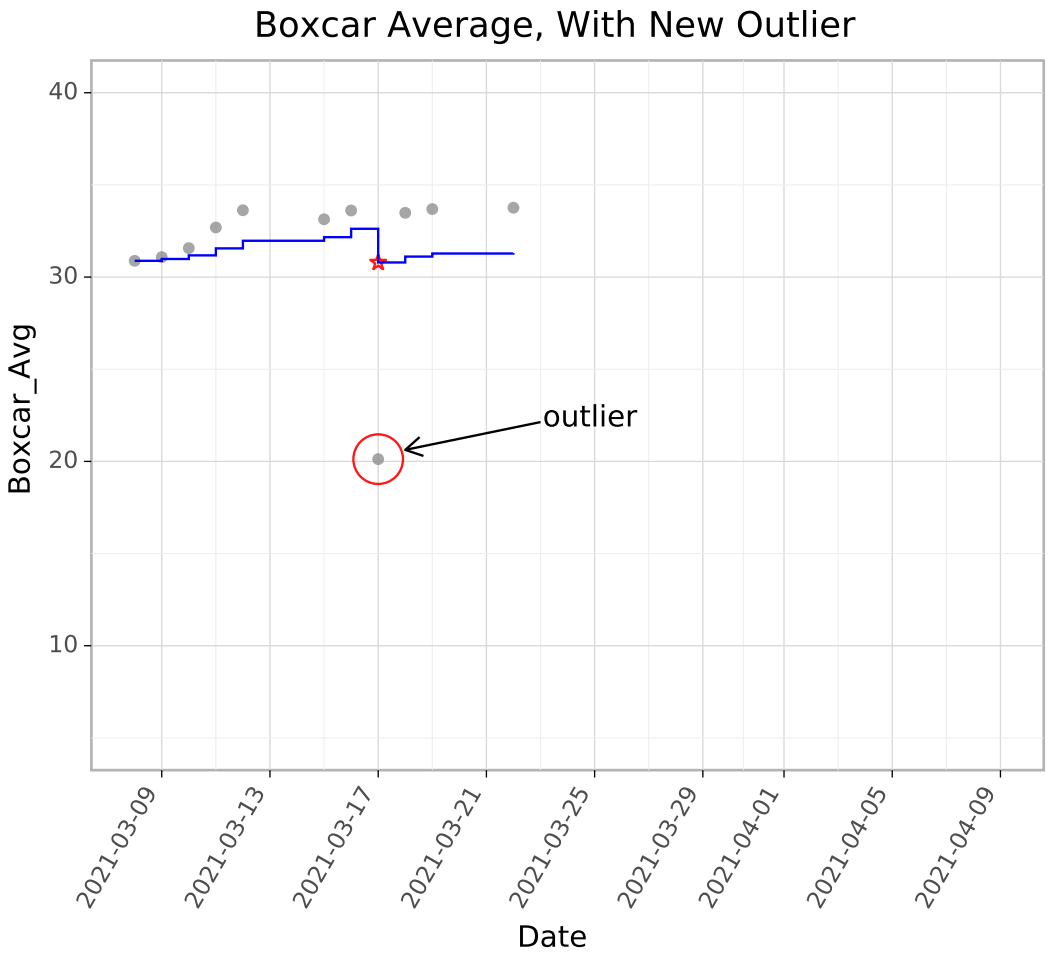

FIGURE 2. New information drops the average quite a bit, as expected

And then, after our window has passed, the average jumps again, in the opposite direction

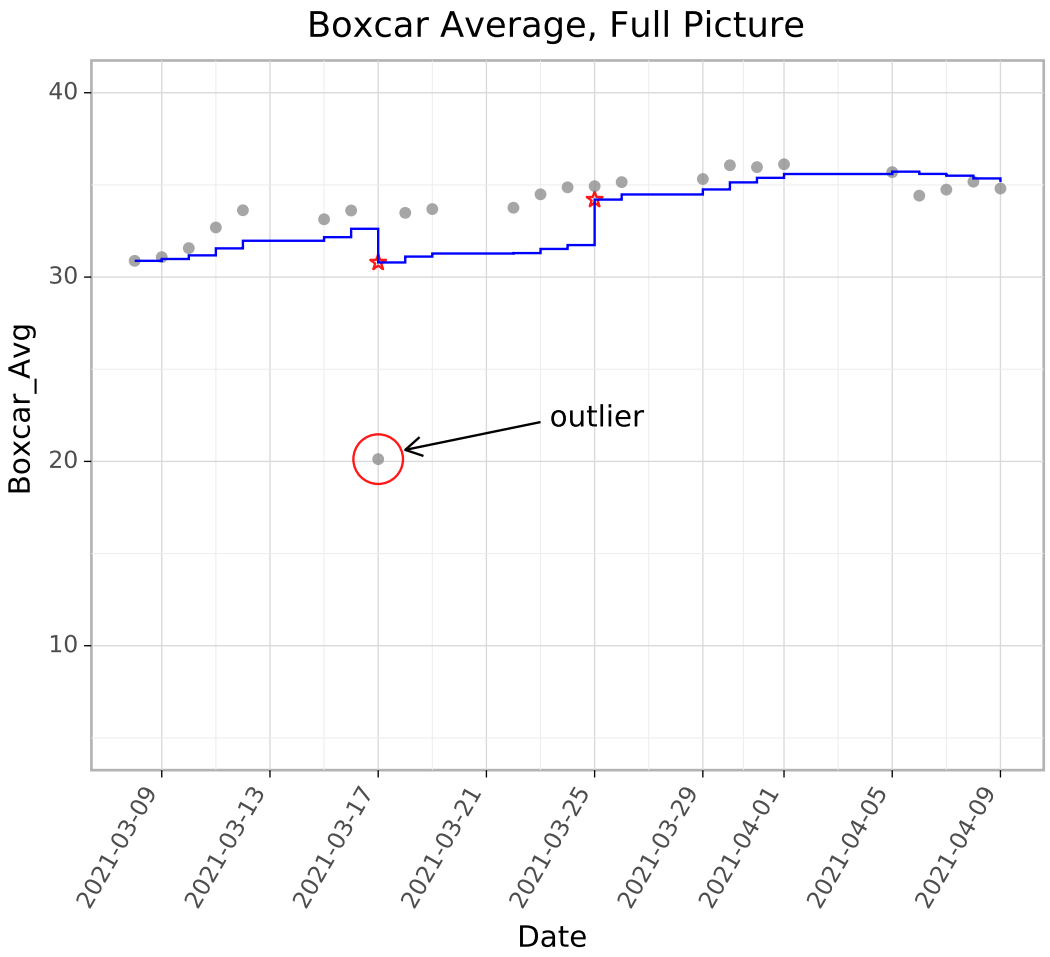

FIGURE 3. After the boxcar window has passed, we get a *second* jump

It's perfectly normal to have your systems react in a big way to a new outlier. After all, that outlier represents important new information arriving. What is *not* normal is to have a big reaction to the outlier falling out of your averaging window. The window size was your arbitrary choice, not a bit of important new data. Especially if the averaging period is more than a few weeks, people will forget about the outlier and then go into panic mode when model parameters take big jumps due to this behavior.

We can avoid this by choosing a "tent" function where weight declines linearly up to our limit, which on daily data with a 30 day limit would set $w = 0$ for $s \geq 30$ and otherwise choose for $i = 0, \ldots, 29$

$$w_i := w(s_i) = \frac{30 - i}{30\left(\frac{30+1}{2}\right)}$$

Any weighting scheme we desire is available to us, but it turns out one of them is special. Namely, if we choose the *exponential weighting* scheme with coefficient $\eta < 0$ and $i = 0, \ldots, \infty$

$$w_i := w(s_i) = \eta^i \cdot (1 - \eta)$$

with the coefficient $1 - \eta$ chosen because $1 + \eta + \eta^2 + \cdots = \frac{1}{1-\eta}$.

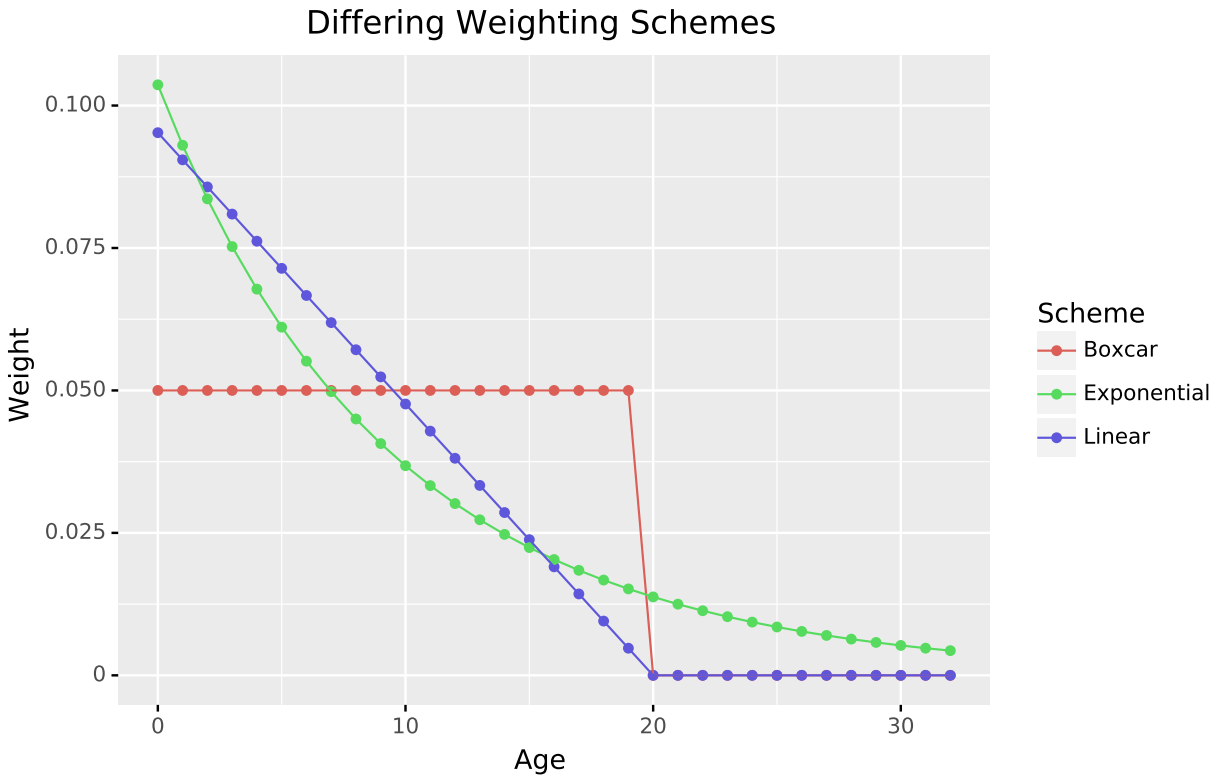

FIGURE 4. Weights decaying with age. Here the exponential center of mass was set to 10.

Examining the effect on actual data, we can apply these weighting schemes to a few years' Sunoco closing prices. On the full scale it is hard to see much difference

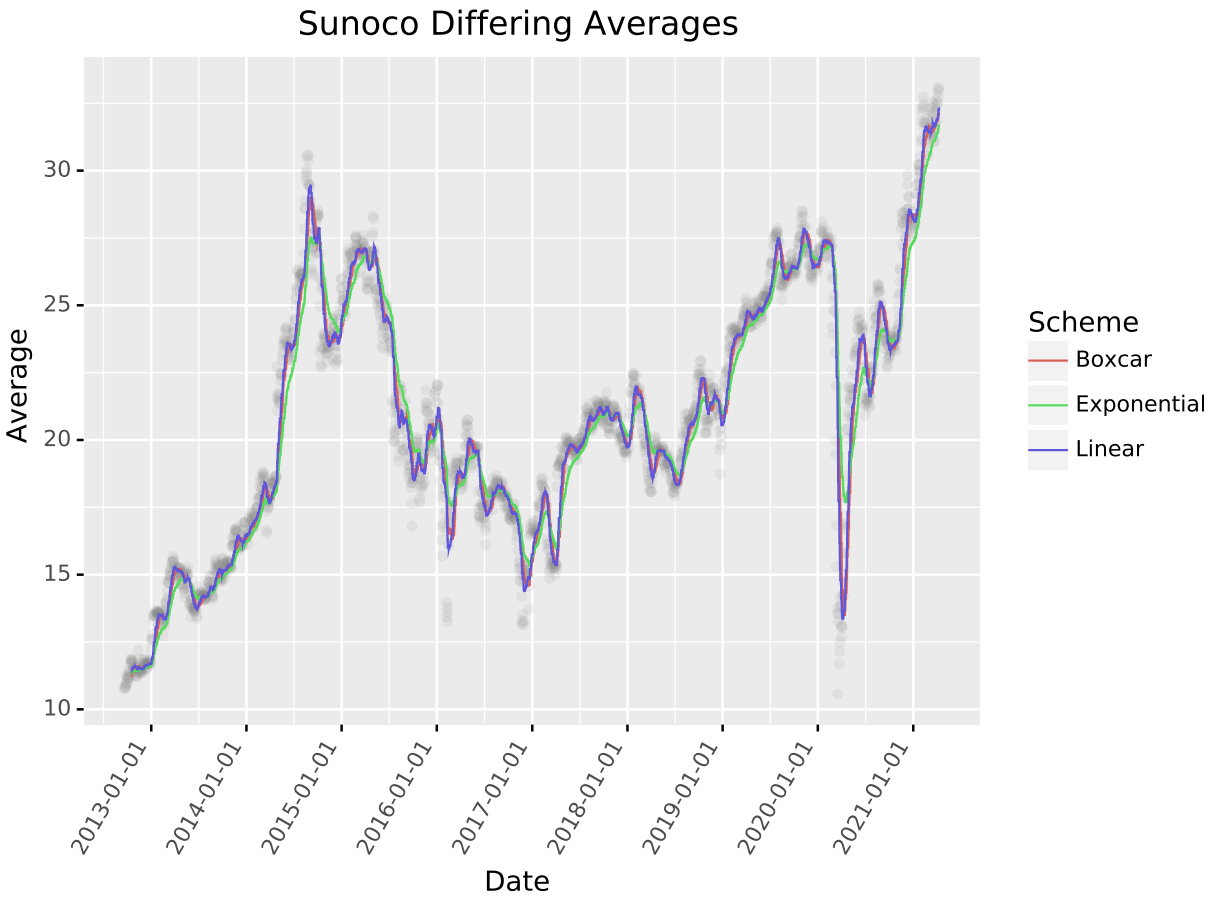

FIGURE 5. Little difference in weighting schemes on timescale much larger than the window

but on shorter time horizons close to the window size we more significant effects.

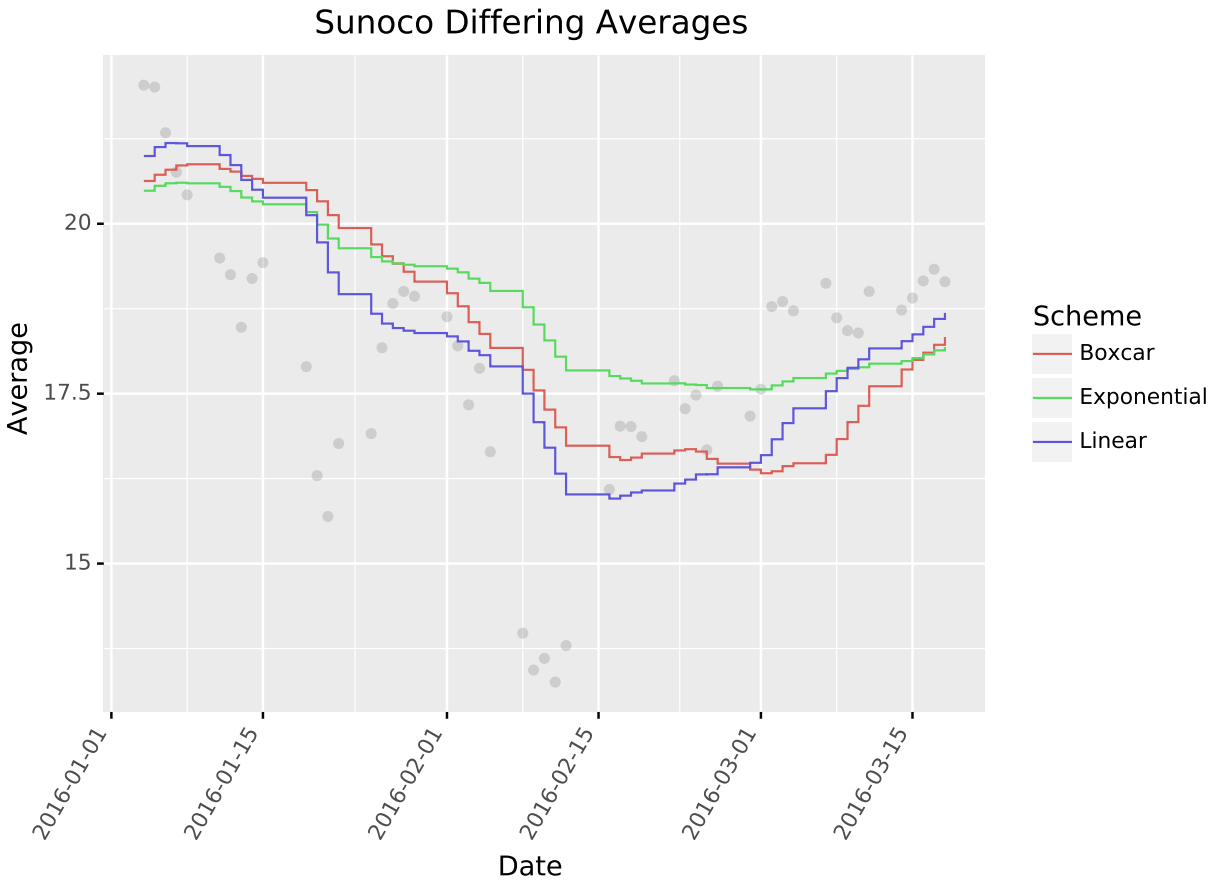

FIGURE 6. Significant difference in weighting schemes on timescale close to the window

1.1. **Comparison With Outliers.** As we can see, exponential and tent schemes, with their decaying weights, do not suffer from the jumping problem of boxcars.

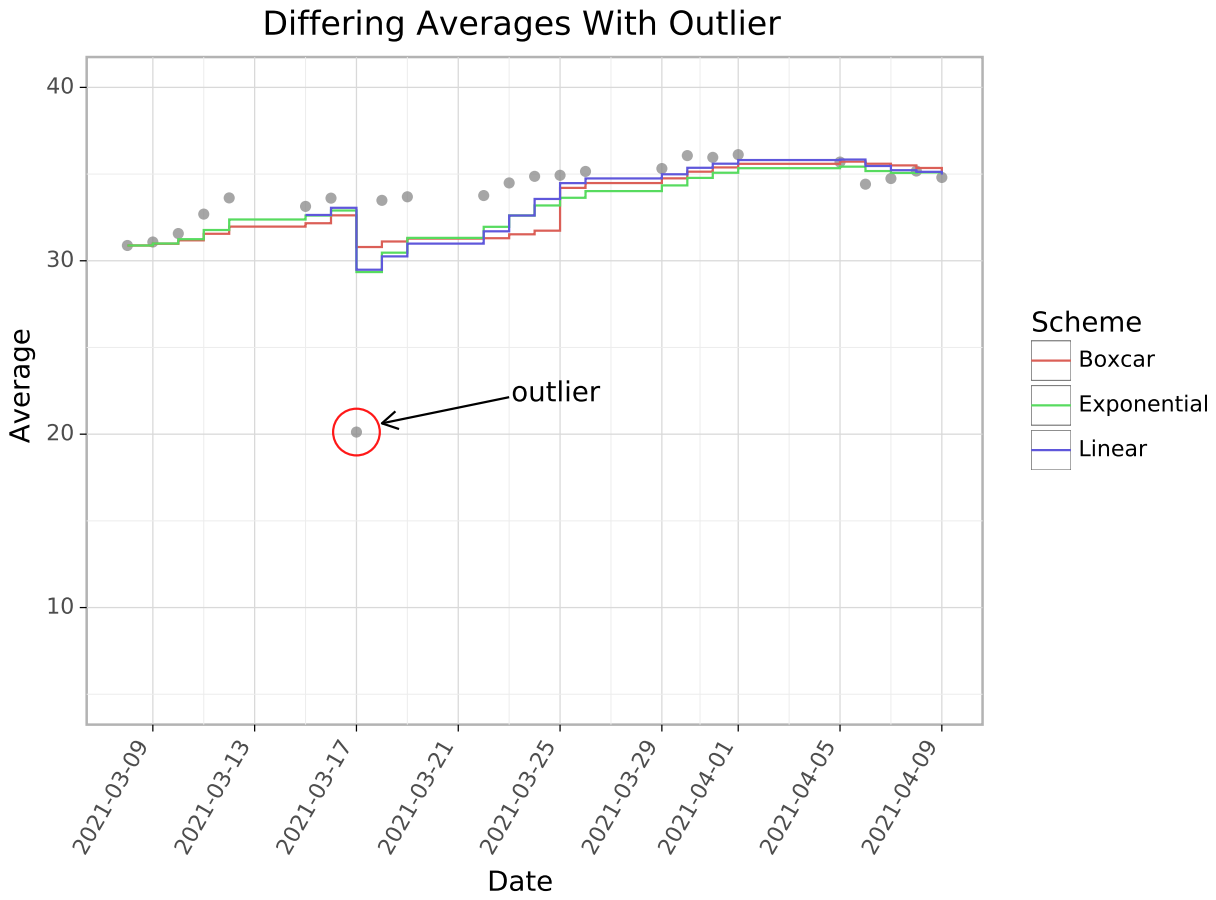

FIGURE 7. All three schemes react to the outlier, but only boxcar suffers a severe "bounce-back" effect

2. Efficient Algorithms

Among available weighting schemes, quantitative practitioners greatly favor the boxcar and the exponential. Boxcar is popular due to its simplicity of understanding and implementation. Exponential weighting is popular because it enjoys a particularly clean and efficient algorithm for computation. To begin with, note that each weight w_i can be computed as $w_i = \eta w_{i-1}$. Now, note that if we have already computed the old, previous average

$$A_{t_{j-1}} = (1-\eta)\sum_{i=0}^{\infty} \eta^i x_{j-1-i}.$$

then it can be rewritten as

$$A_{t_{j-1}} = (1-\eta)\frac{1}{\eta}\sum_{i=0}^{\infty} \eta^{i-1} x_{j-1-i} = (1-\eta)\frac{1}{\eta}\sum_{i=1}^{\infty} \eta^i x_{j-i}.$$

Our new average is

$$\begin{aligned} A_{t_j} &= (1-\eta)\sum_{i=0}^{\infty}\eta^i x_{j-i} \\ &= (1-\eta)x_j + (1-\eta)\sum_{i=1}^{\infty}\eta^i x_{j-i} \\ &= (1-\eta)(x_j) + \eta A_{t_{j-1}} \end{aligned}$$

Therefore, to update our average, we need to perform only a two multiplications and an addition. Furthermore, we need not store any arrays of values or intermediate results (unless we want to). We can discard $A_{t_{j-1}}$ the moment it has been used[1] to compute A_{t_j}. Thus, along with very few computations, updates to the exponential average or sum cost essentially no computer memory.

The implications for efficient computation almost cannot be overstated. Backtesting and parameter searches become far faster when computation is efficient.

We can contrast with, say, boxcar updating. Computationally it is not terribly expensive, but for a window size N we have to keep track of the last N values along with their sum S_i. Updating is then a matter of dropping the oldest item out of the list, updating S_i by subtracting its value, and then recomputing the quotient for the average[2].

2.1. **Infinite Lookback.** Since exponential decay has strictly positive weights, we might want to correct for the use of ∞ in our computations above. For example, if $N = 3$ and we take $A_1 = x_1$, then our "efficient" formula gives

$$A_3 = (1-\eta)(x_3) + \eta\left((1-\eta)(x_2) + \eta x_1\right)$$

which is not the same thing as the truncated geometric series averaging we might expect

$$\aleph_3 = \frac{\left(x_3 + \eta x_2 + \eta^2\right)}{1+\eta+\eta^2}$$

In cases of small sample counts, we may well want to adjust for this difference.

3. Time-Weighted Averaging

So far, we have covered exponential weighting suitable for regular time series, or other cases where the relative importance of data depends on observation count.

3.1. **Decay By Time.** Let us now consider the case where the importance of data declines with respect to "wall clock" time t. This is particularly relevant where data updates come in bursts. Here we choose a exponentially decaying weight, dependent on the age Δt of all former data

$$w = w_\lambda(\Delta t) = e^{-\lambda\cdot\Delta t}$$

where λ has units of 1/time, and our update formula becomes, with observation x_j at time t_j

$$A_{t_j} = e^{-\lambda\cdot\left(t_j - t_{j-1}\right)}A_{t_{j-1}} + \left(1 - e^{-\lambda\cdot\left(t_j - t_{j-1}\right)}\right)x_j$$

We refer to the quantity $1/\lambda$ as the *characteristic time* of our averaging, and the weight of data up to age $1/\lambda$ is of course $1/e$.

[1] From a stochastics point of view we could say that he entire operation is Markov.

[2] A nice interview question is: What are some ways to make a boxcar average as efficient as possible?

For regularly spaced data at intervals $\Delta t = \tau$, this is equivalent to our previous formulas with $\eta = e^{-\lambda \cdot \tau}$.

3.2. **Updates For Time-Weighted Averaging.** Our update algorithm is nearly identical, but now we need to track not only our previous average, but also the time t_{j-1} at which it was computed. This is still a trivial burden, and our calculation and memory efficiency remain essentially unaffected.

It is important to note that the time-weighted average will put very low weight on fresh data items if they come in rapid succession, since the decay of old data will be small. Whether or not this property is desirable depends on the specifics of the economic calculation in play.

4. Terminology In Exponential Weighting

If we consider either a discrete scheme with decay on old data of $c_\lambda = e^{-\lambda}$, or a time-weighted scheme with decay of size $e^{-\lambda \Delta t}$, there are a few equivalent ways in which these may be described:

- Characteristic Time
- Decay Coefficient λ
- Unit Decay η (which $= e^{-\lambda}$)
- Half-life
- Center Of Mass
- Span

All of these fully specify an exponential weighting scheme, so you will see various colleagues use these terms, according to their preferences.