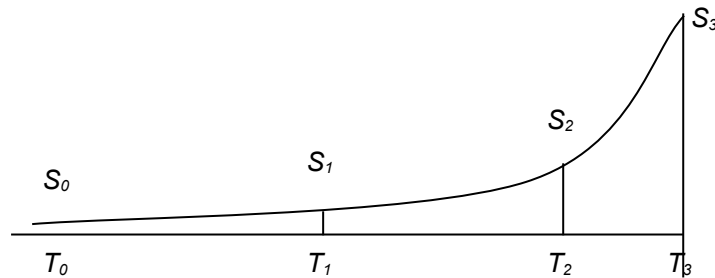


The Exponentially Decayed Window

This is an efficient technique of maintaining a statistic computed over a window of time on a data sequence. To maintain a moving window, so that say a derived variable average can be computed over this window, can involve a great deal of storage and housekeeping. To maintain a window of length N requires the storage of $2N$ items (the value and the time) and to manage the moving forward of the window requires a mechanism like a ring-buffer.

Another much more efficient approach is to use an exponentially decayed window as follows:



If say we want to simply sum the values of some derived variable over a window of time then if the current time is T_3 then we want values that are in the past to contribute increasingly little to any sum as they become more distant from the current time. We therefore require that the value V_2 at time T_2 (see graph above) contributes:

$$S_2 = V_2 e^{-k(T_3 - T_2)} \text{ where } k \text{ is a constant that determines the rate of decay.} \quad (1)$$

Equally we want older values to contribute less in a similar way.

To achieve this all we need to do is to take the current value V_3 and add it to a decay-adjusted last value S_2 as follows:

$$S_3 = V_3 + S_2 e^{-k(T_3 - T_2)} \text{ or more generally } S_j = V_j + S_{j-1} e^{-k(T_j - T_{j-1})} = V_j + S_{j-1} e^{-k\delta T_{j-1}} \quad (2)$$

By simply recursively applying this formula we will satisfy our requirement. To show how this works consider the following sequence where v is the input value at any step and a is the adjusted value.

$$\begin{aligned} a_0 &= v_0 \\ a_1 &= a_0 e^{-k\delta t_0} + v_1 \\ a_2 &= a_1 e^{-k\delta t_1} + v_2 \\ a_3 &= a_2 e^{-k\delta t_2} + v_3 \\ &\text{etc} \end{aligned}$$

By simple back substitution we can write the following:

$$\begin{aligned} a_2 &= (a_0 e^{-k\delta t_0} + v_1) e^{-k\delta t_1} + v_2 = v_0 e^{-k(\delta t_0 + \delta t_1)} + v_1 e^{-k\delta t_1} + v_2 \\ a_3 &= (a_1 e^{-k\delta t_1} + v_2) e^{-k\delta t_2} + v_3 = v_0 e^{-k(\delta t_0 + \delta t_1 + \delta t_2)} + v_1 e^{-k(\delta t_1 + \delta t_2)} + v_2 e^{-k\delta t_2} + v_3 \end{aligned}$$

This shows that the older contributions to the sum are effectively decayed, as we required, by the simple application of the recursive formula (2). This is fairly obvious but worth formalising.

As in generally we can write
$$a_n = \sum_{i=0}^n v_i e^{-k \sum_{j=i}^{n-1} \delta t_j} \quad (3)$$

then by induction we know that (2) is valid for arbitrary n .

1 Storage

This method only requires the storage of two variables:

1. the last adjusted value
2. the time

The simplicity of the calculation and the minimal storage requirement make it possible to implement complex window-based calculations very simply.

2 Half Life

The value of the constant k in the above equations can be interpreted in terms of a half-life if we set $k = -\ln 0.5 / h$ where h is the half-life.

3 Initial Rate Values

When computing rates over a normal window it is usual to require a minimum of two data points as otherwise there is no elapse-time to compute a rate. With an exponentially decaying window it is possible to include the first data point by assuming an elapse time of some multiple of the half-life.

The contribution of any data-point older than say (3 * half-life) is small (~0.12). Hence it is logical to compute an initial rate based on an elapse time of (3 * half-life).

This principal can be extend so that where there is a gap of greater than 3*half-life then we treat this like the beginning of new sequence.

The rate computed above is then profiled as a log-quantity in the usual way.