Accurately computing running variance

Adapted from http://www.johndcook.com/standard_deviation.html

The most direct way of computing sample variance or standard deviation can have severe numerical problems. Mathematically, sample variance can be computed as follows.

$$s^2 = \frac{1}{n(n-1)} \left(n \sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i \right)^2 \right)$$

The most obvious way to compute variance then would be to have two sums: one to accumulate the sum of the x's and another to accumulate the sums of the squares of the x's. If the x's are large and the differences between them small, direct evaluation of the equation above would require computing a small number as the difference of two large numbers, a red flag for numerical computing. The loss of precision can be so bad that the expression above evaluates to a *negative* number even though variance is always positive. See Comparing three methods of computing standard deviation for examples of just how bad the above formula can be.

There is a way to compute variance that is more accurate and is guaranteed to always give positive results. Furthermore, the method computes a running variance. That is, the method computes the variance as the x's arrive one at a time. The data do not need to be saved for a second pass.

This better way of computing variance goes back to a 1962 paper by B. P. Welford and is presented in Donald Knuth's Art of Computer Programming, Vol 2, page 232, 3rd edition. Although this solution has been known for decades, not enough people know about it. Most people are probably unaware that computing sample variance can be difficult until the first time they compute a standard deviation and get an exception for taking the square root of a negative number.

It is not obvious that the method is correct even in exact arithmetic. It's even less obvious that the method has superior numerical properties, but it does. The algorithm is as follows.

Initialize $M_1 = x_1$ and $S_1 = 0$.

For subsequent x's, use the recurrence formulas

$M_i = M_{i-1} + (x_i - M_{i-1})/i$
$S_i = S_{i-1} + (x_i - M_{i-1})*(x_i - M_i)$.

For $2 \le i \le n$, the ith estimate of the variance is $s^2 = S_i/(i - 1)$.

```
Pseudo code ( assumes 'i' starts at 1 )
If i==1 then
        Mean[i]=x[i];
        RunningSumVar[i]=0;
        Variance =0;
Else
        Mean[i]= Mean[i-1]+ (x[i]-Mean[i-1])/i
        RunningSumVar[i] = RunningSumVar[i-1] + (x[i]-Mean[i-1])* (x[i]-Mean[i])
        Variance = RunningSumVar[i]/(i-1)
```

The above pseudo-code shows how the running variance and running mean can be calculated. However, the code uses arrays for RunningSumVar and Mean which will use up a lot of memory. A more memory efficient set of pseudo code is illustrated below:

```
Pseudo code that uses less memory ( assumes 'i' starts at 1 )
If i==1 then
        Mean=x[i];
        RunningSumVar=0;
        Variance =0;
        //set up for next iteration
        Mean_old=Mean;
        RunningSumVar_old=RunningSumVar;

Else
        Mean= Mean_old+ (x[i]-Mean_old)/i
        RunningSumVar = RunningSumVar_old + (x[i]-Mean_old)* (x[i]-Mean)
        Variance = RunningSumVar/(i-1)
        //set up for next iteration
        Mean_old=Mean;
        RunningSumVar_old=RunningSumVar;
```

Here are a couple references on computing sample variance.

Chan, Tony F.; Golub, Gene H.; LeVeque, Randall J. (1983). Algorithms for Computing the Sample Variance: Analysis and Recommendations. The American Statistician 37, 242-247.

Ling, Robert F. (1974). Comparison of Several Algorithms for Computing Sample Means and Variances. Journal of the American Statistical Association, Vol. 69, No. 348, 859-866.