A Closer Look at Variance Implementations In Modern Database Systems

Niranjan Kamat Arnab Nandi
Computer Science & Engineering
The Ohio State University
{kamatn,arnab}@cse.osu.edu

ABSTRACT

Variance is a popular and often necessary component of aggregation queries. It is typically used as a secondary measure to ascertain statistical properties of the result such as its error. Yet, it is more expensive to compute than primary measures such as `SUM`, `MEAN`, and `COUNT`.

There exist numerous techniques to compute variance. While the definition of variance implies two passes over the data, other mathematical formulations lead to a single-pass computation. Some single-pass formulations, however, can suffer from severe precision loss, especially for large datasets.

In this paper, we study variance implementations in various real-world systems and find that major database systems such as PostgreSQL and most likely System X, a major commercial closed-source database, use a representation that is efficient, but suffers from floating point precision loss resulting from catastrophic cancellation. We review literature over the past five decades on variance calculation in both the statistics and database communities, and summarize recommendations on implementing variance functions in various settings, such as approximate query processing and large-scale distributed aggregation. Interestingly, we recommend using the mathematical formula for computing variance if two passes over the data are acceptable due to its precision, parallelizability, and surprisingly computation speed.

1. INTRODUCTION

New large-scale distributed data management and analytics systems are being developed at a rapid pace, with the scalability aspect of computation being their predominant development focus (excepting [10]). Comparatively lesser efforts have been expended on ensuring numerical correctness and stability of algorithms. While such an approach can result in the queries being answered more quickly, it can also cause the computation to have a higher level of numerical imprecision.

The concern of achieving numerical stability and precision is pertinent in numerous computational scenarios; it is especially important in variance calculation, which has an ubiquitous presence in large-scale analytics and is known to suffer from precision issues [4]. Variance is an important aggregate function and an essential tool in sampling-based aggregation queries. Typically used as a secondary measure, it augments measures such as `AVERAGE` and provides an insight into data distribution beyond the primary measure. Computation of variance, however, is susceptible to precision loss when the variance is much smaller than the mean [1].

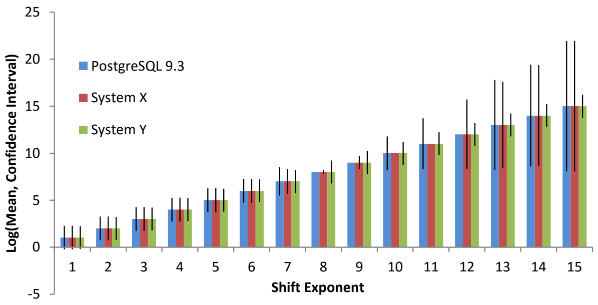


Figure 1: Effect of Variance Error on T-Test Confidence Intervals: As the magnitude of values increases (x-axis, true margin of error is kept consistent for each dataset), mean is expected to increase, and size of error bars is expected to stay the same. However, PostgreSQL and System X error bars ($\alpha = 0.05$) vary widely, while System Y has correct error bars (details in Section 1.1).

There exist several techniques to compute variance. The standard formula uses two passes and provides an accurate estimate (*Two Pass*). Due to its perception of being more expensive, other techniques have been developed that require a single pass over the data. One such formula, while fast, is known to suffer from precision loss (*Textbook One Pass*) due to *catastrophic cancellation* [4], an undesirable effect of a floating point operation that causes relative error to far exceed absolute error. Figure 1 demonstrates this problem. As a side note, this problem affects calculators as well [4].

Another formula (*Updating*), as recommended by Knuth [8], has found a strong foothold in the database community, with numerous implementations citing him. However, this formula is constrained by the fact that it can only incorporate a single value into the current running estimates. It is unable to combine the estimates from different subsets of data.

Given the rise of large-scale data processing, massive multi-core support and availability of GPUs, it is prudent to consider representations such as *Pairwise Updating*, that can combine partial results at a larger scale instead of incrementally incorporating a single data point. Further, *Pairwise Updating* is also known to provide better precision for both single ([1]) and double precision input (Section 4).

Contributions & Outline:

- We catalog usage of different variance formulas in various open source database systems (Table 2).
- We experiment with different closed source and open source databases to investigate precision loss issues. We find that precision of PostgreSQL and System X deteriorates the most. After looking at the PostgreSQL source code, we can verify that it uses *Textbook One Pass*, and hypothesize that System X does so as well.
- We empirically study the accuracy of the different representations under varying additive shifts and dataset sizes including a hitherto unstudied one, which we call *Total Variance*.
- We recommend using *Two Pass* if performing two passes over the data is acceptable (Section 5), which seems counter-intuitive, but works due to its computational simplicity.

In the next subsection, we look at the adverse effects of imprecise variance calculation. Section 2 presents different variance representations and their properties. We then note the representations used by modern databases in Section 3. Section 4 lists our analysis of the behavior of the different formulas (using double precision input compared with single precision in Chan et al. [1]). We conclude with our recommendations for choosing an appropriate variance representation in current environments.

1.1 Impact of Variance Calculations

Due to the pervasive use of variance, a loss of precision can have an impact in a variety of different domains. In the following paragraphs, we look at some use cases where the lack of precision in variance calculation can have adverse consequences.

Incorrect Output: It is possible to experimentally observe the loss of precision as incorrect output. To illustrate the pitfalls in using *Textbook One Pass*, 100 values were generated from a $Uniform(0, 1)$ distribution and shifted by $10^{\textit{Shift Exponent}}$ for *Shift Exponent* varying from 1 to 15. The variance as a result of data being shifted should be similar to the one without any shift. We verify this by adding and subtracting the shift exponent and note that the variance of the resultant dataset was close to the true sample variance. However, Figure 1, which depicts the sample mean and confidence interval, shows that PostgreSQL and System X suffer from variance calculations being susceptible to precision loss due to the shift. We know that PostgreSQL uses *Textbook One Pass* and the pattern of the erroneous calculations displayed by both hints towards System X using it as well. In contrast, other database systems suffered minor precision loss as expected (these results are not shown since they do not add any additional information to the figure). It should be noted that System Y was found to be highly immune to precision loss.

Visualization: Erroneous variance calculation can have a notable impact on visualizations as shown by Figure 1. While error bars should be similar, they instead vary widely and inaccurately for higher shift values for PostgreSQL and System X. We also found *Datavore*, which powers the *Profiler* visualization system [7], to use *Two Pass*.

Negative Variance: It is possible for variance to be negative while using *Textbook One Pass* – a theoretically impossible result. We observed in the PostgreSQL source code that variance is set to zero, if negative. Figure 1 shows numerous values of 0 (missing error bars) for PostgreSQL (shift exponent 8, 9, and 12) and System X (shift exponents 10 and 11), providing evidence of System X employing a similar strategy for handling negative variance values and using *Textbook One Pass*.

Decision Support Systems: As a building block in popular algorithms, flaws in variance implementations can have far-reaching impacts, e.g., in hypothesis testing, which is an integral part of decision support systems. Having imprecise or incorrect variance estimates can greatly change the result of hypothesis testing.

Data Mining: Variance is an important tool in statistical analysis and machine learning algorithms such as Gaussian Naive Bayes, or Mixture of Gaussians based algorithms such as background modeling, clustering, or topic modeling. For example, we found usage of *Textbook One Pass* within a graphics library of the *R* language. Similarly, MADlib [3] was also found to have a call to the PostgreSQL variance function: thus, an erroneous calculation of variance can extend from the underlying databases to the systems built on top of them.

Name	Formula	Accuracy	Passes	Storage	Parallel
Two Pass	$S = \sum_{i=1}^{N}(x_i - \bar{x})^2$, $\bar{x} = \frac{\sum_{x=1}^{N} x_i}{N}$	✓	2	O(1)	✓
Textbook 1 Pass	$S = \sum_{i=1}^{N} x_i^2$ - $\frac{1}{N}(\sum_{i=1}^{N} x_i)^2$	✗	1	O(1)	✓
Shifted 1 Pass	$S = \sum_{i=1}^{N}(x_i - \bar{x})^2 - (\sum_{i=1}^{N}(x_i - \bar{x}))^2/N$	Varies	1	O(1)	✓
Pairwise Updating	$T_{1,m+n} = T_{1,m} + T_{m+1,m+n}$, $S_{1,m+n} = S_{1,m}$ $+S_{m+1,m+n} + \frac{m}{n(m+n)}(\frac{n}{m}T_{1,m} - T_{m+1,m+n})^2$	✓	1	O(ln(N))	✓
Updating-YC	$T_{1,j} = T_{1,j-1} + x_j$ $S_{1,j} = S_{1,j-1} + \frac{1}{j(j-1)}(jx_j - T_{1,j})^2$	✓	1	O(1)	✗
Updating-WWH (Updating)	$M_{1,j} = M_{1,j-1} + \frac{x_j - M_{1,j-1}}{j}$, $S_{1,j} = S_{1,j-1}$ $+(j-1) \times (x_j - M_{1,j-1}) \times (\frac{x_j - M_{1,j-1}}{j})$	✓	1	O(1)	✗
Total Variance	$S = \sum_{i=1}^{groups} \left(n_i(m_i - \bar{x})^2 + (n_i - 1)v_i\right)$	✓	3	Varies	Varies

Table 1: Commonly used Formulas for Variance.

2. VARIOUS VARIANCE FORMULATIONS

Table 1 presents the common variance representations [1]. We use a similar naming convention to that used by Chan et al. [1]. S represents the sum of squares. The sample variance can be given by $\frac{S}{N-1}$, where N is the sample size. x_i is the i^{th} data point. $\bar{x}$ is the sample mean. $M_{m,n}$ is the mean of the data points from indexes m to n (both inclusive). $T_{m,n}$ is the total of the data points from indexes m to n (both inclusive). We have also provided *Total Variance* (derivation in the technical report [6]). In its formula, n_i, m_i, and v_i represent the count, mean, and variance respectively, of the i^{th} group.

Textbook One Pass can be computationally dangerous as the quantities $\sum_{i=1}^{N} x_i^2$ and $\frac{1}{N}(\sum_{i=1}^{N} x_i)^2$ can nearly cancel each other out. The *Pairwise Updating* formula hierarchically combines pairs of variance values and uses $O(log(N))$ storage while reducing the relative errors from $O(N)$ to $O(log(N))$ [1]. *Updating-YC* represents Youngs and Cramer formula [13] and is essentially identical to *Updating Pairwise* when $m = 1$ or $n = 1$. The *Updating-WWH* formula refers to the nearly identical formulas used by Welford et al. [11], West et al. [12], and Hanson et al. [2] and has similar precision as *Updating-YC*. We have used the *Updating-WWH* representation for updates using a single data point, and denote it by *Updating*. Shifting the data by an exact or approximate value of $\bar{x}$ (*Shifted One Pass*) can also result in substantial accuracy gains [1].

2.1 Total Variance

Since this is the first paper to introduce the *Total Variance* representation, we explain its steps in more details below. In the first pass, which is over the tuples, the variance (using one of the other formulas), mean, and count, of individual groups are computed. The second pass, over the groups thus formed, finds the overall mean of the data. In the third pass, over the groups, the overall variance is then found. Since the second and third passes are over the groups obtained as a result of the first pass, and different formulas can be used to compute variance of individual groups in the first pass, complexity of the overall algorithm can vary widely. While second and third passes are highly parallelizable, its overall parallelizability is dependent upon the formula used to find variance of the groups. It is designed for combining variances of different groups and is agnostic to the representation used in the first pass – our implementation uses *Updating*.

Computing mean of individual groups is a well-researched subject with Tian et al. [10] providing a good overview. We use a single pass algorithm to compute mean of individual groups and to combine means of groups as well. To handle a large number of groups, one can look into using an aggregation tree to combine means. The usual technique of mean estimation can be used in case the number of groups is large, at the cost of decreased precision.

There does not appear to be a theoretically ideal group size for *Total Variance*, and we could not determine one experimentally either [6]. In distributed execution, one natural way is to consider data across different nodes as groups. Further, data within a node can be equally partitioned, so that each core works on a single subgroup.

2.2 Properties of Different Representations

While Chan et al. [1] provide an overview of the accuracy, passes, and storage required for most of the formulas given in Table 1 (other than *Total Variance*), their classification as being distributive, and thus the ability to be parallelized, has not been explicitly listed before, which we do. In Table 1, the *Storage* column depicts the extra space needed for computing variance, which is above and beyond that needed to store the data itself.

The accuracy of *Shifted One Pass* depends on that of the mean estimate. *Pairwise Updating* is the only representation providing accurate results while being highly parallelizable and requiring a single pass. Additionally, as we will see in Section 4, the precision of *Total Variance* is slightly better than that of *Updating Pairwise*, which has the best precision amongst all single pass algorithms. As a side note, *Two Pass*, *Total Variance* and *Textbook One Pass* are the only representations that can be represented using a standard SQL query. Note that Table 2.1 of [1] succinctly enumerates the error bounds of different formulations. Further, Kahan summation [5, 10] can help improve their precision.

2.3 Data Conditioning

Data shifting and scaling are immensely useful in improving accuracy of algorithms [4]. For example, shifting the data by its mean is the basis for *Shifted One Pass*. Indeed, Chan et al. [1] demonstrate the usefulness of shifting by an approximate mean computed using a sample of the data by proving that it reduces the bounds of the condition number. Further, techniques such as dividing by the mean or using the log function [4] can be helpful in improving the accuracy. However, along with requiring additional computational resources these techniques can also worsen the accuracy under malicious datasets [1], and need careful user supervision.

2.4 Hybrid Formulae

It is clear that different implementations can be used to find variance of different groups, and combine partial results. Indeed, it has been brought to our attention that a commercial system uses the *Updating-YC* formula to compute variance at individual nodes, and combines them using *Pairwise Updating* formula. *Total Variance* is a hybrid formula as well. This provokes an interesting piece of future work – choosing different representations at different computation steps, based on factors such as numerical precision, data partitioning, time for first result, number of passes permissible (Section 5). This idea is elaborated upon in Section 5.

2.5 Current Recommendation Guidelines

Chan et al. [1] provide detailed recommendation guidelines for different variance formulas. They recommend usage of *Pairwise Updating* for combining variances across multiple processors since it reduces the errors and is massively parallelizable if extra $O(log(N))$ space is available. Further, it is also the safest (least precision loss) algorithm to use within each processor, under the constraint of a single pass.

2.6 Extensibility to Other Measures

Standard deviation, standard error, and coefficient of variation are important statistical measures, and perform variance computation. Thereby, they are also affected by the properties of the underlying representation. Similarly, the properties will also extend to any user-defined measure whose variance can be expressed in a closed form as a function of the variance of one of the measure dimensions. For example, for a user-defined measure given by $a *$ `AVG`$(agg) + b$, where a and b are constants and agg is a measure dimension, the variance of the measure can be given in closed form as a^2*`VARIANCE`(agg).

3. VARIANCE IMPLEMENTATIONS IN MODERN DATABASE SYSTEMS

We looked at the code of multiple open source databases to find their variance representations. We also conjecture about two closed source ones through our experiments.

Database	Formula
PostgreSQL 9.4.4	Textbook One Pass
MySQL 5.7	Updating
Impala 2.1.5	Updating Pairwise
Hive 1.2.1	Updating Pairwise
Spark 1.4.1	Updating Pairwise
SQLite	No Variance Support
System X	Textbook One-pass *(Conjecture)*
System Y	Cannot Conjecture

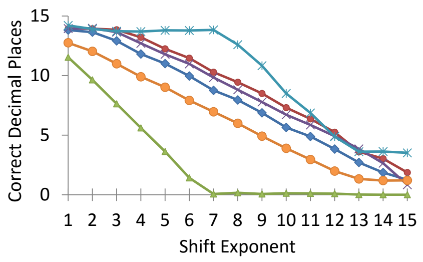
Table 2: Variance Implementations in Databases.

PostgreSQL uses *Textbook One Pass* and is thus susceptible to precision loss. MySQL uses Knuth's modification [8] of Welford's updating formula. Therefore, it can only process a single additional data point, and cannot avail of the possible parallelization. Spark 1.4.1 and Impala 2.1.5, on the other hand, use a modified version of *Updating Pairwise.*

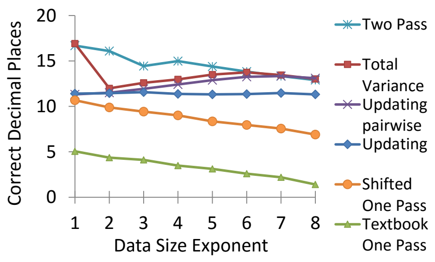
Although the source code for System X is not available, we conjecture that it uses *Textbook One Pass* as its precision behavior was similar to that of PostgreSQL. System Y was found to have the best precision. We hypothesize that it uses higher precision variables, but cannot make any conjecture about the exact representation.

4. EXPERIMENTAL ANALYSIS

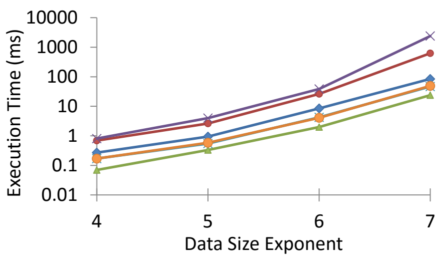
Chan et al. [1] have looked at the precision of different algorithms using single precision input. We present the precision results using double precision input. We also evaluate the precision of *Total Variance.* We look at the precision in the variance cal-



(a) With increasing shift exponent, all representations experience precision loss, though some more severely than others.



(b) Precision generally decreases with increasing dataset size.



(c) *Two Pass* provides results faster than others, excepting *Textbook One Pass*, which has the least numerical precision.

Figure 2: *Two Pass* not only has the highest precision, but also requires second lowest execution time.

culation offered by the different databases. We also present the execution times of different algorithms on data sizes up to 100 million tuples. The results are the average over 100 runs. Experiments were performed using Ubuntu 14.04.05 LTS with a 4 core, 2.4 GHz Intel CPU, with 16 GB RAM, and 256 GB SSD storage, using a single execution thread.

Dataset: Although numerous benchmarks exist to evaluate the accuracy of numerical algorithms, they are constrained by their dataset size. For example, the biggest dataset in the NIST StRD [9] benchmark consists of 5000 points. Furthermore, for this dataset, the mean is not significantly larger than the standard deviation ($\mu = 4.5348$, $\sigma = 2.8673$). Therefore, in a similar vein as Tian et al. [10], we created synthetic datasets of different sizes using $Uniform(0,1)$ (variance being $\frac{1}{12}$). They were shifted by adding values ranging from 10^1 to 10^{15}.

4.1 Impact of Shift

Numerical precision was evaluated using varying additive shifts, over a dataset of size 10000. Group size was set at 10 for *Total Variance*. We present our findings in Figure 2a, where Y-axis represents the number of correct decimal digits (non-fractional part of the result was 0). The results were as expected [1], with *Two Pass* having the best precision, and *Textbook One Pass* the worst.

4.2 Impact of Data Size

Since precision errors typically accumulate, we tried datasets of sizes from 10 to 100 million. The shift was set at 10^5. Figure 2b shows that precision generally worsens with increasing data size. *Two Pass* again outperforms other algorithms. *Textbook One Pass* consistently exhibits the worst precision.

Counter-intuitively, the precision of *Total Variance* and *Updating Pairwise* was found to increase for the data size exponents from 2 to 6. We are unable to conjecture the reason behind this behavior. The precision error for *Updating Pairwise* increases as $O(log(n))$, while that for others (except *Total Variance*) increases as at least $O(n)$ [1], where n is the data size. Therefore, while we can expect the error in *Updating Pairwise* to not increase at the same rate as others, the error decrease is unexpected. In the absence of theoretical error bounds for *Total Variance*, we cannot hypothesize about the possible cause. To ensure there were no irregularities, the experiment was repeated multiple times with similar results.

4.3 Single-Threaded Execution Speed

We also looked at the execution time of different algorithms with increasing data size (Figure 2c). Results with lower data sizes have not been presented due to the computation taking minimal time. Surprisingly, there was no discernible difference in execution time between *Two Pass* and *Shifted One Pass*. Only *Textbook One Pass* took lesser time than *Two Pass*. We attribute the low execution time of *Two Pass* to simplicity of its computation.

4.4 Impact of Shift on Different Databases

We look at variance precision for the different databases under varying additive shifts. We took efforts to ensure different systems have similar data types. 100 points were chosen from a $Uniform(0,1)$ distribution. Figure 3 shows that precision loss follows a similar pattern in System X and PostgreSQL. Impala and MySQL have a similar error profile as well.

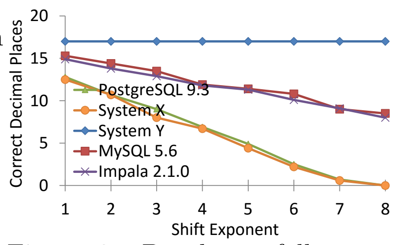


Figure 3: Databases follow expected precision patterns.

4.5 Miscellaneous Experiments

In one of the other experiments, details in [6], we noted that changing group size in *Total Variance* did not have a significant effect on the precision. In another experiment, multi-threaded execution gave us expected speedups for the parallelizable formulations. Finally, we inspected the mantissa of the two terms that compose *Textbook One Pass* to demonstrate the cause of catastrophic cancellation.

5. CONCLUSION & RECOMMENDATIONS

Precision issues associated with *Textbook One Pass* have been well documented. However, we have seen that databases such as PostgeSQL and likely System X still use it. We recommend from the perspective of safety to discontinue its usage. Though there might be arguments for its continued usage after warning the users in certain scenarios, the arguments against it far outweigh the speedup benefit and its ease of implementation. Although error inherently exists in approximate query processing, numerical precision errors are easy to eliminate and hard to apportion and therefore should be avoided whenever possible. Hence, we recommend to the designers of databases, and statistics and analytics packages, to discontinue its usage. Further, it would be wise for users to perform a sanity check using experiments similar to those given in Section 4.1.

Previous work has recommended *Pairwise Updating* from the perspective of precision, speed, and parallelizability [1]. However, we have seen from our experiments of up to 100 million data points, that the most accurate algorithm, *Two Pass*, takes lesser time than *Updating*, *Updating Pairwise*, and *Total Variance*. Further, it takes around the same amount of time as *Shifted One Pass*, which relies on mean estimation. *Two Pass* is also easy to implement and parallelize. Therefore, in the case that **performing two passes over the data is acceptable, *Two Pass* should be the preferred algorithm**. Determining whether two passes are acceptable, however, is a nuanced decision. When the data fits in memory, performing two passes over the data is clearly acceptable as all representations will incur the identical data read I/O cost. When the data cannot fit in memory, summing up the estimated I/O and computation times can help determine whether *Two Pass* will need the least amount of time, in which case it should be chosen.

In other cases, i.e., whenever *Two Pass* is not estimated to require the least execution time, there does not exist a clear winner, due to different algorithms having different strengths and weaknesses. *Updating* provides faster results at lower precision, compared with *Updating Pairwise*, without needing additional memory. *Updating Pairwise* is parallelizable, whereas *Updating* is not. While *Shifted One Pass* provides quick results, its accuracy is dependent on correctness of the mean estimate. *Total Variance* has good accuracy, although it takes longer to execute, and is dependent on the algorithm used to compute group statistics, while also needing multiple passes. Hence, there does not exist any algorithm that dominates every other algorithm, resulting in there not being a clear choice. We can see that a query planner that devises hybrid formulas, while taking the data distribution, estimated I/O and computation costs, and the overall strengths and weaknesses of different algorithms into consideration, appears to be an important and ideal piece of future work.

6. ACKNOWLEDGMENT

We acknowledge the generous support of U.S. NSF under awards IIS-1422977 and CAREER IIS-1453582. We would also like to thank the reviewer for their highly insightful and helpful comments.

7. REFERENCES

[1] T. Chan et al. Algorithms for Computing the Sample Variance: Analysis and Recommendations. *Am. Stat.*, 1983.

[2] R. J. Hanson. Stably Updating Mean and Standard Deviation of Data. *ACM*, 1975.

[3] J. M. Hellerstein, C. Ré, F. Schoppmann, et al. The MADlib Analytics Library: or MAD Skills, the SQL. *VLDB*, 2012.

[4] N. J. Higham. Accuracy and Stability of Numerical Algorithms. *SIAM*, 2002.

[5] W. Kahan. Further Remarks on Reducing Truncation Errors. *ACM*, 8(1):40, 1965.

[6] N. Kamat and A. Nandi. A Closer Look at Variance Implementations In Modern Database Systems. *Arxiv TR*, 2016.

[7] S. Kandel et al. Profiler: Integrated Statistical Analysis and Visualization for Data Quality Assessment. *AVI*, 2012.

[8] D. E. Knuth. *Art of Computer Programming, Volume 2: Seminumerical Algorithms.* 2014.

[9] J. Rogers, J. Filliben, et al. StRD: Statistical Reference Datasets for Testing the Numerical Accuracy of Statistical Software, 1998.

[10] Y. Tian, S. Tatikonda, and B. Reinwald. Scalable and Numerically Stable Descriptive Statistics in SystemML. *ICDE*, 2012.

[11] B. Welford. Note on a Method for Calculating Corrected Sums of Squares and Products. *Technometrics*, 1962.

[12] D. West. Updating Mean and Variance Estimates: An Improved Method. 1979.

[13] E. A. Youngs et al. Some Results Relevant to Choice of Sum and Sum-of-Product Algorithms. *Technometrics*, 1971.