Wilkinson's Tests and SQL Packages

B. D. McCullough
Decision Sciences & MIS
Drexel University
Philadelphia, PA USA
bdmccullough@drexel.edu

Taha Mokfi
Independent Researcher
Connecticut, USA
mokfi.taha@gmail.com

Mahsa Almaeenjad
Independent Researcher
Connecticut, USA
almaee.mahsa@gmail.com

ABSTRACT

Wilkinson's Tests are used to benchmark the accuracy of some statistical functions in six SQL packages: Apache Hive, Microsoft Access, Microsoft SQL Server, MySQL, Oracle 11g SQL, and SAP Hana. Using the best choice of data type, we find that different packages use different rounding schemes, two packages use unreliable algorithms to compute the sample variance, one package returns the population standard deviation when the sample standard deviation is called, and one package has an unstable algorithm for computing the correlation coefficient. Using the wrong data type all but guarantees inaccurate results.

1. INTRODUCTION

A database is an organized collection of data. A database is managed by a database management system (DBMS). The popularity of DBMSs increased with the advent of relational database management systems (RDBMS). Query operations on tables were introduced through the Structured Query Language (SQL). Even though the SQL syntax was standardized, many vendors developed SQLs on their own. Hence, some of the standard functionality might not be the same and the underlying code might also make a difference. This is especially important since many authors advocate the use of SQL for data mining activities: Wei et al [12], Trueblood and Lovett [20], Linoff [10], Alexander [1], Celko [4], Fotache and Strimbel [7], Ordonez and Pitchaimalai [16], and Pearson and Mackey [17]. None of these authors warns the reader that SQL software might be inaccurate for statistical purposes.

In particular, different SQL packages might give different answers to the same problem. Indeed, the situation is even more problematic: the same package can give different answers to the same problem! Recently Niranjan and Nandi [15] found errors in various SQL packages in the implementation of the calculation of sample variance; not all packages could accurately compute the sample variance. We extend their work with an eye toward elementary statistical calculations using a collection of benchmark tests known as "Wilkinson's Tests" based on Wilkinson's [21] *Statistics Quiz* that presents six suites of tests: reading an ASCII file; real numbers; missing data; regression; analysis of variance; and operating on a database. Wilkinson's Tests have a long track record of being applied to statistical packages by, among others, Wilkinson [22], Sawitzki [18], Bankhofer and Hilbert [3], McCullough [13], Choi and Kiefer[6], Lomax [11], Keeling and Pavur [8], and McCullough and Yalta [14].

While *Statistics Quiz* offers six suites of tests, for present purposes only the "Real Numbers," "Missing Data," and "Regression" suites are relevant. All the applicable tests are applied to the most recent versions of RDBMS packages. We have tested:

- Hive version 1.2.1
- MS Access 2016 v16.0 (64 bit)
- Microsoft SQL Server Management Studio v12.0 (henceforth, MS SQL)
- MySQL workbench v6.3.6 buit 511 CE (64 bit)
- Oracle Database Express edition 11g
- SAP Hana version 2.0 [Express Edition]

All programs were run on a laptop wtih Intel(R) Core(TM) i5-5200U CPU 2.2 RAM: 8 GB System with a 64 bit Windows 10 Operating System.

To date, we were unable to find any research assessing the accuracy of statistical functionality in SQL packages. This is important, because SQL is frequently advocated for both statistical analysis of data and data mining:

2. THE DATA

Table 1 below displays the data set "Nasty". The values for BIG are about the size of the population of Egypt, while the values of HUGE are the same order of magnitude as the American federal budget deficit.

LABEL$	X	ZERO	MISS	BIG
ONE	1	0	.	99999991
TWO	2	0	.	99999992
THREE	3	0	.	99999993
FOUR	4	0	.	99999994
FIVE	5	0	.	99999995
SIX	6	0	.	99999996
SEVEN	7	0	.	99999997
EIGHT	8	0	.	99999998
NINE	9	0	.	99999999

LITTLE	HUGE	TINY	ROUND
0.99999991	1.0E12	1.0E-12	0.5
0.99999992	2.0E12	2.0E-12	1.5
0.99999993	3.0E12	3.0E-12	2.5
0.99999994	4.0E12	4.0E-12	3.5
0.99999995	5.0E12	5.0E-12	4.5
0.99999996	6.0E12	6.0E-12	5.5
0.99999997	7.0E12	7.0E-12	6.5
0.99999998	8.0E12	8.0E-12	7.5
0.99999999	9.0E12	9.0E-12	8.5

Table 1: Data Set NASTY.DAT

Immediately we encountered difficulties reading the data. For SQL and MySQL we tried to import the data from a .csv file but both packages read all the columns as character when they should be numeric, so we manually changed the type of all columns to be double precision. In contrast, Oracle could import data from csv with numeric types. In general, after reading in the data for each package, we had to set the type for each column. This turned out to be a critical step. One might think that NUMERIC, BINARY_FLOAT and BINARY_DOUBLE would give similar answers, but such is not the case. For example, when computing the standard deviation of the variables, to four decimals all answers should be 2.7386 raised to some power of ten, with the exceptions of ZERO which should be 0 and MISS which should be missing. Yet, Table 2 shows some results from Oracle for various data types.

Observe that the first column, NUMERIC, gives correct answers. In contrast, BINARY_FLOAT and BINARY_DOUBLE correctly compute the sample standard deviation for X, HUGE, TINY and ROUND, but fail for other variables. Oracle was not alone in exhibiting this type of behavior. Specifically why Oracle and other packages erratically compute the standard deviation for the BINARY_FLOAT and BINARY_DOUBLE types is beyond the scope of this paper, but the point is that specifying the best data type is critical. Therefore we ran all the tests for all the data types for all the packages and only used the best choice.

variable	NUMERIC	BINARY_ FLOAT	BINARY_ DOUBLE
X	2.7386	2.7386	2.7386
ZERO	0	0	0
MISS	–	–	–
BIG	2.7386	0	2.4494
LITTLE	2.7386E-08	0	1.4901 E-08
HUGE	2.7386E12	2.7386E+12	2.7386E12
TINY	2.7386E-12	2.7386E-12	2.7386E-12
ROUND	2.7386	2.7386	2.7386

Table 2: Oracle Std. Dev. Results for Various Data Types

The failures of MS SQL are presented in Table 3. None of the types provides correct answers for all the variables. See that the standard deviation of LITTLE can take on three different values, none of which is correct. REAL gives the same answers as SINGLE, and so is omitted from the table.

variable	NUMERIC	DECIMAL	SINGLE	DOUBLE
X	2.7386	2.7386	2.7386	2.7386
ZERO	0	0	0	0
MISS	NULL	NULL	NULL	NULL
BIG	2.4495	2.4495	4.2426	2.4495
LITTLE	0	2.9802E-8	3.6500E-8	2.9802E-8
HUGE	2.7386E12	2.7386E12	2.7386E12	2.7386E12
TINY	0	2.7386E-12	2.7386E-12	2.7386E-12
ROUND	2.7386	2.7386	2.7386	2.7386

Table 3: MS SQL Std. Dev. Results for Various Data Types

MS SQL server supports different numeric data types. The type DECIMAL has fixed precision and scale and takes two arguments: precision (Maximum total number of decimal digits which is between 1 to 38) and scale (The number of decimal digits that will be stored to the right of the decimal point which is between 0 and precision). FLOAT and DOUBLE types are approximate and have only one argument n (precision and storage size which can be between 1 and 53). A precision from 0 to 23 results in a 4-byte single-precision FLOAT column. A precision from 24 to 53 results in an 8-byte double-precision DOUBLE column. All the other types such as: INT, BIGINT, SMALLINT, and TINYINT are exact-number data types that use integer data depending on the data points. For MS SQL, DECIMAL is specified as DECIMAL(38,12), SINGLE as FLOAT(24), and DOUBLE as FLOAT(53). In MySQL, NUMERIC is implemented as DECIMAL, so the following remarks about DECIMAL apply equally to NUMERIC.

For the six packages, the options and our best choices for data type are given in Table 4 (we exclude integer types), where an asterisk denotes that the data type could not correctly compute the standard deviation. The prevalence of asterisks shows that whether a package can correctly compute the standard deviation depends on the data type specified.

package	variable types	most accurate
Hive	FLOAT*, DOUBLE, DECIMAL	DOUBLE
Access	SINGLE*, DOUBLE*	DOUBLE
MS SQL	NUMERIC*, DECIMAL*, SINGLE*, DOUBLE*, REAL*	FLOAT(53)
MySQL	NUMERIC*, DECIMAL, FLOAT*, DOUBLE	DOUBLE
Oracle	NUMERIC, BINARY_FLOAT*, BINARY_DOUBLE*	NUMERIC
Hana	FLOAT, REAL*, DOUBLE, DECIMAL	DOUBLE or FLOAT

Table 4: Options and best data type for each package

3. THE TESTS

3.1 Real Numbers

3.1.1 Test II-A: Print ROUND to one digit.

The program should follow IEEE-754, which specifies "round to even", so the correct answer is $0, 2, 2, 4, 4, 6, 6, 8, 8$. For all packages, we used the command: Round(X,1) and results are presented in Table 5.

package	result	package	result
Apache Hive	fail	MySQL	pass
MS Access	fail	Oracle 11g	fail
MS SQL	fail	SAP Hana	fail

Table 5: Results of Test II-A: print ROUND

Only MySQL passes this test. All other packages return $1, 2, 3, 4, 5, 6, 7, 8,$ 9 instead of the correct answer, which indicates that they are not using IEEE-754 rounding. We see again that different SQL packages will return different answers to the same problem.

For an example of why correct rounding is important, consider this example from an SQL discussion board [2]:

> I am currently working on POC for migrating Oracle to SQL Server 2008 R2. Stuck with an issue related to rounding off calculations.
>
> There is this "rate" value we are calculating using POWER functions (it has nested POWER functions actually). I am getting mismatches because the values are not correctly round off during calculation. There are around 258 rows where the values are 0.01 less than the required value (ex. I get the rate value as 6.31 where actually it should be 6.32).
>
> In order to correct this issue, I changed the data types of the fields in the calculation from FLOAT to NUMERIC to correct out the precision. This is able to resolve the issue with the above 258 rows, but now I have another 61 rows where the value is 0.01 more than the required value (I get the rate value as 7.42 where actually it should be 7.41).
>
> All the fields in the Oracle calculation is using NUMBER datatype (without any precision), so cant really figure out what equivalent data type should be used in SQL Server. Currently I am using FLOAT datatype.
>
> Please advise if you have any pointers.

With different SQL packages using different rounding schemes, migrations are needlessly complicated and can produce unintended errors.

3.1.2 Test II-B: Plot HUGE against TINY and plot BIG against LITTLE in a scatter plot.

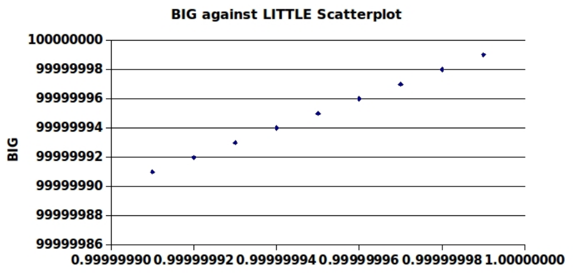

Figure 1: Test II-B Results for MS Access.

For each case the answer is a 45-degree straight line. MS ACCESS is the only package offering plotting capability. As Figure 1 shows, MS ACCESS passes this test; the other two graphs are correct and are omitted to conserve space.

	MS ACCESS
HUGE v. TINY	pass
BIG v. LITTLE	pass
X v. ZERO	pass

Table 6: Results of Test II-B

3.1.3 Test II-C: Calculate the mean and standard deviation of each variable.

The mean should be equal to the fifth value of each variable. Each package passes this test.

Standard deviations should be "undefined" or missing for MISS, zero for ZERO, and 2.738612788 (times 10 to some power) for all other variables (in the table the powers of ten are omitted). The relevant command for each package is: STDDEV_SAMP. The standard deviation results by different packages are listed in Table 7.

	Hive	ACCESS	MS SQL	MySQL	Oracle	Hana
X	2.582	p	p	p	p	p
ZERO	p	p	p	p	p	p
MISS	p	p	p	p	p	p
BIG	p	2.309	2.450	p	p	p
LITTLE	p	2.810	2.980	p	p	p
HUGE	p	p	p	p	p	p
TINY	p	p	p	p	p	p
ROUND	p	p	p	p	p	p

Table 7: Results of Test II-C – calculate the standard deviation (correct answers indicated by 'p' for 'pass')

Accurately computing the sample standard deviation is a solved problem, and there is no excuse for any software developer to use a bad algorithm. For a discussion of the various algorithms, see Ling [9] or Chan, Golub and Leveque [5]. Of course, if one uses the command STDDEV, one gets different answers depending on the package. STDDEV is supposed to be an alias for STDDEV_SAMP, but Hive returns 2.5820: in Hive, STDDEV is an alias for STDDEV_POP, so the documentation for STDDEV is wrong. Observe that MS SQL and MS Access both are using bad algorithms for computing the sample standard deviation, but apparently different bad algorithms! The same software company appears to be using two different algorithms, and neither of them is correct.

Consider performing a one-sample test of means on 29 observations. Let $H_0 : \mu = 11$ against $H_1 : \mu \neq 11$. Suppose the correct standard deviation is 2.7386; then the calculated t-statistic is 1.966 and the null hypothesis is just barely rejected. If the standard deviation were incorrectly calculated as 2.980 then the calculated t-statistic would be 1.81 and the null hypothesis would not be rejected. If the standard deviation were incorrectly calculated as 2.309 and many would incorrectly conclude that the null hypothesis had been even more strongly rejected.

	X	ZERO	BIG	LITTLE	HUGE	TINY	ROUND
X	1	NA	1	1	1	1	1
ZERO	NA	NA	NA	NA	NA	NA	NA
BIG	1	NA	1	1	1	1	1
LITTLE	1	NA	1	1	1	1	1
HUGE	1	NA	1	1	1	1	1
TINY	1	NA	1	1	1	1	1
ROUND	1	NA	1	1	1	1	1

Table 8: Correct answer for II-D, Correlation Matrix

3.1.4 Test II-D: Compute the correlation between all the variables.

The correlation coefficient is calculated as

$$\rho_{wz} = \frac{cov(w, z)}{\sigma_w \sigma_z} \qquad (1)$$

where $cov(w, z)$ is the covariance between w and z and and σ_w is the standard deviation of w. Since the standard deviation of ZERO is zero, Equation 1 has zero in the denominator and so its correlation with any other variable is undefined. Only Oracle and Hive offer a correlation function. The correct answer is given in the top of Table 8.

For the data type FLOAT, in addition to sometimes giving the correct answer of 1.0, running HIVE with FLOAT also produces answers 0.614, 0.836 and 0.866 instead of 1.0 for some pairs of variables. For the preferred data type DOUBLE, correlating ROUND with X or ROUND produces NaN instead of the correct answer of 1.0.

Oracle, using the preferred data type NUMERIC, correctly calculates the correlation matrix. However, when using either BINARY_FLOAT or BINARY_DOUBLE it gives wildly incorrect answers.

The user-guides for these packages should warn that accuracy depends on choosing the correct data type, as the packages will not warn the user if a bad data type has been chosen.

3.1.5 Test II-E: Plot X against X.

In this test we try to plot the same variable with itself. The answer should be a graph with a 45 degree line.

Only MS Access lets us make a plot but it lets us select a variable only once and we cannot select it again. Therefore, we are unable to make this graph.

3.1.6 *Test II-F: Regress BIG on X.*

If the constant intercept is 99999990, then we can expect the slope to be 1. Only Oracle 11g offers regression, and it gives the correct answer. Oracle passes.

3.2 Missing Data

Missing values have been encountered commonly in all the areas. It could be due to bad data entry or simply no reading at all at that respective time. SQL queries do handle NULL or missing values quite efficiently and it might simply exclude the readings or include them in the results of queries. We however define handling missing values and simply excluding them in a different way.

3.2.1 *Test III-A: MISSING in a conditional statement.*

Implement the below pseudo-code in the package:

```
IF MISS = 3
THEN TEST = 1
ELSE TEST = 2
```

Ideally, the correct result should be equal to 2, since the MISS values do not have any value and definitely do not equal to 3. We use a select statement with the below syntax:

package	command
Hive	if(Miss = 3,1,2)
SQL/MySQL	Iif(Miss = 3,1,2)
Oracle	CASE WHEN Miss = 3 THEN 1 ELSE 2 END
MS Access	IIf(Miss=3,1,2)
SAP Hana	CASE WHEN Miss = 3 THEN 1 ELSE 2 END

All packages pass this test. In cases, where an IF..ELSE.. statement could not be used, we could use a similar functional statement. For example, we have used NVL2 function to test the IF.ELSE loop since it offers a similar functionality. All packages pass.

3.2.2 *Test III-B: Is MISSING summable?*

We apply the following test in this section

```
IF MISS = <missing> THEN MISS = MISS + 1
```

Ideally, the correct answer is missing, since we cannot add to any NULL value. The specific command used for each package is given in Table 9.

MS Access does not return a missing value indicator, but simply an empty table. This is an error.

package	command
SQL/MySQL	isnull(x, x+1)
Oracle	NVL(x, x+1)
MS Access	IIf(IsNull([miss]),[MISS]+1)
Hive	if(isnull(Miss), Miss +1, Miss)
Hana	ifnull(MISS,MISS+1)

Table 9: Commands for IF MISS... test

Hive	MS Access	MS SQL	MySQL	Oracle	Hana
pass	fail	pass	pass	pass	pass

Table 10: Results of Test III-B

3.3 Regression

Another very common computation is the computation of regression models. These models are useful in many ways and form the basis of data analysis. We can calculate the regression models in different ways by using the formula. Only Oracle offers functions to calculate the regression values and coefficients. It does not give us all the information about the regression model, but it gives us specifically what we might query. For example, the slope and intercept are displayed separately.

3.3.1 *Test IV-A*

In this test, we regress X on a constant term and powers of X. However, none of the packages offers multiple regression, so we cannot apply this test.

3.3.2 *Test IV-B: Regress X with a constant and X.*

The intercept should be zero and the slope should be one. Oracle passes this test.

3.3.3 *Test IV-C: Regress X on a constant, BIG and LITTLE.*

This is a multiple regression, and no package can perform this test.

3.3.4 *Test IV-D: Regress ZERO vs a constant and X.*

Because ZERO has no variance, the program should either fail to compute coefficients and report that ZERO has no variance, or it should return zero for both the intercept and the slope. Oracle returns zero for the intercept and the slope. Oracle passes this test.

4. CONCLUSIONS

Wilkinson's test are designed to uncover flaws in statistical function of software packages. In our analysis of six SQL packages, we have identified several flaws:

1. An incorrect choice of data type all but guarantees inaccurate statistical results; a correct choice does not guarantee correct results.

2. Different packages use different rounding schemes. Only MySQL adheres to IEEE-754 rounding; the others do not.

3. MS Access and MS SQL use unreliable algorithms to compute the sample variance.

4. Hive returns the population standard deviation instead of the sample standard deviation using STDDEV.

5. The Oracle correlation function is unstable. It gives a correct answer for the preferred data type and incorrect answers for other data types.

6. MS Access fails to correctly handle a missing value case.

Developers should fix these errors qiuckly and, until then, users should avoid them. Further, "accuracy of algorithms" should be incorporated into the SQL standards [19].

5. REFERENCES

[1] Michael Alexander. *Microsoft Access 2007 Data Analysis*. Wiley, 2007.

[2] anonymous. `https://social.msdn.microsoft.com/forums/sqlserver/en-US/9daa1b60-d11c-421b-8b87-e38a299e372c/rounding-off-issue-during-oracle-to-sql-server-migration`, 2013. Accessed: 2019-07-15.

[3] U. Bankhofer and A. Hilbert. Statistical software packages for windows: A market survey. *Statistical Papers*, 38:393–407, 1997.

[4] Joe Celko. *SQL for Smarties: Advanced SQL Programming, 5e*. Morgan Kaufman, 2015.

[5] T. F. Chan, Golub G. H. and R. J. Leveque. Algorithms for computing the sample variance: Analysis and recommendations. *American Statistician*, 37:242–247, 1983.

[6] Hwan-sik Choi and Nicholas Kiefer. Software evaluation: Easyreg international. *International Journal of Forecasting*, 21(3):609–616, 2005.

[7] Marin Fotache and Catalin Strimbel. Sql and data analysis. some implications for data analysis and higher education. *Procedia Economics and Finance*, 20:243–251, 2015.

[8] Kellie Keeling and Robert Pavur. Statistical accuracy of spreadsheet software. *The American Statistician*, 65(4):265–273, 2011.

[9] R. F. Ling. Comparison of several algorithms for computing sample means and variances. *Journal of the American Statistical Association*, 69:859866, 1974.

[10] Gordon S. Linoff. *Data analysis using SQL and Excel*. Wiley, 2015.

[11] Richard G. Lomax. Statistical accuracy of ipad applications: An initial examination. *The American Statistician*, 67(2):105–108, 2009.

[12] Wei Lu, Jiajia HouYing, YanMeihui, Zhang Xiaoyong, and Thomas Moscibroda. Msql: efficient similarity search in metric spaces using sql. *The VLDB Journal*, 26(3):829–854, 2017.

[13] B. D. McCullough. Wilkinson's tests and econometric software. *Journal of Economic and Social Measurement*, 29(1-3):261–270, 2004.

[14] B. D. McCullough and A. Talha Yalta. Spreadsheets in the cloud – not ready yet. *Journal of Statistical Software*, 52(7):1–14, 2013.

[15] Kamat Niranjan and Arnab Nandi. A closer look at variance implementations in modern database systems. *SIGMOD Record*, 45(4):28–33, 2016.

[16] C. Ordonez and S. K. Pitchaimalai. Bayesian classifiers programmed in sql. *IEEE Transactions on Knowledge and Data Engineering*, 22(1):139–144, 2010.

[17] William R. Pearson and Aaron J. Mackey. Using sql databases for sequence similarity searching and analysis. *Current Protocols in Bioinformatics*, 59(1):1–22, 2017.

[18] G. Sawitzki. Report on the numerical reliability of data analysis systems. *Computational Statistics and Data Analysis*, 18(2):289–301, 1994.

[19] Charles Severance. Elizabeth Fong: Creating the SQL Database Standards. *Computer*, 47(8):7–8, 2014.

[20] Robert P. Trueblood and Jr. John N. Lovett. *Data mining and statistical analysis using SQL*. Apress, 2001.

[21] Leland Wilkinson. *Statistics Quiz*. Systat Inc., 1985. `http://web.stanford.edu/~clint/bench/wilk.txt`.

[22] Leland Wilkinson. Practical guidelines for testing statistical software. In P. Dirschedl and R. Ostermann, editors, *Computational Statistics*. Physica-Verlag, Heidelberg, 1994.