Whitepaper

embarcadero

All Skewed Up

By Joe Celko

For Embarcadero Technologies

September, 2015

Table of Contents

Descriptive Statistics 3
How Does This Happen to a Database? 6
The Data is not on a Valid Scale 7
Data Can Just Be Skewed 7
Bell Curve versus Tracy-Widom Distribution 8
Data Can be Normal, but "Filtered" 9
How to Handle the Database 10
Adjust the Data 10
Change Your Scales 11
Physically Partitioning the Disk 12
Conclusion 14
About the Author 14

Pick any two positive integers. Add them and get a third number. Then use the last two numbers and do it again. And again, for a long time. Finally, take the ratio of the last two numbers. Without any other information I am sure that ratio will be ~1.618033988749894848204586834. This is called the Golden Ratio or ●, and it works out to (1 + √5)/2. It occurs as a law of nature. Nobody programmed it, per-arranged it or used a magic trick on you.

The normal distribution (aka "Gaussian distribution" or "Bell curve") is also like that. It occurs from the averages of random variables *independently* drawn from *independent distributions* are normally distributed. This is called the Central Limit Theorem. More specifically, where x1, x2, .., xn are independent and identically distributed random variables with the same *arbitrary* distribution. If you had a good statistics course, you might have seen something similar to the demonstration we did for phi. Take a set of numbers, and pull out all the subsets of all the possible sizes and take their averages. Then graph them. This is easy to do with a computer and if you have graphics, it is impressive to watch the bell curve appear as you add each subset computation, no matter what the original set.

But even more that this, the normal distribution is often a good approximation to other common distributions. In particular, the binomial distribution (think about flipping coins), Poisson distribution (think waiting queues), the chi-squared distribution (think a goodness of fit in inferential statistics) and the Student's t-distribution (think of smaller sample sizes) are often assumed to be normal to make math easier. But the truth is that while they are unimodal distributions (in English, their curves have a single "peak" in them), they are skewed by their nature, while a perfect normal distribution is symmetric about the mean.

The only place you will find a perfect normal distribution is a textbook and perhaps a physics problem. Data that comes from people into a database will never be so perfect.

Descriptive Statistics

Everything you learned about descriptive statistics is wrong. Not completely wrong, but wrong enough to make problems in your databases and analysis. Your optimizers are also built on the same models, so they are not doing a perfect job, either. But we are getting closer.

Back in the old days, before insanely cheap computing power and insanely large data sets, we did simpler math and made assumptions. The three most common measures of central tendency in a data set were the Mean, Median and Mode.

The Mode is the most common value in the data set. It is lousy measure of central tendency. Two data sets with totally different shapes can have the same mode. A single data set can have more than one mode. The classic example of a bimodal distribution of income in a country with lots of poor people and lots of very rich people, but nobody in the middle.

The Median is the value which has the same number of cases less than it as are above it. But this is not quite so easy to define. Such a number might not be in the data set, so the common rule is that when the data set has an even number of elements, you average the highest value in the lower half and lowest value in the upper half (basically the two guys in the middle when you sort the data). Now look at the set {1, 2, 2, 3, 3, 3}. Using the rule, we get a median of (2+3)/2 = 2.5; but if we use a weighted median, we consider the number of occurrences of the middle values. This gives us (2+2+3+3+3)/5 = 2.6, which is a better estimate of central tendency than a plain median. It shows the skew to the right.

The Mean the good old high school arithmetic average. But it is thrown off by a few extreme values in the data set. If Bill Gates moves into my neighborhood, the average income for the neighborhood increase by orders of magnitude.

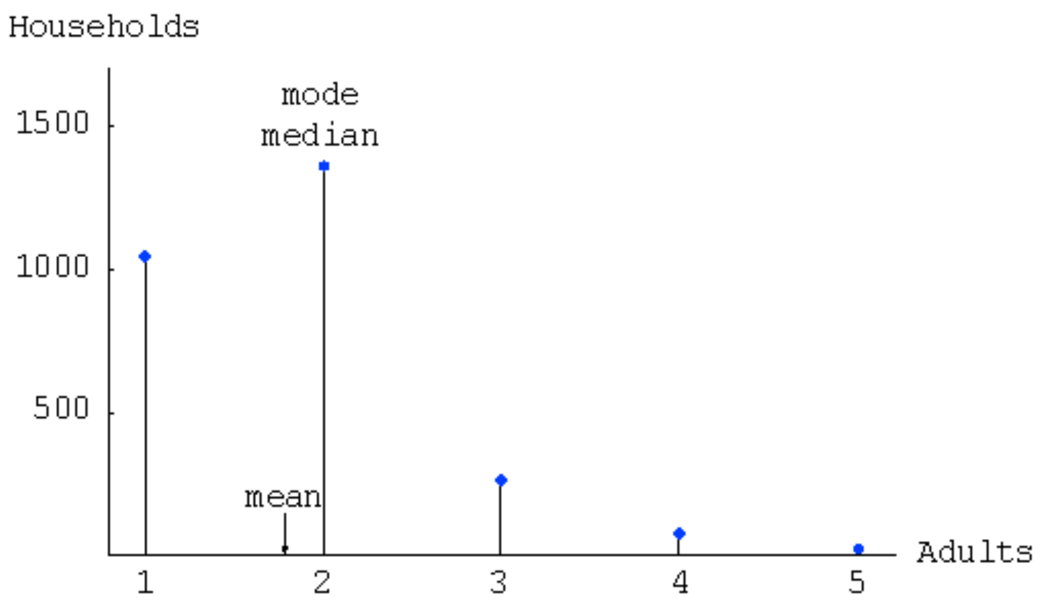
The classic book by Darrel Huff HOW TO LIE WITH STATISTICS (ISBN: 978-0393310726) is still in print since 1954. The example he gives is a very dated set of annual salary data.

Employee count	Annual Salary
1	$45,000.00
1	$15,000.00
2	$10,000.00
1	$5,700.00
3	$5,000.00
4	$3,700.00
1	$3,000.00
12	$2,000.00

The mode is $2000.00 per year, but the median is $3000.00 and the simple average is $5700.00.

The folklore in many introductory statistics books is that mean, median, and sometimes the mode are related to the skew so that the mean is right of the median under right skew, and left of the median under left skew. Pulling quotes, we have: "[T]he mode, median, and mean do not coincide in skewed distributions, although their relative positions remain constant – moving away from the `peak' and toward the `tail,' the order is always from mode, to median, to mean." (Levin and Fox, 2003, p. 85; also Levin and Fox 2004, p. 56).

It does not work when there is no single mode or we have a long tail on one side and a heavy tail on the other. It is also not a good rule for skewed discrete distributions. A good small sized example is the distribution of adult residents across US households. The skew is to the right, yet the mean is left of the median and mode.



(http://www.amstat.org/publications/jse/v13n2/vonhippel_figure2.gif)

This is a classic thin tail/ fat tail distribution. If this were an on-line bookstore, this would be more extreme. A few best sellers would dominate sales, followed by lots and lots of 1 or 2 copy sales of obscure titles and self-published e-books.

How do I compute the skew? Unfortunately, the answer is "which skew?" because there are several. We can start with Pearson's moment coefficient of skewness, which is the third standardized moment. Frankly, the only thing in that definition that makes sense to most people is that they might remember Pearson was a famous statistician and has a lot of stuff named after him. The formula is awful and can be undefined for some distributions.

The easiest for computations is ((mean – mode) / standard deviation). This is called "Pearson's second skewness coefficient" or the median skewness. This is nonparametric and the math is easy. Database people like those properties because query optimizers often keep these statistics.

How Does This Happen to a Database?

Obviously trying to query data that badly skewed will have problems. The algorithms to collect, aggregate and buffer data on the fat tail will not be the same as the ones for the thin tail. It would be nice to have nice smooth data so one algorithm could be used for all queries to the data set.

Two queries can have the same basic code in the FROM clause, but what values are used to search the same column in the WHERE clause The user should look at the column statistics and test a query using column values with the worst cardinality as well as with the best cardinality in order to understand best and worst case run times. To give an extreme example, consider the ISO sex codes (0 = unknown, 1 = male, 2 = female, 9 = lawful person; organizations). Searching a table of combat Marines with (sex_code = 1) is a waste of time, since you get back 99.78% of the rows. But searching that table of combat Marines with (sex_code = 2) is a very strong filer that returns the remaining 0.22 rows.

The idea that the same query does not always execute the same way every time is very strange when you began programming in a deterministic procedural language. You should look at the column statistics and test a query using column values with the worst cardinality as well as with the best cardinality in order to understand best and worst case run times.

The one place that a query plan is “locked in concrete” when a stored procedure is compiled and executed the first time. Unless you re-compile the procedure before *every* invocation, all we have is the initial plan. If the values (called arguments in SQL) given to the parameters in that invocation, were typical then we are probably okay. But if I invoked jt with a skewed value, like looking for female combat Marines, the query performance can be truly awful.

The Data is not on a Valid Scale

There is no mean value on a nominal scale. There is an old IT joke that the most common first name on Earth is “Mohammed” and the most common last name on Earth is “Wang”; ergo, the most common name on Earth must be “Mohammed Wang” of course! Nominal scales to not work that way. If you used text for the values on such scale, you would immediately see that (“John” + “Fred')/2 is absurd. You can get “clusters” of values, like “Wang” and “Mohammed”, but this is not the same as skew.

I can fool myself if I improperly used digits in a nominal scale and treated them as if they are integers, I can get a meaningless numeric result ('3' + '4')/2 = 3.5, *if you ignore the quote marks*). Currently, my favorite tee-shirt has the slogan “On a scale from 1 to 10, what color is your favorite letter of the alphabet?” because people will actually stop and try to answer the question.

The mean taken on a log or exponential scale also makes no sense. For example, the Richter scale for earthquake is exponential – a force (n+1) earthquake is *ten times* the power of a force (n) earthquake. The same problem applies to decibels for noise. Yes, this data is numeric, but it is not linear.

Data Can Just Be Skewed

Data can be skewed by its nature. Using the bookstore example, assume the small sales titles arrive first. An index or other access method based on the author's names will take a shape over a large range, over the whole alphabet, because the tail will have a larger number of authors. Now assume the fat tail sales titles (best sellers) arrive first. The number of best selling author is relatively small, so the same tree-based index will have a different shape this time. It will be shallower.

This is not just a problem with just tree-based indexes. Hashing algorithms also depend on the number of elements, their length and their statistical distribution. Data access has no magical, one-size-fits-all tools.

If the data distribution changes over time, it is best to re-index after each major shift in the data. In this example, when the "Brick & Mortar" stores data comes in, it is probably a good idea to re-index. When the electronic sales data has a surge, then re-index or re-hash.

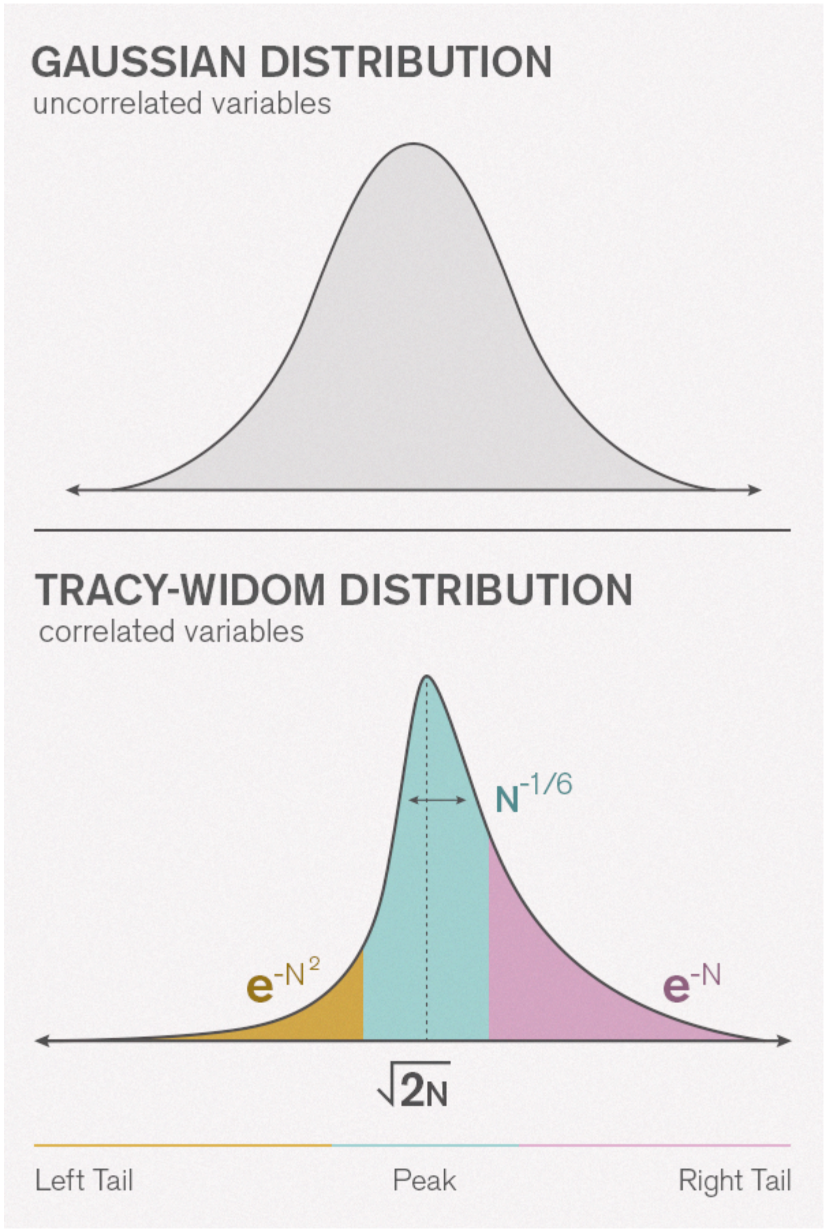
There will be occasional events that break the pattern, but things tend to return to normal (pun intended; it is called regression to the mean). For example, I was the best selling author on Amazon UK for almost a week many years ago. A teacher ordered a case of one of my books for a class. Then some guy named "John Grisham" had a new thriller and my 50 copy sale disappeared from the list. Sic transit gloria mundi.

Bell Curve versus Tracy-Widom Distribution

Three is a special case of naturally skewed data that is becoming more important because of Big Data and real time data. Classic statistics deals with independent variables. But we deal with interrelated systems that have feedback mechanisms among many variables!

In 1972, biologist Robert May created a model of an ecosystem with competing species. His goal was to discover if some species will inevitably wipe out others or if the system can stabilize. He discovered systems have tipping points, but he did not find a statistical distribution.

Years later, mathematicians Craig Tracy and Harold Widom proved that the tipping point in that kind of model was the peak of a statistical distribution and came up with a formula. Then the distribution started showing up in physics and mathematics. It also shows up in financial markets and the Internet. This is one of the laws that emerge from complex systems, regardless of what the system happens to be. This is the one for systems with feedback!



Data Can be Normal, but “Filtered”

The data can have a normal distribution, but we get a filtered sample. Garrison Keillor's running joke on the Prairie Home Companion radio show (http://prairiehome.org/listen/podcast/) is that in the mythical town of Lake Wobegon, “'all the women are strong, all the men are good looking, and all the children are above average”; they only report selected data or false data colored by ego.

This was the situation in Atlanta when public school system teachers conspired to alter student test scores to get performance bonus pay. They are now in federal prison on RICO charges.

A classic folktale in statistics was bread rationing in Nazi Germany. The Nazi regime cut domestic bread rations from 12,450 grams in May 1944 to 9,700 in August, 8,900 in December, and 3,600 in April 1945. Various mathematicians are made the lead character in the story. The professor plots out his bread ration and finds the distribution is short of what it should be. He confronts the baker and gets a promise that loafs will be made with a correct bread mold. The professor then tracks the new bread, but now the statistical distribution is only *the upper half of the old normal distribution* from the old molds. Lake Wobegon with a German accent! The baker is simply giving the professor heavier loafs to avoid getting in trouble with the Nazi government.

These biases do not have to criminal or deliberately erroneous. Reporting the sales of e-books is instantaneous; reporting the sales of print books in brick and mortar bookstores is not. This means that the long tail of the sales curve arrives in the database before the fat best sellers tail that comes from bulk orders, book tours and paper report forms. But you check it for crime or error out anyway.

How to Handle the Database

Broadly speaking, you can try to work around skewed data two ways. Adjust the data that you have in the software or look for hardware solutions.

Adjust the Data

First determine if the data is *really* skewed. Yes, objectivity sounds obvious, but people will try to "correct" or "adjust" the data to match their expectations. The New York Time Best Seller list is famous for this in the book trade. People do not look for the reasons that the data is skewed because that can be hard work, so it is easier to just assume that something cannot be right. In the old days, the NY Times barred titles from the Best Seller list for "strategic bulk sales" based on their opinion and estimates by a few New York distributors. Today, you can get precise data from Barnes and Noble, Amazon, the Wall Street Journal and Nielsen, which all provide data with and without bulk sales in computerized format.

Another example of "data adjustment" problems is the wholesale grocery business. They work with a small profit margin selling goods that spoil. You buy in bulk in metric units, by weight, volume or a combination of weight and volume, or perhaps by traditional packages. Then you sell "by the each" or in US Traditional units (no, we are not on the Imperial system) or re-package it with a new bar code.

Just consider Canadian and American beer bottles. Many Canadian brewers package beer in 12 imperial-fluid-ounce bottles, which are 341 ml each. American brewers package their beer in 12 US-fluid-ounce bottles, which are 355 ml each.
Arrgh! Heineken is well known for its famous green bottle and red star. But for many years it had to be sold in brown bottles in parts of the world by law.

A classic data warehouse horror story involves trying to resolve the packaged meats sold by the POS system in the stores against the sides of beef in the butcher shop at headquarters. The data implied that the company was buying six legged cattle.

Change Your Scales

Modern SQL has computed columns, which can be virtual or physically materialized in the table. If you can find an old engineer, ask about all kinds of graph paper we used back then. It is very hard to draw logarithmic curves freehand, but if the graph paper is scaled logarithmically, then you just have to draw a straight line with a ruler. Today, the computer does all of the artwork for you.

This is a mathematical concept called a transform. A transform maps a complex function into a simpler function, we do the computation with the simpler function and then an inverse transform restores the result back to the original function's domain. You know this concept already, if you had logarithms in High School algebra! Logarithms turn multiplication division into addition and subtraction. The anti-log turns that sum back to the product or quotient we originally wanted.

This is very nice for mathematical functions but trickier when you try to apply this tool to real data.

Let me go back to the example of books sales. I can use a "split transform" to un-skew the data. The fat, short tail (the best sellers) is the original sales figures by author and title. These are separate data elements. But then I switch to a categorical scale for the long thin (other books) on the other side of the peak. The categories are subjective, but the publishers pretty much agree on this informal system classification. Occasionally a title will cross categories (for example, J. D. Robb's Eve Dallas series of Romance – Science Fiction – Police novels), but the major bookstore chains resolve it.

So we see the top ten best sellers, then the sales by category. This is heresy in pure data theory; remember my tee shirt slogan about the color of letters of the alphabet? The first principle of scales is that you can only convert scales when they are of the same kind.

The mixed scales *can still give you information*, but you have to be careful. Again, remember trying to make sense of my tee-shirt? If HUNGER GAMES makes the best sellers (the fat tail), perhaps sales of cook books also increased? Maybe or maybe not. Correlation is not causality. Or perhaps sales of movie related books, specifically things related to that movie, went up? But I do not have a "Hunger Games materials" category in my thin tail. This is a key problem with this technique; categories are often fluid in the real world. Teenage vampire romance actually became its own category!

Physically Partitioning the Disk

Database systems today are more sophisticated than a simple disk. In particular, we have user-controlled partitions. A partition is a set of logical containers for data or for an index, and it gets its name from the same mathematical concept in set theory. The union of all of the partitions of a set gives us the original set; the intersections of all of the partitions of a set gives us the empty set. In English, we have sliced up a pizza.

The advantage is that each partition can hold a known subset and have a local processor. The partitions are often based on temporal divisions or organizational units or other divisions related to reporting.

There is no ANSI/ISO Standard for partitioning and I hope there never is. This is an implementation feature and not a proper part of a programming language. The CREATE INDEX statement syntax came from the SQL/Access group. Having said that, the syntax is usually something like this:

```
CREATE PARTITION FUNCTION <partition function name>
(<input_parameter_type>)
AS RANGE [LEFT | RIGHT]
FOR VALUES (<boundary value list>);
```

The <partition function name> explains itself. But it is partitioned on only one column and the possible data types often excludes exotic data types, such as text, image, XML, oversized CHAR and VARCHAR and external user-defined data types. The partitioning column, is specified in the CREATE TABLE or CREATE INDEX statement.

Partitions are based on half-open intervals, the same as the ISO-8601 temporal model. In English, this means that a partition has one side that does not include the boundary point that defines it. The most common example is that 24:00:00 hrs today is *really* 00:00:00 hrs tomorrow.

The range option tells us which side of the partitions are open; it has to be one or the other. The boundary value list tells where the cut points are (they will be sorted). It has to be a list of constants or values that can be cast exactly to the input parameter type. There is an upper limit to how many "slices" you can "cut up your pizza" in the implementation, but it is usually large enough.

```
CREATE PARTITION FUNCTION Partition_by_Magnitudes (INTEGER)
AS RANGE LEFT FOR VALUES (1, 100, 1000);
```

effectively becomes an expression to determine a simple rule for each partition from 1 to (n).

```
CASE WHEN partition_col <= 1 THEN 1
WHEN 2 partition_col > 1 AND partition_col <= 100 THEN 2
WHEN 3 partition_col > 100 AND partition_col <= 1000 THEN 3
ELSE 4 END AS partition_nbr
```

I can combine physical partitions with split scales, but with great effort and lots of guessing.

Conclusion

Mitigating skewed data is hard and not obvious. There is no magic formula, just a few tricks that have to be justified on a case-by-case basis.

About the Author

Mr. Joe Celko serves as Member of Technical Advisory Board of Cogito, Inc. Mr. Celko joined the ANSI X3H2 Database Standards Committee in 1987 and helped write the ANSI/ISO SQL-89 and SQL-92 standards. He is one of the top SQL experts in the world, writing over 700 articles primarily on SQL and database topics in the computer trade and academic press. The author of six books on databases and SQL, Mr. Celko also contributes his time as a speaker and instructor at universities, trade conferences and local user groups.