Aggregate Deceptions: Finding Suitable Aggregation Strategies and Appropriate Functions

Michael J. Handley

Michael J. Handley is the director of information strategy at the YMCA Association of Middle Tennessee.
mhandley@ymcamidtn.org;
mike@mikehandley.com

Abstract

Whether within a powerful visualization tool, an OLAP database, long aggregations in MongoDB, or simple report queries directly over an on-premises data warehouse, the mathematical algorithms written into the common aggregate functions can at times create deception over data sets, particularly when there is a less-than-perfect distribution of details across the range. Statistical functions can be used in tandem and with other aggregate functions to mitigate numeric reporting risks.

Moreover, the periodicity choices—the backbone of any typical aggregation strategy—can be thought of as an orthogonal informational complement to the longitudinal approach for describing an entity and how it changes over time

No Grain, No Pain

We gather data—lots of it. We become intimately aware of just how much detail can or cannot be absorbed at a glance or displayed in our reporting tools. We want less information but need complete information. How else can important business decisions be made?

So we aggregate—and then we aggregate some more. Sometimes we even aggregate our aggregations, which itself can lead to numeric deception in values. Whether they are baked into the recurring build of sales and marketing's favorite OLAP cube, within a modern visualization tool that offers data management, or a materialized summary table created to link from accounting's Excel spreadsheet via a custom database procedure written in SQL, an aggregate function is an algorithmic construct that delivers a single value across a given subset of records (defined by the chosen groupings).

Because of this, the aggregate is, in every sense of the phrase, "information lossy." It throws detail overboard and makes the best attempt at producing a single set attribute that we can better digest. We accept it as rock-solid truth, but the algorithms do make compromises.

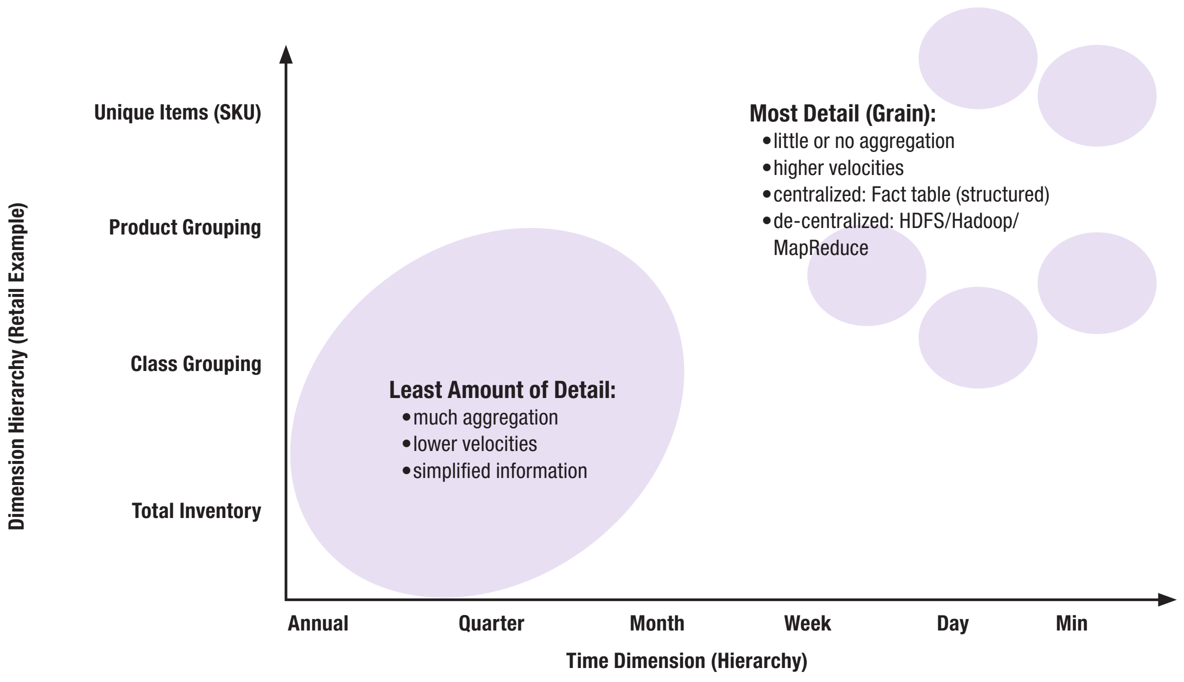

Figure 1: Information detail vs. aggregation.

The output of that mathematical construct is generally accepted to be correct with given inputs, but it does not always tell the whole story. In this article, we will revisit the common aggregate functions in a new light, explore two categories of "information deception" that arise, and discuss how they are distinctly different.

Moreover, we will look at how the aggregate functions, over some grouping, can be thought of as existing orthogonally to a longitudinally changing state of an entity instance such as "customer" or "product." We will then look at simple things we can do to increase confidence in the aggregate values that are essential for business reporting.

Information Loss through Aggregation

To aid in our discussion, let's start with an analogy of information loss due to aggregation.

Our cellphones can snap images at a very high resolution (on the order of 12 megapixels), but what happens when we view these in a tool that cannot display that many pixels? We see only one little section of the image but in great detail. If the image is converted to, say, a JPEG format (which uses lossy compression), the file size becomes much smaller because the number of pixels has been reduced, and we can now see the Big Picture with even a limited viewer.

If we zoom in on this JPEG, however, we would no longer see the detail. In fact, what we would see are colored squares emerging. The amount of information has been reduced and we can no longer see the grain from whence it came. The JPEG "algorithm" has decided how to "aggregate" all that "grain" into fewer pixels.

Of course, we like to be able to see the Big Picture and then zoom into areas of detail. A modern image viewer can indeed do that, but it must aggregate the pixels (by

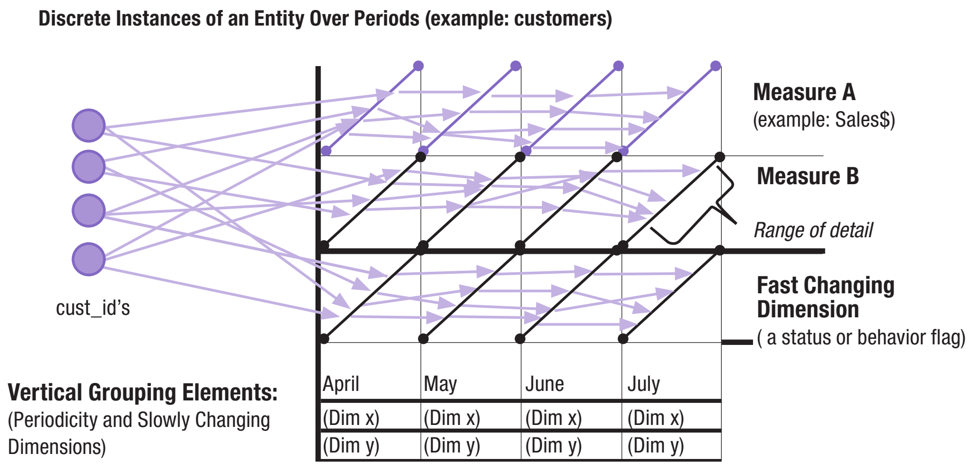
Discrete Instances of an Entity Over Periods (example: customers)
cust_id's
Measure A
(example: Sales$)
Measure B
Range of detail
Fast Changing Dimension
(a status or behavior flag)
Vertical Grouping Elements:
(Periodicity and Slowly Changing Dimensions)
April
May
June
July
(Dim x)
(Dim y)

Figure 2: Longitudinal aggregation.

some algorithm) as we zoom out. You can see the bride and groom in the photo image, but you can also zoom in to see the ring on her hand. (This might be thought of as drill-down using HOLAP or some other mechanical reporting means to get from the aggregated Big Picture down to the detail with a click or two.)

With or without the zooming drill-down, the Big Picture requires an algorithm to aggregate the detail and compress that information. At that point, the lower grain is gone from your information perception. For the time being, it is lost and assumed to be contained within the Big Picture view.

Does this sound like what happens with some of your reports—that is, some part of the truth is ultimately sacrificed for the Big Picture view? How can we best qualify these Big Picture values? How can we say, "If you look closely at the wedding guests seated behind the groom in the image, you'll see his college girlfriend seated in row five"? Simply, we can't say for sure. After aggregation, that important information is gone.

Two Kinds of Deception: The Zoom-In-Zoom-Out and the Criss-Cross Loss

The image pixel analogy is an example of what I will refer to as zoom-in-zoom-out information loss due to aggregate functions having to throw detail overboard. In another kind of deception—referred to as the criss-cross loss of changing-state information—the aggregate function's output tends to exist in that one grouping moment only.

Zoom-In-Zoom-Out

In Figure 2, note the diagonal line that intersects an entity instance (customer or product) and a measure at each period grouping. I refer to it as a timeline only because the time dimension is included in the grouping. It moves across from left to right. This could be thought of as a "slice" in one sense, but the intention here is to show that the vertical aggregations (data points on the diagonal range lines) are strictly bound to the current period grouping.

The zoom-in-zoom-out deception lives within the diagonal line. The data point range exists as a distribution across the diagonal line, but the aggregate function must return one and only one value for the set of data points on the diagonal, and therein is where the zoom-in-zoom-out deception lives. The detail data points might not be so evenly distributed. If we do not use a statistical function that speaks to the spread or distribution, we know very little about the set of detail data points.

Criss-Cross Loss

Looking at Figure 2 once again, moving from left to right (along what I will refer to as the longitudinal), notice that the value of each measure (the ranges are represented on the diagonal lines) can change longitudinally. This is the changing-state information, the detail history, and it can only be observed through full detail or measured within a longitudinal aggregation. In the vertical period, you can ask, "How many instances are there of each status and of each measure value?" or "What is the total for measurement A for the period?" However, you cannot know which particular entity instances constitute those results unless you listed them in full detail.

Longitudinally, you can ask, "Which entity instances have changed from this status to that status to some other status?" You won't know how many until you gather up and count each of the longitudinal "threads" that fit your multiperiod description. As another example, you could ask, "How many customers bought item Q in store X and then bought item Q online at least three times in the year following?"

You would first define the rules of the "thread" longitudinally, then count the instances or perhaps sum the gross revenue of all the customers that fit that pattern. In what ways can aggregate functions be used longitudinally across the changing state of history? In an example below, I will show how a longitudinal aggregation can be used to increase accuracy using a longer range of periods.

Why We Aggregate on the Way Out of the Data Store

Yes, life would be easier for the ETL team if everything was rolled up on the way in to the data warehouse. Think of how much space we could save! However, we know that the level of grain (detail) provides us with the flexibility to aggregate (or follow the changing state of) the measurements across the many dimensions of our choosing.

Unfortunately, like the compressed photo image, the aggregate function will and does reduce our information footprint across that particular grouping subset. We need to know more about the granular detail that went into the making of the aggregate value, without always dragging the endless detail records along with it.

Average, Variance, Standard Deviation, Count, Sum, Max, and Min

I want to quickly review some of these functions to point out some differences in them. Of the most common functions that we use in code or cube or visualization tools, I will split them into three basic classifications for expedience:

- Statistical functions (average, variance, standard deviation, mode, median)
- Accumulators (count, sum)
- Range endpoint finders (max, min)

I've ordered them by how likely they are to create a numeric deception. There are also several variations of each of the statistical functions (mainly as a result of sample size differences) that would take us beyond the scope of this article.

Rangers: Range endpoints, the last classification in the list, are transparent. These indicate the boundary and are usually honest depending on certain product specifications, such as how the NULL condition is handled. A runtime error may sometimes occur as a result of data-type casting on the fly, but for the most part, these are straightforward.

Aside from the important role of finding the range endpoints, I sometimes use these to bubble up a value from the detail recordset if and only if it is one-to-one with one of the grouping elements. This avoids having to actually include that element in the expensive grouping element list.

Accumulators: Sum will always tell an explicit numeric truth, but it is coerced into deception by simple cardinality issues. Recall that a set join (inner or outer) multiplies the base sets. The join condition is where we stipulate the rules of that multiplication and thereby are able to predict the number of records in the result set, but the condition logic can be tricky and product specific. The count function can also be impacted by these same cardinality boundaries, which is why it should be used within a quality measure to ensure that the join condition is behaving as intended.

If you want to arrive at the annual figure, you might be tempted to use 12 months' worth of these average values. However, the "average of averages" is generally taboo.

One Important Statistical Function

Of all the aggregate functions, the statistical functions are poised to be the most deceiving unless used alongside each other to mitigate the risk of deception (more on that later). The most common of these, the *average* (*arithmetic mean* in statistical terminology), can be the root cause of many struggles to audit reports. A common misuse, the "average of averages," can lead us to reach for some type of weighted average scheme that is baked into the tool but never with any guarantee of correctness.

Average: Aggregating Month and Year (Using the Vertical in Figure 2)

Imagine that you have configured the tool (or have written code) to produce a one-month period aggregation that expresses Average Sales$ per Customer by Month. This could be from brick-and-mortar points of sale or your online shopping cart. You will most likely use Total Sales$ / Total Count of Customers and the number will be correct for that one month, using "month" as a discrete time period.

Referring to Figure 2, the Total Sales$ would be the sum of all the points on the "Measure A" diagonal for any given period. The Total Count of Customers is a distinct count of the cust_id that intersects there. (We would use a "distinct" count here because a customer may make more than one purchase in that month period.)

If you want to arrive at the annual figure, you might be tempted to use 12 months' worth of these average values stored in a table and then average those values. However, we all know this as the "average of averages" that I mentioned earlier and that it is generally taboo (it would work only if the number of customers were exactly the same every month).

Another common approach to get the annual figure would be to use the same Total Sales$ / Total Count of Customers that we use for a one-month period but now taken over a year period, as in Total Sales$ for the Year / Count of Customers for the Year. This is still using the vertical column approach given in Figure 2, but the time dimension grouping has been modified to be a full year, not just a month.

Is this annual figure correct now? Not really. How do we account for quicker customer turnover cycles? Won't that skew the average downward? Can we use an average customer count for the year or will that mask seasonality? This mayhem is a direct effect of using the "aggregation / aggregation over period" approach (the vertical approach in Figure 2).

Average: The Longitudinal Approach (Using the Horizontal in Figure 2)

Let's look at the Sales$ per Customer example flipped on its side. In Figure 2, follow any one instance across from left to right. Notice how the "stateful" information for that entity instance is preserved. That is to say, we see the rich information about how the measures change from period to period. This is an approach that an analyst might use to assemble a linear regression over a set of variables.

For this same Sales$ example, let's instead look at each customer for a one-year period. I'll call this the Sales$ per Customer by Year, with detail points on each of 12 monthly periods. It will be a recordset that has one record for each customer (think "grouped by cust_id") with a field element Sales$ for the year for that one customer. These will each be summarized across the 12 periods using the SUM accumulator longitudinally.

We need to understand the many ways a number can be arrived at and what informational compromises have been made to arrive at informational simplicity.

From that recordset, we can now vertically aggregate the average of the Sales$ across all of the distinct customer records. This should work out to be the same as the above Average Sales$ per Customer by Year because total sales didn't change and the denominator didn't change (we used the full set of customers regardless of their duration).

Longitudinal aggregation offers an advantage: we can assign a duration coefficient for each customer to correspond to the number of months within which they made a purchase (1/12). Now, the Sales$ per Customer will be the same as before, but the denominator is reduced to better represent the agile customer presence. It doesn't matter how seasonality has changed your customer counts; the longitudinally derived Average Sales$ per Customer will not vary indirectly with how many customers you had (or didn't have) in any one of those 12 periods.

If your line of business sees a great deal of customer churn as a result of your business model, this can be a real eye-opener and a new metric to explore. The longitudinal approach to the customer (or product) also enables us to extend across more dimensions by simply adding them into the groupings. The key to making it work is to hold the entity as atomic, such as customer or product, as the lowest grain.

Other Statistical Functions That Help Paint the Backdrop

I outlined two categories of deception earlier: zoom-in-zoom-out information loss due to aggregate functions throwing detail overboard and the criss-cross loss of changing-state information because the aggregate functions are living in that grouping moment only.

Without wading into deeper statistical waters that speak to sample size and error (which is beyond the scope of this article), here are some ways we can use the common statistical aggregate functions to inspect our boundaries and give us a better understanding of what the average is or isn't representing. In short, we want to quantify the zoom-in-zoom-out deception. These functions process the data points on the diagonal lines in Figure 2.

Standard deviation: This statistical function is used as a measure of data spread or distribution of the detail data points. In Figure 2, this would describe the points on the diagonal lines, vertically. If the points are evenly distributed across the range, about 68 percent of all the data points will be within one standard deviation and 95 percent of all data points will be within two standard deviations.

That being said, your data might be all over the place. I use the standard deviation to tell me about possible data-point outliers and general distribution. I want to know, for example, that if I get an average value of, say, 215, it is not composed of three data points at 850 and nine points in the 3–5 range.

Standard deviation is, mathematically speaking, the square root of the variance, the latter of which tends to be less useful in typical business intelligence/data warehousing (BI/DW) scenarios. When used alongside the well-worn average, standard deviation can provide a broader understanding of what the "average" is intending to numerically represent, and it reveals in a different context some of the zoom-in-zoom-out information that was thrown overboard by the average function.

Variance: This tells us about the spread and distribution of our detail data points across the range with respect to the average. Simply put, it first finds the average; then, for each data point, it finds the difference between it and the average and then squares the result (it ends up an absolute value). After all of the point differences-to-average have been arrived at and each one squared, variance takes an average of those values.

Mode: This function tells us where there is the largest "cluster" of detail data points. Maybe it is right next to the average or maybe it is far away.

Median: This is the most central detail value in the range. If you take the max and the min of a range, find the midpoint, and then find the nearest value; that is the median.

Conclusion

Depending on how we aggregate our data, we can expect different results. As stewards of the truth, we need to understand the many ways a number can be arrived at and what informational compromises have been made to arrive at informational simplicity. These strategies and approaches are independent of the tools and are part of the very fiber of the mathematics that underlies DW, BI, and information theory in general. ■

Instructions for Authors

The *Business Intelligence Journal* is a quarterly journal that focuses on all aspects of business intelligence, data warehousing, and analytics. It serves the needs of researchers and practitioners in this important field by publishing surveys of current practices, opinion pieces, conceptual frameworks, case studies that describe innovative practices or provide important insights, tutorials, technology discussions, and annotated bibliographies.

The *Journal* publishes educational articles that do not market, advertise, or promote one particular product or company.

Visit tdwi.org/journalsubmissions for the *Business Intelligence Journal's* complete submissions guidelines, including writing requirements and editorial topics.

Submissions

For complete submission guidelines and suggestions, visit tdwi.org/journalsubmissions

Materials should be submitted to:
James Powell
Editorial Director
Email: journal@tdwi.org

Upcoming Deadlines

Volume 21, Number 3
Submission Deadline: May 13, 2016
Distribution: September 2016

Volume 21, Number 4
Submission Deadline: August 5, 2016
Distribution: December 2016