



Missing Data And What To Do About It

Tom Breur
January 2010

Introduction

Missing data are a fact of life. And sometimes they cause serious disruption. Fortunately, there are often quite acceptable ways of dealing with missings. The key lies in accurately discerning different flavors of “missing” that may exist, and dealing with each kind accordingly.

In this paper we will present a simple framework to deal with missings. We propose a simple 2×2 matrix. On one axis we have voluntary versus involuntary registration of data. This typically equates to, for instance, survey data versus electronic recording of activity as in a database (or data warehouse, DWH). On the other axis we have rightfully or wrongful missing. Is a field “allowed” to be missing? Based on these two criteria you decide how to deal with missings.

Because missing data can be so problematic to deal with, you’d really like to *avoid* this problem altogether. How can we do that? And even if some missings remain inevitable, how can we mitigate disruptions to analysis and reporting?

What is “missing”, really?

A “missing” or “Null” field can have several meanings. For the purpose of this paper we’ll define “Null” as no value in a text field (not even an empty “space”), and either no value or a designated value¹ (like the smallest possible integer) in a numeric field.

For now we make a gross distinction between missings in surveys, and missings in a database context. At a more “philosophical” level, this represents whether some voluntary action was required by subjects.

¹ The reason why some databases use such a designated number is purely technical: in the old days they just couldn’t “digest” true Null values, so a proprietary number was used to represent “no value.”

Sometimes people need to actively “do” something, as in surveys, and sometimes they don’t, as in DWH extraction, transform, load (ETL).

In a survey context, missing could mean that a respondent has indicated a certain question was not applicable for him (or her). But of course it could also mean they failed to answer the question. And it could mean they might or might not have answered the question, but the data entry typist missed their response.

In a database context, missing can likewise have multiple meanings. In his excellent treatment, Pyle (1999) distinguishes between “missing” and “empty” values. “Missing” values are Nulls for which no value has been entered in the database, but for which an actual value exists. For “empty” values, no real-world value can be supposed.

The distinction between “rightful” and “wrongful” missings is crucial. In a database context it can be tempting to ignore this distinction. When looking at a marketing database, you could say that a customer’s balance for a savings product she doesn’t own is “zero.” And in a way this is true. But it is a poor representation of reality. It is much more appropriate to represent this value as “empty”, using Pyle’s terminology.

We have found, time and time again, that ignoring these distinctions can wreck havoc for your predictive models. Also, reporting can become awkward or downright erroneous (see also: Thomsen, 2002). The way OLAP tools deal with these distinctions differs greatly among software providers, and is far from trivial. The short message is: do not assume your tool will calculate derived measures accurately without verifying!

Another example might be derived variables that require some “minimal” amount of data to be calculated. Let’s suppose you want to calculate an average savings balance over the past 12 months. This value requires at least 12 months worth of history. If a customer has been with the bank less than 12 months, no value can be calculated. It *could* be imputed, though, if the analytic purpose warranted this. Of course this is *not* the same as the average of the months for which *a* balance was available.

Some purists might consider these distinctions flaws in data modeling, and we don’t object. What is important is that data are and remain a truthful representation of reality, and that derived calculations that are based on these numbers (containing sometimes inevitable missings) remain accurate and auditable.

How to deal with missings?

The way you deal with a missing value depends on the “flavor” of missing you are facing. But also on the analytic requirements you are negotiating. There are preventive and contingent actions. Sometimes you can avoid or minimize missings (preventive action). And after you are “stuck” with missings, the options at your disposal to deal with them (contingent action), depend on the context in which the data were acquired. And on how you plan to use (or analyze) the data.

A value that isn't there, simply isn't there. Don't ever forget that although we may come up with elegant replacement algorithms, that's still what they are: an artificial way to “guess” what the true value most likely was. There is no substitute for capturing data right in the first place. No algorithm, no matter how smart, will ever be as good as capturing data correctly at the outset. As is so often the case, an ounce of prevention may well be worth a pound of cure.

Preventing missings in survey data

For surveys or data that are gathered after volition of humans, you prevent missings by good questionnaire or form design. Much has been written about questionnaire design, in particular in the scientific community. Graham Rhind has written an excellent eBook on form design, more specifically internet forms. His treatment of, in particular, name and address data is simply outstanding. It is available for free on his site (www.grcdi.nl/book4.htm).

And after you have done everything you can to enable your survey respondents or customers to provide the information you are seeking, apply a bespoke data capturing mechanism. For surveys this should include a dedicated code (unique!) to indicate the *respondent* didn't provide an answer. For handwritten materials you may even want to use a separate code for entries that were too difficult to decipher (if at all).

When we were conducting a large manual reentry data quality project, we found that many entries that were changed, and during a subsequent pass² changed *again*, were caused by sloppy handwriting (and/or ambiguous attempts to correct an entry). “Can't read” is something different from “did not provide an answer.”

² The reason we customarily perform multiple passes of manual data reentry is because we like to have an *empirical* estimate of the resulting data quality. Ironically, even after multiple passes through the same data, you still don't arrive at 100% quality. You can get arbitrarily close, but not quite at 100%. It's almost always valuable to know exactly *how* close to 100% you are getting.

One of the reasons to break these different sources of “missing” into subcategories, is *also* to distinguish between “respondent did not provide an answer” and data entry staff that missed a particular entry which then might *still* show up as a Null value.

Preventing missings in databases

For databases (or DWH's) you avoid missings by carefully and judiciously designing your ETL. Much has been written about that already (e.g. Kimball & Caserta, 2004). You *mitigate* nasty effects of unavoidable missings by choosing an appropriate data modeling paradigm. It's exactly in this area where a Data Vault (Linstedt et al 2008) really shines as an enterprise data warehouse (EDW) model. The reason for this is threefold:

1. many to many relations are universally applied throughout the model, and therefore point to every (unexpected) missing
2. registration of bad/missing data is preserved for eternity, and hence remains available for (further) analysis
3. reporting about missing data is extremely easy as this is an almost automatic “byproduct” of modeling itself

Using a Data Vault, very simple queries can generate detailed reports about (wrongful) missings, and these reports are easily specified *per data source*. Being able to identify *per source* where issues arise in the business has an extremely powerful signaling effect. You can't manage what you don't measure.

After you migrate data from your EDW to a data mart, and you choose to replace missings, the result using either a Data Vault architecture or a Kimball DWH, will be more or less the same for end-users. The difference lies in the fact that a Data Vault (intrinsicly) preserves a history of missings. Even the history of business rules that identify missings remains available for analysis. This can provide further insight to drive process improvement.

What Kimball qualifies as an “audit dimension” has the same functional purpose as the identification mechanism for missings in a Data Vault. The only difference is that a Data Vault preserves the *history* of missings (and replacement rules), and you can always specify these *per data source*.

Why replace missing values?

The case for replacing (or imputing) missing values applies solely to wrongful missings. If a value is “allowed” to be missing, you usually want to make this distinction. Unfortunately, precious few tools can deal with a numeric column where some of the records have an alpha entry like “n/a.” Until the

time where this becomes standard practice, we are stuck with appending an additional column (typically a boolean) to indicate whether values in the target column are rightfully (“empty”) or wrongfully missing.

Sometimes it makes sense for estimation purposes, to impute values. This will allow reports to be run across the entire database without any problem, even when some parts of the data may be “spotty.”

Another reason to impute missing fields is because some tools can only include records that are “complete.” Regression or Neural Network procedures, for instance, require that all fields in a record are non-Null. In particular if you have many columns, it would be a pity if you had to drop all records, even if only one single field were absent. Decision trees, on the other hand, can deal elegantly with missing fields.

The problem you face when imputing missings has to do with *bias*: in real-life it is exceedingly rare that missing data occur truly “at random.” When you get lucky, there is no significant relation with a target variable you are estimating. But even *if* there is significant association between the pattern in missings and the target value of your (predictive) model, you may still need to find an appropriate procedure for replacing missing values. Needless to say your imputation procedure should not introduce bias (easier said than done).

Conclusion

Life isn’t perfect. Sometimes data are missing. There are many ways to prevent this from happening, which we have alluded to. Even so, install processes that mitigate the effects of missings, because your prevention may not be perfect.

Depending on the context, missing data may or may not be acceptable. When working with the data, it is crucial to understand which of these two situations applies.

Often you need to lump several different sources of missing field values together during analysis (or reporting). But make that transformation as late as possible. After you have mapped multiple values onto one, the distinction is lost forever. Analyzing different *sources* of missings can provide valuable information for process improvement (*future* enhancements). This holds equally for survey data, and database missings.

When replacing missings, which is often necessary, it makes a lot of sense to indicate (with boolean variables) which values are and are not allowed to be

missing. Also, after replacement, you want to add an indicator that flags the records that actually *were* imputed. Since the objective of replacement is to *improve* the value of your data set as a whole, it is imperative the imputation algorithm does not add additional bias!

References

Dorian Pyle (1999) Data Preparation for Data Mining. ISBN# 1558605290

Eric Thomsen (2002) OLAP Solutions 2nd Edition. ISBN# 0471400300

Ralph Kimball & Joe Caserta (2004) The Data Warehouse ETL Toolkit. ISBN# 0764567578

Dan Linstedt, Kent Graziano & Hans Hultgren (2008) The New Business Supermodel – the Business of Data Vault Modeling. ISBN# 9781435719149