Strategic Data Handling for Pharmacometrics

Timothy Bergsma
Metrum Research Group, Tariffville, CT

Objectives

The relational database model [1] specifies that record storage order must be non-informative. In contrast, many pharmacometric and dosing software systems make use of highly informative record order. Our objective was to develop succinct, expressive, rapid data assembly techniques that respect record order.

Methods

Software was implemented within the package MIfuns (current version 4.2.2): a publicly available [2] extension of the R programming language [3]. Novel functions were developed by strategic extension of base R functions. The function stableMerge extends base merge to produce an order-stable left-join. The functions first, last, nth, and only extend base match to efficiently identify singularities and repeat them at all positions in a level. The function distance analyzes positional information with respect to singularities identified by nth; derived functions before, after, and aftersummarize distance information. Finally, reapply extends base apply, stretching each sub-result (using base c) to the original length while preserving indexed order.

To simplify this discussion, we classify functions by the dimensionality of their values: scalar functions (e.g., mean), vector functions (e.g., match), and table functions (e.g., merge). Further, we define functions as processes whose value has the same dimensionality as the primary argument and isometric functions as those whose value has the same extent (length of first dimension) as the primary argument. Last, we define stable functions as processes that do not permute the first dimension more than necessary. The term is borrowed from sorting theory. In the present context, “not more than necessary” implies “not at all.” Under these definitions, stableMerge is a stable isometric table processor, while nth (and) are stable isometric vector processors. Stable isometric processors have strategic utility in pharmacometric data assembly, where the dominant workflow paradigm involves “building up” a table by stepwise modifications. Processors support the paradigm by returning data in the same form as input: either a variant of a table or a variant of a column. Isometric functions prevent unintentional deletion or additions. Stable functions respect the paradigm with isometric information and order without the need for additional sorting and interchanging data objects. Stable isometric processors combine these benefits with potentially powerful effect.

Many pharmacometric data handling operations may be expressed succinctly using the functions described, all of which respect original record order. In fact, record order is central to the computational strategy of nth (etc.). In contrast, record order is computationally immaterial for reapply when used with a scalar function argument, such as mean, but will be consequential when reapply is used with “compound” isometric vector processor, e.g., csum and sort (but not nth).

In summary, stableMerge, nth (etc.) and reapply promote strategic data handling for pharmacometrics by enabling succinct, expressive, rapid data assembly techniques that respect record order.

References


Results

stableMerge

Pharmacometric data handling frequently requires a left-join, e.g., adding subject-level demographic variables to a table ordered by subject and time. In base R, left-joins are achieved using merge (x, y, all.x=TRUE, all.y=FALSE). However, the R help for merge only clearly indicates that record storage order must be non-informative. In contrast, many pharmacometric analyses use data formats with highly informative record order. The relational database model [1] specifies that record storage order must be non-informative. In contrast, many pharmacometric analyses use data formats with highly informative record order. The relational database model [1] specifies that record storage order must be non-informative. In contrast, many pharmacometric analyses use data formats with highly informative record order.

#### Conclusions

In summary, stableMerge, nth and reapply implicitly apply a selection-function to records in an indexed vector, repeating the scalar result for all elements in the cell. Using reapply(x, INDEX, FUN, ...), a function is named explicitly, and need not return a scalar value. The function reapply applies a function to each cell of a vector, as specified by levels of the index. However, reapply repeats each sub-result as necessary to match the number of input elements per cell, and returns the global result as a vector in an order corresponding to the original index. The result is a highly flexible, compact mechanism for manipulation of indexed vectors.

For example, consider a table of blood pressure data. We wish to compute missing MMHG using list, and add columns for baseline and mean, per ID and DIA. The following strategic and conventional methodologies give identical output, but the former is more compact, expressive, and understandable.

#### Strategic

```r
call(x)

s <- c(4,6,8,10)

x <- do.call(x, s)

s <- do.call(x, s)
```

#### Conventional

```r
s <- c(4,6,8,10)

x <- do.call(x, s)
```

Conclusions

When the arguments and values of stableMerge are table (class data.frame), the arguments x and y of table, where, within, ... may be vectors. In a table-based assembly context, use of nth is equivalent to a subset-and-left-join operation. The function nth returns, for each position in x, the nth element of x, optionally limiting candidate elements by where and optionally breaking the evaluation across subsets, as specified by within. The argument x is 0, returning all x; or negative, which counts instances from the end of the vector (or subset). If x is n, all elements are returned. "Intended" levels are handled correctly: if x is missing, nth returns subscripts (useful for subsetting other variables).

Using nth, one can compute for (instance) subject-wise, compartment-wise differences of each value in a vector from some arbitrary subset member.

```r
x <- c(1,2,3,4,5)

x

x[-1]
```

nth

Whereas the arguments and values of stableMerge are table (class data.frame), the arguments x and y of table, where, within, ... may be vectors. In a table-based assembly context, use of nth is equivalent to a subset-and-left-join operation. The function nth returns, for each position in x, the nth element of x, optionally limiting candidate elements by where and optionally breaking the evaluation across subsets, as specified by within. The argument x is 0, returning all x; or negative, which counts instances from the end of the vector (or subset). If x is n, all elements are returned. "Intended" levels are handled correctly: if x is missing, nth returns subscripts (useful for subsetting other variables).

Using nth, one can compute for (instance) subject-wise, compartment-wise differences of each value in a vector from some arbitrary subset member.

```r
x <- c(1,2,3,4,5)

x

x[-1]
```

stableMerge

Pharmacometric data handling frequently requires a left-join, e.g., adding subject-level demographic variables to a table ordered by subject and time. In base R, left-joins are achieved using merge (x, y, all.x=TRUE, all.y=FALSE). However, the R help for merge only clearly indicates that record storage order must be non-informative. In contrast, many pharmacometric analyses use data formats with highly informative record order. The relational database model [1] specifies that record storage order must be non-informative. In contrast, many pharmacometric analyses use data formats with highly informative record order. The relational database model [1] specifies that record storage order must be non-informative. In contrast, many pharmacometric analyses use data formats with highly informative record order.

#### Conclusions

In summary, stableMerge, nth and reapply implicitly apply a selection-function to records in an indexed vector, repeating the scalar result for all elements in the cell. Using reapply(x, INDEX, FUN, ...), a function is named explicitly, and need not return a scalar value. The function reapply applies a function to each cell of a vector, as specified by levels of the index. However, reapply repeats each sub-result as necessary to match the number of input elements per cell, and returns the global result as a vector in an order corresponding to the original index. The result is a highly flexible, compact mechanism for manipulation of indexed vectors.

For example, consider a table of blood pressure data. We wish to compute missing MMHG using list, and add columns for baseline and mean, per ID and DIA. The following strategic and conventional methodologies give identical output, but the former is more compact, expressive, and understandable.

#### Strategic

```r
x <- c(4,6,8,10)

x <- do.call(x, s)
```

#### Conventional

```r
x <- c(4,6,8,10)

x <- do.call(x, s)
```

Conclusions

When the arguments and values of stableMerge are table (class data.frame), the arguments x and y of table, where, within, ... may be vectors. In a table-based assembly context, use of nth is equivalent to a subset-and-left-join operation. The function nth returns, for each position in x, the nth element of x, optionally limiting candidate elements by where and optionally breaking the evaluation across subsets, as specified by within. The argument x is 0, returning all x; or negative, which counts instances from the end of the vector (or subset). If x is n, all elements are returned. "Intended" levels are handled correctly: if x is missing, nth returns subscripts (useful for subsetting other variables).

Using nth, one can compute for (instance) subject-wise, compartment-wise differences of each value in a vector from some arbitrary subset member.

```r
x <- c(1,2,3,4,5)

x

x[-1]
```