Self-auditing Data Tables for R

Timothy Bergsma*, Scott Pivirotto

Metrum Research Group, LLC, Tariffville, CT, USA

METRUM RESEARCH GROUP

Objectives

Exclusions from pharmacometric analysis datasets should be well-documented for transparency. Literate programming techniques [1,2] preserve exclusion detail but generally lack both integration and specificity. Data flagging techniques are quite specific but are only applicable to format-compliant records.

To supplement existing documentation, we sought methods to summarize the impact of exclusions on subject and record counts and to capture the excluded records themselves – without significant increases in programming burden.

Methods

We implemented the R class `audited` [3] as a variant of `data.frame`. While supporting typical operations, methods for this class populate an internal transaction table and optionally preserve dropped records as an attribute. The transactions capture record counts, subject counts, and a descriptive label for each operation. Additional methods extract and report transactions; plot them using `igraph` [4]; and write the dropped records to an Excel workbook using `xlsx` [5]. We created a test assembly to illustrate use of self-auditing tables.

Results

Use of the `audited` class effectively summarized counts of dropped records and optionally captured the records themselves. Transaction tables were easily saved as files or visualized as directed graphs; captured records were written to workbook spreadsheets with minimal effort. Workflow required only minor modification; graph labels and worksheet names could be precisely controlled without secondary intervention. The visualizations of the test assembly (Figures 1, 2) effectively accounted for dropped subjects and records. Support for other operations, such as data joins and additions, emerged naturally as a property of the technique.

Illustration

We illustrate the use of `audited` by manipulating an example pharmacometric data set. The `audit` option identifies an aggregating data item used to assess subject counts (optional). The `artifact` option lists the transaction types for which changed records will be written to spreadsheet.

```
library( audited )
library( metrumrg )
options( audit = 'SUBJ' )
options( artifact = 'drop' )
```

We derive dosing and PK data subsets from built-in data `Theoph`, marking `DV:0` records as below-limit-of-quantitation (`BLQ`) and arbitrarily designating some records missing-at-random (`MAR`). Later, we treat subjects numbered 9 or greater as placebo.

```
y <- Theoph
names( y ) <- c( 'SUBJ', 'WT', 'RATE', 'TIME', 'DV' )
y$SUBJ <- as.numeric( as.character( y$SUBJ ))
y <- data.frame( y )
y$AMT <- with( y, signif( digits = 3, RATE * WT ))
y$BLQ <- as.integer( y$DV == 0 )
y$MAR <- as.integer(( y$DV * 100 ) %% 10 == 3 )
dose <- y[ y$TIME == 0, c( 'SUBJ', 'WT', 'TIME', 'AMT' )]
pk <- y[, c( 'SUBJ', 'WT', 'TIME', 'DV', 'MAR', 'BLQ' )]
pk$EVID <- 0
dose$EVID <- 1
```

```
head( dose, 3 )
```

```
   SUBJ   WT TIME AMT EVID
1     1 79.6    0 320    1
12    2 72.4    0 319    1
23    3 70.5    0 319    1
```

```
head( pk, 3 )
```

```
  SUBJ   WT TIME   DV MAR BLQ EVID
1    1 79.6 0.00 0.74   0   0    0
2    1 79.6 0.25 2.84   0   0    0
3    1 79.6 0.57 6.57   0   0    0
```

We combine pk and dosing data, classifying each as `audited` and re-identifying the result using `alias`.

```
pk <- as.audited( pk )
dose <- as.audited( dose )
x <- alias( pk + dose, id = 'data' )
```

The following cleaning operations do not alter record count and are not tracked.

```
x$BLQ[ is.na( x$BLQ )] <- 0
x$MAR[ is.na( x$MAR )] <- 0
```

The following removal operations affect record count and are therefore tracked.

```
x <- x - x[ !!x$BLQ,, id = 'BLQ' ] # requires metrumrg
#  x  <- x[ !x$BLQ,,  od = 'BLQ' ] # alternate syntax
x <- x - x[ !!x$MAR,, id = 'no sample']
x <- x - x[ x$SUBJ >= 9,, id = 'placebo']
```

The audit results are available as a directed graph, a transaction table (`data.frame`), or an Excel workbook.

Directed Graph

The transaction sequence is visualized using `plot` (Figure 1). Arrows in the directed graph suggest data flow. Gold, blue and red correspond to create, add, and drop operations in the transaction table. We supply a non-default format string that stacks vertex label, subject count, and record count within each vertex.

```
plot(x, format = '%l\n%a subj.\n%r rec.', proportion = 2 )
```

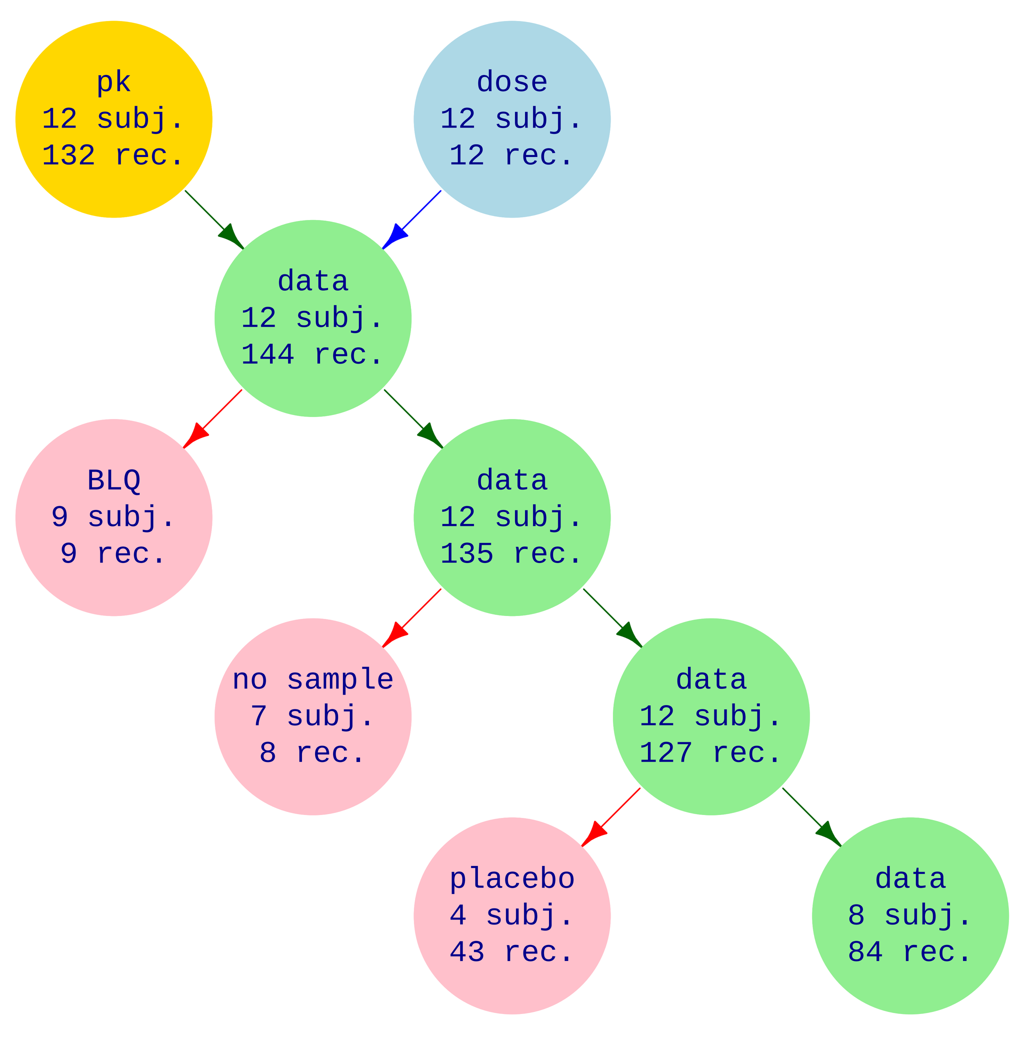

Figure 1: Disposition of Subjects and Records During Data Assembly

Default aesthetics can be overridden by the user. Graphs are scaled by default to fit the dimensions of the current device (*e.g.* vertex size decreases with transaction count). Below (Figure 2) is a visually more-neutral display of the same data, after independently excluding two more record sets.

```
x <- x - x[ x$SUBJ == 1,, id = 'subject 1' ]
x <- x - x[ x$SUBJ == 2,, id = 'subject 2' ]
plot(
+   x,
+   proportion = 1.3,
+   format = '%l\n%a subj.\n%r rec.',
+   vertex.shape = 'square',
+   color = FALSE,
+   modify.edge.label = NA
+ )
```

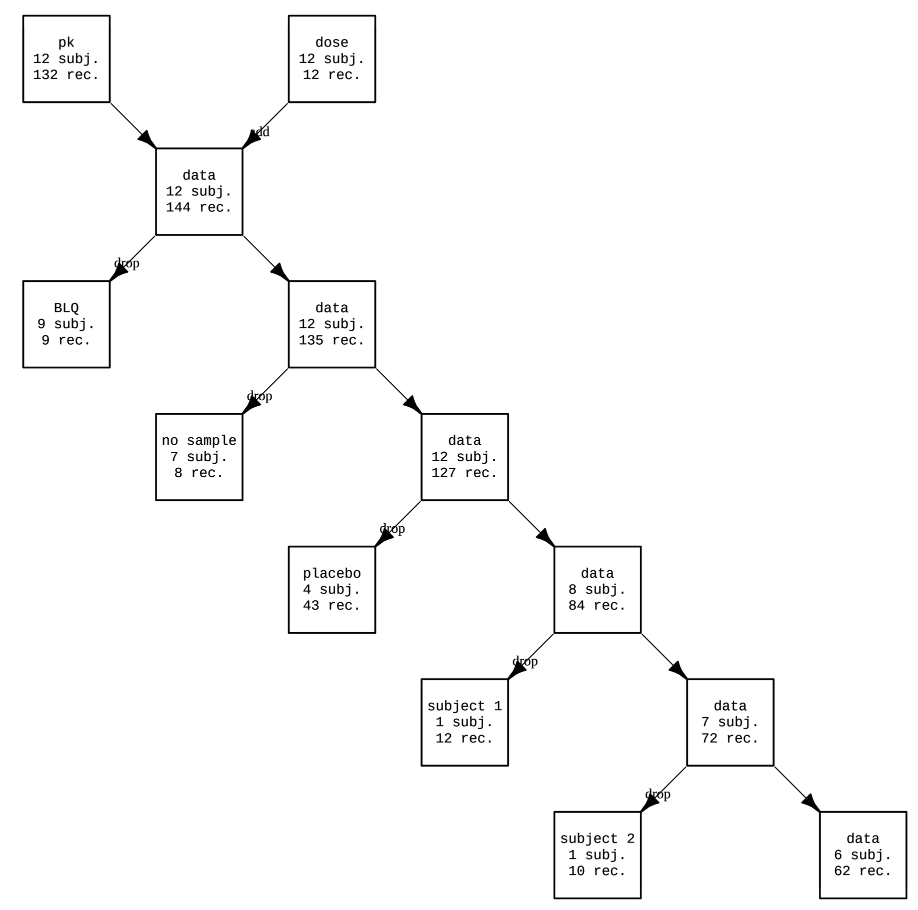

Figure 2: Alternative Aesthetic for Displaying Record Disposition

Transaction Table

The table of accumulated transactions can be extracted using `audit`. The result inherits `data.frame` and can be manipulated for other purposes.

```
audit(x)
```

	op	arg.id	arg.agg	arg.rec	res.id	res.agg	res.rec
1	create	pk	12	132	pk	12	132
2	add	dose	12	12	data	12	144
3	drop	BLQ	9	9	data	12	135
4	drop	no sample	7	8	data	12	127
5	drop	placebo	4	43	data	8	84
6	drop	subject 1	1	12	data	7	72
7	drop	subject 2	1	10	data	6	62

Excel Workbook

Dropped records are written to a workbook for further examination using `as.xlsx`. Worksheet names correspond to the labels of the directed graph.

```
as.xlsx(x, file = 'exclusions.xlsx')
```

Discussion

The common practice of scripting pharmacometric data assemblies helps assure the data quality on which the analysis depends. Among other features, the script provides critical details (intermediate scale) about how and why particular observations were selected for exclusion. However, considerable secondary effort may be required either to generate an integrated summary of exclusions (broad scale) or to recover the excluded observations themselves (fine scale).

Fine scale exclusion details may be supplied by techniques that capture console output, such as `R CMD Sweave` (.pdf) or `R CMD BATCH` (.Rout). However, console output is not easily manipulated and may not be suitable for large quantities of data. Alternatively one may retain the suspect records while flagging them for exclusion by the analysis tool (*e.g.* comment flag in NONMEM). But retention is impractical for records that do not comply with formatting expectations (*e.g.* missing date or time) and can create additional assembly burden (*e.g.* imputation of otherwise-moot covariate values).

The `audited` methodology supplies both broad and fine exclusion detail for very little additional effort. The transaction table and its corresponding directed graph provide broad detail: changes in subject and record counts across the entire assembly are summarized. The Excel workbook provides fine detail: the dropped records themselves are captured to a widely-supported file format that allows independent scrutiny. Since a separate worksheet is devoted to each drop event, no between-event conflicts of table structure occur.

For existing work, required interventions are modest. It suffices to reclassify audit targets using `as.audited`: class methods will supply informative event labels based on context. For clarity, record sets may be identified explicitly using the `id` argument (or `od` for alternative syntax). The user may want to register a column name to support calculation of subject counts, and may want to override default graph aesthetics.

Although the original intent was to document dropped records, support for other operations emerged naturally as the software developed. Currently methods exist that generate `create`, `add`, `drop`, `modify`, `transform`, or `merge` events, for which the directed graph has corresponding default aesthetics. The paradigm can be extended with methods for other generic functions.

Conclusions

With modest alteration of workflow, use of self-auditing data tables allowed generation of tables, graphs, and workbooks capturing the broad and fine detail of data exclusions during pharmacometric data assembly. Support for other table operations emerged naturally from the self-auditing paradigm.

References

[1] Knuth, DS. *Literate Programming.* (1984). The Computer Journal 27(2): 97-111.

[2] Leisch, F. Sweave: *Dynamic generation of statistical reports using literate data analysis.* (2002). In Wolfgang Härdle and Bernd Rönz, editors, Compstat 2002 - Proceedings in Computational Statistics: 575-580.

[3] `http://cran.r-project.org/web/packages/audited` Version 1.9

[4] Csardi G, Nepusz T. *The igraph software package for complex network research.* (2006). InterJournal (Complex Systems): 1695. `http://igraph.org`

[5] Dragulescu, AA. *xlsx: Read, write, format Excel 2007 and Excel 97/2000/XP/2003 files.* (2013). R package version 0.5.5. `http://CRAN.R-project.org/package=xlsx`