Self-auditing Data Tables for R
Timothy Bergsma*, Scott Pivirotto
Metrum Research Group, LLC, Tariffville, CT, USA

Objectives
Exclusions from pharmacometric analysis datasets should be well-documented for transparency. Lite versions of R
support exclusion detail but generally lack both integration and specificity. Data flaying techniques are quite specific
but are only available for 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` 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 transaction captures record counts, subject counts, and a descriptive label for each subject. Additional methods extract and report transactions; plot them using `graph4`, 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. Workflow required only minor modification; graph labels and worksheet names could be pre-written to workbook spreadsheets with minimal effort. Workflow required easily saved as files or visualized as directed graphs; captured records were optionally captured the records themselves. Transaction tables were self-auditing tables.

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 clean, classify and desecond operations in the transaction table. We supply a non-default format string that stacks vertex label, subject count, and record count within each vertex.

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

Illustration

```
head( data, 3 )
SUBJ WT TIME AMT EVID
1  1 9.6 0 0 300 1
12 2.72 4 0 319 1
23 3 7.05 0 319 1
```

```
head( pk, 3 )
SUBJ WT TIME AMT EVID
1  1 9.6 0 0 319 1
12 2.72 4 0 319 1
23 3 7.05 0 319 1
```

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

```
x <- x %>% \# requires metrurgy
  %>% \# requires auditing
  \# requires auditing
  \# requires auditing
  \# requires auditing
  \# requires auditing
```

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

```
x <- x %>% \# requires metrurgy
  \# requires auditing
  \# requires auditing
  \# requires auditing
  \# requires auditing
  \# requires auditing
```

Transaction Table
The table of accumulated transactions can be extracted using `audit()`. The resulting `data.frame` can be manipulated for other purposes.

```
up.arg.id arg.agg arg.rec res.id res.agg res.rec
1 create pk 12 132 9 9 12 133
1 add dose 12 12 data 12 144
1 drop BLQ 9 9 data 12 135
1 drop no sample 7 7 data 12 127
1 drop placebo 4 4 data 8 84
1 drop subject 1 1 data 7 72
1 drop subject 2 1 data 6 62
```

Discussion
The common practice of scripting pharmacometric data assemblies helps ensure 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 (spread sheet) or to recover the excluded observations themselves (fine scale).

Fine scale exclusion details may be supplied by techniques that capture console output, such as `cat` and `write()` but not `R CMD BATCH` (R Core). 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 data code) and can create additional assembly burden (e.g. imputation of otherwise-constant covariates).

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 across the entire assembly are summarized. The Excel workbook provides fine detail: the dropped records themselves are captured in a widely-supported file format that allows independent scrutiny. Since a separate workbook is devoted to each drop event, no between-event conflicts of table structure occur.

For existing work, required interventions are modest. It suffices to relas-
ch data targets using `audited` class methods will supply informative event labels based on context. For clarity, record sets may be identified ex-
licity using the `id` argument (or `as` for alternative syntax). The user may want to associate a column name to support documentation 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, drop, modify, attribute, transform or merge events, for which the directed graph has corresponding default aesthetics. The paradigm can be extended with methods for other genetic functions.

Conclusions
With modest alteration of workflow, use of self-auditing data tables ab-
holored 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
[2] Leisch, F. Sweave: Dynamic generation of statistical reports using literate data analysis. (2002). In Wolfgang Hardle and Bernd Ritzi,
[3] https://cran.r-project.org/web/packages/xlsx/Version 1.9
[5] https://cran.r-project.org/package=xlsx