Towards a Grammar for Processing Clinical Trial Data

by Michael J. Kane

Abstract The goal of this paper is to help define a path toward a grammar for processing clinical trials by a) defining a format in which we would like to represent data from standardized clinical trial data b) describing a standard set of operations to transform clinical trial data into this format, and c) to identify a set of verbs and other functionality to facilitate data processing and encourage reproducibility in the processing of these data. It provides a background on standard clinical trial data and goes through a simple preprocessing example illustrating the value of the proposed approach through the use of the **forceps** package, which is currently being used for data of this kind.

Introduction: On the use of historical clinical data

There are few areas of data science research that provide more promise to improve human quality-of-life and treat a disease than the development of methods and analysis in clinical trials. While adjacent, data-focused areas of biomedicine and health related research have recently seen increased attention, especially the analysis of real-world evidence (RWE) and electronic health records (EHR) in particular, clinical trial data maintains several distinct quality advantages, enumerated here.

1. Features and measurements are selected for their relevance. Unlike EHR's or other similar data, variables collected for a clinical trial are included because they are potentially relevant to the disease under consideration or the treatment whose efficacy is being analyzed. This makes the variable selection process considerably easier than that where data collection has not been designed for a targeted analysis of this type.
2. Data collection procedures are carefully prescribed. Clinical trial data is uniform in both which variables are collected and how they are collected. This ensures data quality across trial sites ensuring that variables are relatively complete as well as consistent.
3. Inclusion/Exclusion criteria define the population. Since RWE studies are observational, the populations they consider are not always well-understood due to bias in the collection process. On the other hand, clinical trial data sets are generally controlled and randomized, with well-documented inclusion and exclusion criteria.

Along with maintaining higher quality clinical trial data is more available and more easily accessible when compared to real-world data sources, which often require affiliations with appropriate research institutions as well as infrastructure and appropriate staff, including data managers, to extract data. By contrast, modern clinical trial data organizations allow users to quickly search and download thousands of trials, including anonymized patient-level information. These data sets tend to include control-arm data, which can be used to understand prognostic disease populations construct historical controls for existing trials. However, some also include treatment data, which can be used to characterize predictive patient subtypes for a given treatment, understand safety profiles for classes of drugs, and aid in the design of new trials. We note that, for oncology, Project Data Sphere (PDS, 2020) and outside of oncology, Immport (Imm, 2020) have been invaluable in our own experience by facilitating these types of analyses.

Clinical trial analysis data sets

During a clinical trial, patient-level data is collected in case report forms (CRFs). The format and data collected in these forms are prescribed in the trial design. These forms are the basis for the construction of analysis data sets and other documents that will be submitted to governing bodies, including the Food and Drug Association (FDA) and European Medicines Agency (EMA), for approval if the sponsor (party funding the trial) decides it is appropriate. The Clinical Data Interchange Standards Consortium (CDISC) (CDI, 2020) develops standards dealing with medical research data, including the submission of trial results. Adhering to these standards is necessary for a successful trial submission.

There are several data sets included with a submission that tend to be useful for analysis. This paper focuses on the Analysis Data Model (ADaM) data, which provides patient-level data, which has been validated and used for data derivation and analysis. An ADaM data set is itself composed of several data sets, including a Subject-Level Analysis Data Set (ADSL) holding analysis and treatment information. Other information, including baseline characteristics, demographic data, visit informa-

tion, etc., are held in the and Basic Data Structure (BDS) formatted data sets. Finally, adverse events are held in the Analysis Data Sets for Adverse Events (ADAE).

Challenges to analyzing these data sets

ADaM data for a clinical trial is generally made available as a set of SAS7BDAT (Shotwell et al., 2013) files. While neither the FDA nor the EMA require this format for submission nor do they requires the use of SAS (SAS Institute, 2020) for analysis, there is a heavy bias toward the data format and computing platform. This is partially because they are validated and approved by governing bodies and because a large effort has gone into their use in submissions. Packages like **sas7bdat** (Shotwell, 2014) and, more recently, **haven** (Wickham and Miller, 2020) have gone a long way to make these data sets easily accessible to R (R Core Team, 2012) users working with clinical trial data.

Despite the effort that has gone into defining a structure for the data as well as the tools implemented to aid in their analysis, the data sets themselves are not particularly easy to analyze for two reasons. First, the standard is not "tidy" as defined by Wickham et al. (2014). In particular, it is not required that each variable forms a column. In fact, multiple variables may be stored in one column, with another column acting as a key as to which variable's value is given. This case is often seen in the ADSL data set where a single column may primary and secondary endpoints. For data sets like these, the value variable is held in the Analysis Value (AVAL) if the corresponding variable is numeric, Analysis Value Character (AVALC) if the variable is a string, the Parameter Character Description (PARAMCD) column giving a shorted variable name, and the Parameter column providing a text description of the variable. As an example, consider the `adakiep.xpt` data set, which is provided as an example on the CDISC website and whose data is included in the supplementary material.

```
library(readr)

adakiep <- read_csv("adakiep.csv")
adakiep
```

```
#> # A tibble: 24 x 8
#>    USUBJID     PARAM      PARAMCD AVALC   ADY ADT        SRCDOM SRCSEQ
#>    <chr>       <chr>      <chr>   <chr> <dbl> <date>     <chr>   <dbl>
#>  1 XYZ-001-001 Death      DEATH   Y        85 2013-11-02 DS          1
#>  2 XYZ-001-001 Dialysis   DIALYS~ Y        80 2013-10-29 PR          2
#>  3 XYZ-001-001 eGFR 25 P~ EGFRDEC N        85 2013-11-02 <NA>       NA
#>  4 XYZ-001-001 Composite~ AKIEP   Y        80 2013-10-29 <NA>       NA
#>  5 XYZ-001-002 Death      DEATH   Y        82 2015-03-20 DS          1
#>  6 XYZ-001-002 Dialysis   DIALYS~ Y        73 2015-03-11 PR          2
#>  7 XYZ-001-002 eGFR 25 P~ EGFRDEC N        82 2015-03-20 <NA>       NA
#>  8 XYZ-001-002 Composite~ AKIEP   Y        73 2015-03-11 <NA>       NA
#>  9 XYZ-001-003 Death      DEATH   N        94 2010-10-12 DS          1
#> 10 XYZ-001-003 Dialysis   DIALYS~ Y        64 2010-09-12 PR          2
#> # ... with 14 more rows
```

The data set includes minimal information about the trial. However, we can infer that it is from a study focusing on kidney disease. There are four distinct endpoints, death, whether dialysis was needed, whether a 25% decrease in estimated glomerular filtration rate, indicating a decrease in kidney function. For analysis, these data will need to be re-arranged so that each endpoint has it's own column along with another column per endpoint indicating the trial day where the measurement was taken (from the ADY column).

The task of transforming these types of data into into appropriate analysis is complicated by the fact that there may be other files with relevant information with similar layout or layouts slightly more complicated if they include longitudinal information, for example. The rest of this paper focuses on shaping these types of data so that they can be quickly understood; they are amenable to many different types of analyses at the individual patient level; and they can be reformatted for an even larger class of analyses through a minimal set of verbs, including cohorting, which is introduced in this paper and is implemented in the **forceps** package (Kane, 2020). The package is currently in development and has not been released to CRAN. However, it has been tagged for prerelease on Github and can be installed with the following code.

```
devtools::install_github("kaneplusplus/forceps@v0.0.5")
```

The next section specifies the target data shape, which can be thought of as a restriction on the tidy format. The following section specifies the steps needed to prepare clinical trial data so that it conforms

to this restriction and includes an anonymized trial example. The final section provides a roadmap if near-term development as well as directions for enhancements and integration of the larger R ecosystem.

A tidy representation for a consolidated analysis data set

Clinical trial data is collected to be used in an analysis that determines whether or not a treatment for a disease provides a benefit when compared to those receiving either a placebo or the standard of care. "Benefit" is quantified by one or more endpoints, defined before the trial starts (in the *design*), which are compared across arms (treatment and placebo) at the conclusion of the trial using a statistical test. These data provide a wealth of information, and their usefulness extends well beyond the scope of the trial. For example, they can also be used to understand prognostic characteristics of the disease-population; they can be used to create a "historical control' ' for another trial; they can be used to identify patients' characteristics associated with better outcomes, etc.

As shown in the previous section, while ADaM-formatted data is structured, the structure does not lend itself to analysis without first performing some data transformations. We propose that the result of these transformations is a single data set with the following characteristics.

1. Each row corresponds to a single patient.
2. A variable with one value per patient should be included as a column variable.
3. Longitudinal, time series, or repeated measures data should be stored as an embedded `data.frame` per subject.

Data conforming to these characteristics provide several advantages to ADaM data sets. First, they are oriented towards trial analysis. Essentially, trials compare response rates between treatment and control arms. Having those values coded as their own variables in a single data set minimizes the complexity and effort that would otherwise go into extracting data from multiple files, cleaning them and joining them. Second, it minimizes the reshaping effort for other types of analyses. For example, response rates are often analyzed by the site at which patient measurements were taken in order to check for certain types of enrollment heterogeneity. The described patient-centric format can be transformed into a site-centric format by nesting or grouping on a site variable followed by the extraction of site-specific features and analyses, which can then be compared across sites. Transforming between these formats requires a single operation. Likewise, the patient-centric format can be transformed to a patient-longitudinal format by unnesting on the embedded variable holding the relevant longitudinal information. Third and finally, creating a single patient-centric data set minimizes the chance of inconsistent analyses. Primary and secondary analyses often use similar variables and may require similar preprocessing. If these preprocessing steps are performed separately for parallel analyses, then the probability that at least one of them contains an error in these steps is greater than when a validated patient-centric data set is created. It also makes it easier to provide provenance for an analysis if they are dependent on the same preprocessed data.

Processing ADaM data to reach the tidy representation

This section provides an example of how to use the functionality provided in the **forceps** package, in the order that the operations take place. The data set is provided with the package, and the variable names are taken from several example lung cancer studies. The data set has been significantly reduced in size, and some values and variable names have been preprocessed. This allows the example to remain easy to follow. It also allows us to illustrate the formation of a patient-centric data set in a single pass. In practice, this is often an iterative process, requiring several revisions as bugs are found, and hypotheses change.

The data sets used are as follows, and the task will be to create a patient-centered data set as described above.

1. `lc_adverse_events` - adverse events longitudinal data.
2. `lc_biomarkers` - patient biomarkers.
3. `lc_demography` - patient demographic information.
4. `lc_adsl` - response data.

Creating the data dictionary

SAS ADaM formatted data sets generally include extra information about variables, including a short description of each of the variables and possibly formatting information. The **haven** package

keeps this as attributes of each of the columns of a `tibble` that is read from these files. The **forceps** package is capable of extracting this meta-information to create a `tibble` that can be used as a data dictionary using the `consolidated_describe_data()` function, shown below. In practice, we have found it helpful as a starting for a fuller description of the data and often add columns to further categorize individual variables for analyses.

```
library(forceps)

data(lc_adverse_events)
data(lc_biomarkers)
data(lc_demography)
data(lc_adsl)

consolidated_describe_data(lc_adverse_events,
                           lc_biomarkers,
                           lc_demography,
                           lc_adsl)
```

```
#> # A tibble: 27 x 5
#>    var_name      type      label              format_sas data_source
#>    <chr>         <chr>     <chr>              <chr>      <chr>
#>  1 usubjid       double    Randomization Code <NA>       lc_adverse_e~
#>  2 ae            character AE Preferred Term  character  lc_adverse_e~
#>  3 ae_type       character System Organ Clas~ character  lc_adverse_e~
#>  4 grade         integer   Adverse Event Gra~ numeric    lc_adverse_e~
#>  5 ae_day        double    Days From First D~ <NA>       lc_adverse_e~
#>  6 ae_duration   double    Adverse Event Dur~ numeric    lc_adverse_e~
#>  7 ae_treat      logical   Was the Adverse E~ logical    lc_adverse_e~
#>  8 ae_count      integer   Total Patient Adv~ integer    lc_adverse_e~
#>  9 usubjid       double    Randomization Code <NA>       lc_biomarkers
#> 10 egfr_mutation character EGFR Mutation +ve~ character  lc_biomarkers
#> # ... with 17 more rows
```

Cohorting

The data dictionary (or data description) provides a summary of the variable types and information held by variables in each of the data sets. Some data sets will include repeated, longitudinal, or time series information about individual patients, like `lc_adverese_events` in our example. Consolidating data sets like these into a single, patient-centric data set generally involves three distinct operations. The first can be thought of as `pivot_wider()` operations that take columns composed of multiple variables and spread them across new columns in the data set. The second takes the data set and `nest()`'s the data so that the the resulting data set contains time-varying data embedded in a `data.frame` variable and those variables that are repeated appear once per patient in the new variables. This verb, which is referred to as `cohort()` in the package, takes the variable to cohort on (the `usubjid` in the example below), checks for values that are repeated by the subject identifier (`ae_count` in the example below) and those that are not, and handles the nesting appropriately. A final operation may be applied to the patient-level embedded `data.frame` objects to extract other features that will be used in subsequent analyses.

```
library(dplyr)

data(lc_adverse_events)

lc_adverse_events %>% head()
```

```
#> # A tibble: 6 x 8
#>   usubjid ae     ae_type    grade ae_day ae_duration ae_treat ae_count
#>     <dbl> <chr>  <chr>      <int>  <dbl>       <dbl> <lgl>       <int>
#> 1    1003 BURNI~ NERVOUS S~     1     27           4 FALSE          15
#> 2    1003 CONST~ GASTROINT~     2      4           4 TRUE           15
#> 3    1003 DEPRE~ PSYCHIATR~     2     66          NA FALSE          15
#> 4    1003 BACK ~ MUSCULOSK~     2     27          NA TRUE           15
#> 5    1003 DYSUR~ RENAL AND~     2      1           3 TRUE           15
#> 6    1003 SKIN ~ SKIN AND ~     1      5          26 FALSE          15
```

```
lc_adverse_events <- lc_adverse_events %>%
  cohort(on = "usubjid", name = "ae_long")

lc_adverse_events %>% head()
```

```
#> # A tibble: 6 x 3
#>   usubjid ae_count ae_long
#>     <dbl>    <int> <list>
#> 1    1003       15 <tibble [15 x 6]>
#> 2    1005       19 <tibble [19 x 6]>
#> 3    1006       11 <tibble [11 x 6]>
#> 4    1009       12 <tibble [12 x 6]>
#> 5    1014        5 <tibble [5 x 6]>
#> 6    1018       10 <tibble [10 x 6]>
```

Identifying conflicts and redundancies

After cohorting, each of the data sets is in the specified format, and we are almost ready to combine them. It is important to first check to see if there are variables that are repeated across the individual data sets and detect conflicts. While ADaM data sets *should* be free of conflicts and redundancies, we have observed multiple cases where this is not true. In order to identify these issues, the `duplicate_vars()` function is provided. The function checks the column names of each of the data sets with those of other column names. The object returned is a named list where the name corresponds to the variable that is repeated. Each list element returns a `tibble`, joined by the `on` parameter with columns corresponding to the `on` variable, the duplicated variable, the data sets where the duplicated variable appears. The example below shows that the `chemo_stop` variable appears in the `demography` and `adsl` data sets. Furthermore, we can see that the values in each of the data sets are different by looking at the correspondence between the `demography` and `adsl` columns. To fix this and move on, we will remove the variable from the `demography` data set.

```
data(lc_adsl)
data(lc_biomarkers)
data(lc_demography)
data_list <- list(demography = lc_demography,
                  biomarkers = lc_biomarkers,
                  adverse_events = lc_adverse_events,
                  adsl = lc_adsl)
duplicated_vars(data_list, on = "usubjid")
```

```
#> $chemo_stop
#> # A tibble: 558 x 4
#>    usubjid var        demography            adsl
#>      <dbl> <chr>      <chr>                 <chr>
#>  1    1003 chemo_stop patient discontinued  adverse events
#>  2    1005 chemo_stop treatment ineffective adverse events
#>  3    1006 chemo_stop <NA>                  treatment ineffective
#>  4    1009 chemo_stop treatment ineffective <NA>
#>  5    1014 chemo_stop <NA>                  adverse events
#>  6    1018 chemo_stop treatment ineffective treatment ineffective
#>  7    1023 chemo_stop <NA>                  adverse events
#>  8    1025 chemo_stop adverse events        adverse events
#>  9    1030 chemo_stop adverse events        adverse events
#> 10    1033 chemo_stop adverse events        treatment ineffective
#> # ... with 548 more rows
```

```
data_list$demography <- data_list$demography %>%
  select(-chemo_stop)
```

Consolidating

The last step is to consolidate the data sets into a single one. This is accomplished by reducing the `data_list` using full joins, along with some extra checking. The `consolidate()` function wraps this functionality. The result, conforming to the provided format, which can easily be used in the exploration and analysis stage.

```
consolidate(data_list, on = "usubjid")
```

```
#> # A tibble: 558 x 18
#>    usubjid site_id sex    refractory   age egfr_mutation smoking ecog
#>      <dbl>   <int> <chr>  <lgl>      <dbl> <chr>         <chr>   <chr>
#>  1    1003       1 male   FALSE         51 negative      former~ ambu~
#>  2    1005       4 female TRUE          44 negative      former~ ambu~
#>  3    1006       2 male   TRUE          22 negative      former~ ambu~
#>  4    1009       8 male   FALSE         44 <NA>          unknown ambu~
#>  5    1014       6 male   TRUE          76 <NA>          former~ ambu~
#>  6    1018      10 female TRUE          35 positive      former~ ambu~
#>  7    1023       6 female TRUE          73 <NA>          former~ ambu~
#>  8    1025       7 male   FALSE         71 <NA>          never ~ ambu~
#>  9    1030       5 female TRUE          20 <NA>          unknown ambu~
#> 10    1033       6 female TRUE          55 <NA>          unknown ambu~
#> # ... with 548 more rows, and 10 more variables: prior_resp <chr>,
#> #   ae_count <int>, ae_long <list>, best_response <chr>,
#> #   pfs_days <dbl>, pfs_censor <dbl>, os_days <dbl>, os_censor <dbl>,
#> #   chemo_stop <chr>, arm <chr>
```

Direction: An integrated approach to processing clinical data

As stated before, the goal of this paper is to help define a path toward a grammar for processing clinical trials by a) defining a format in which we would like to represent data from standardized clinical trial data b) describing a standard set of operations to transform clinical trial data into this format, and c) to identify a set of verbs and other functionality to facilitate data processing of this kind and encourage reproducibility of these steps. Admittedly, this only serves to mitigate the process of preparing these types of data for exploration and analysis. Clinical trial data generally contains many more variables than what was presented, and each of these data sets comes with its own set of "quirks" and other challenges. However, it does serve to make the data preparation better defined and propose a path toward standardization of both the processed data set format as well as the operations to achieve that goal.

Along with further development towards those ends, there is a plethora of development that can be done to provide an integrated data processing experience. For example, the define.xml file, which appears alongside ADaM data sets, gives better descriptions of the variables as well as the variable values. Tools to integrate these data into the construction of the data dictionary would go a long way towards orienting researchers with the data contained and help them more quickly formulate analyses. Packages like **lumberjack** (van der Loo, 2020) could enhance and augment data preprocessing steps by keeping better track of when data are being removed and how they are being manipulated. The artifacts accumulated could then be used by packages such as **ggconsort** (Higgins, 2020) to provide consort diagrams of how patients progress through the trial and how data progresses through preprocessing. In the longer term, these advancements can provide better data provenance, more reproducible processing, quicker debugging of problems in the processing stage, and give rise to more effective and convenient tools for summarizing trial data.

Bibliography

Clinical data interchange standards consortium. https://www.cdisc.org/, 2020. Accessed: 2020-10-25. [p1]

Immport: Bioinformatics for the future of immunology. https://www.immport.org/home, 2020. Accessed: 2020-10-25. [p1]

Project data sphere: Convener, collaborator, catalyst in the fight against cancer. https://www.projectdatasphere.org/, 2020. Accessed: 2020-10-25. [p1]

P. Higgins. *ggconsort: Creates CONSORT Diagrams for RCTs*, 2020. URL https://github.com/higgi13425/ggconsort. R package version 0.0.0.9000. [p6]

M. J. Kane. *forceps: A grammar for manipulating clinical trial data*, 2020. URL https://github.com/kaneplusplus/forceps. R package version 0.0.1. [p2]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. URL http://www.R-project.org/. ISBN 3-900051-07-0. [p2]

SAS Institute. *Base SAS 9.4 procedures guide*. SAS Institute, 2020. [p2]

M. Shotwell. *sas7bdat: SAS Database Reader (experimental)*, 2014. URL https://CRAN.R-project.org/package=sas7bdat. R package version 0.5. [p2]

M. S. Shotwell, C. Cummins, S. Header, S. Pages, S. Subheaders, and S. P. B. Data. Sas7bdat database binary format, 2013. [p2]

M. van der Loo. Monitoring data in r with the lumberjack package. *Journal of Statistical Software*, page Accepted for publication, 2020. URL https://CRAN.R-project.org/package=lumberjack. [p6]

H. Wickham and E. Miller. *haven: Import and Export 'SPSS', 'Stata' and 'SAS' Files*, 2020. URL https://CRAN.R-project.org/package=haven. R package version 2.3.1. [p2]

H. Wickham et al. Tidy data. *Journal of Statistical Software*, 59(10):1–23, 2014. [p2]

Michael J. Kane
Yale University
60 College Street
New Haven, CT 06510, USA
michael.kane@yale.edu