Paper PP20

CDISC Analysis Results Data with the R Package {cards} Including Integrations with {gtsummary}

Daniel D. Sjoberg, Genentech, South San Francisco, California, USA

ABSTRACT

The CDISC Analysis Results Data (ARD) Model is an emerging standard for encoding statistical analysis outcomes in a machine-readable format. Its primary objective is to streamline the processes of automation, ensuring reproducibility, promoting reusability, and enhancing traceability.

The {cards} R package offers a range of functions for ARD generation, from basic univariate summaries like means and tabulations to complex multivariable summaries encompassing regression models and statistical tests.

The package includes functionalities to represent results in various formats, including JSON and YAML. Thanks to its flexible structures, the {cards} package can be harnessed in diverse applications, such as generating tables for regulatory submissions and conducting quality control checks on existing tables. Furthermore, the {cards} ARD object can be accessed through a REST API, allowing writers to dynamically incorporate table results into reports.

While {cards} calculates statistics and stores them in a structured object, it cannot present those results; this, however, is where the {gtsummary} package shines. The {gtsummary} package offers a modular framework to construct summary tables. It is the most widely used package for summary tables in the R ecosystem, and won the American Statistical Association's 2021 award for Innovation in Statistical Programming and Analytics. The {gtsummary} package is currently being refactored to utilize {cards} as its backend, which will allow users to both extract an ARD object from a {gtsummary} table and use an ARD object to construct a {gtsummary} table. The {cards} and {gtsummary} packages stand as robust and versatile tools, poised to assist in a multitude of analytical endeavors.

INTRODUCTION

The CDISC Analysis Results Standard (ARS) Model aims to address the current inefficiencies in how analysis results are reported in clinical trials. These results are often presented in static PDF-based reports that are difficult to navigate, lack standardization, and hinder automation, traceability, and reusability. Two key components of the ARS Model are the Analysis Results Metadata Technical Specification (ARM-TS) and Analysis Results Dataset (ARD).

The ARM-TS defines the metadata needed to prospectively describe and organize analysis results. This includes information about the study design, analysis methods, and results themselves. The ARD is a standardized format for exchanging analysis results data. It provides a predictable and consistent structure that makes it easy to share and reuse results between different systems and organizations.

At the heart of the ARS Model lies the Analysis Results Dataset, a standardized format for exchanging and storing analysis results. Imagine it as a neatly organized warehouse where all the crucial information from your clinical trial analysis resides, readily accessible and easy to interpret. The {cards} R package creates ARD from observed data sets, and provides utilities for working with these objects.

R PACKAGE {cards}

The simplest way to introduce the {cards} R package is with an example of its most basic functionality. In the example below, we are using the ADSL example data set that is included in the package and we are calculating basic summary statistics for continuous variables "AGE" and "BMIBL".

```
library(cards)

ADSL |>
  ard_continuous(by = ARM, variables = c(AGE, BMIBL))

{cards} data frame: 48 x 10

   group1 group1_level variable stat_name stat_label statistic
1     ARM      Placebo      AGE         N          N        86
2     ARM      Placebo      AGE      mean       Mean    75.209
```

```
3     ARM      Placebo      AGE        sd         SD      8.59
4     ARM      Placebo      AGE    median     Median        76
5     ARM      Placebo      AGE       p25  25th Per…        69
6     ARM      Placebo      AGE       p75  75th Per…        82
7     ARM      Placebo      AGE       min        Min        52
8     ARM      Placebo      AGE       max        Max        89
9     ARM      Placebo    BMIBL         N          N        86
10    ARM      Placebo    BMIBL      mean       Mean    23.636

ℹ 38 more rows
ℹ Use `print(n = ...)` to see more rows
ℹ 4 more variables: context, statistic_fmt_fn, warning, error
```

A few items to note from this result:

- The default statistics returned are N, Mean, Standard Deviation, Median, 25th and 75th percentiles, and the minimum and maximum: these can be modified to use *any* univariate statistic whether that function is user-defined or from another package.
- These results are calculated by the treatment arm. There is, however, no requirement to include the `ard_continuous(by)` argument.
- Any unobserved levels of the `by=` column(s), whether that is unobserved combinations of the by= variables or unobserved factor levels, will appear in the returned ARD table.
- The results are returned in a structured data frame common among all ard_*() functions.

There exists similar functionality for categorical data.

```
ard_categorical(ADSL, by = ARM, variables = AGEGR1)

{cards} data frame: 27 x 11

   group1 group1_level variable variable_level stat_name stat_label statistic
1     ARM      Placebo   AGEGR1            <65         n          n        14
2     ARM      Placebo   AGEGR1            <65         N          N        86
3     ARM      Placebo   AGEGR1            <65         p          %     0.163
4     ARM    Xanomeli…   AGEGR1            <65         n          n        11
5     ARM    Xanomeli…   AGEGR1            <65         N          N        84
6     ARM    Xanomeli…   AGEGR1            <65         p          %     0.131
7     ARM    Xanomeli…   AGEGR1            <65         n          n         8
8     ARM    Xanomeli…   AGEGR1            <65         N          N        84
9     ARM    Xanomeli…   AGEGR1            <65         p          %     0.095
10    ARM      Placebo   AGEGR1            >80         n          n        30

ℹ 17 more rows
ℹ Use `print(n = ...)` to see more rows
ℹ 4 more variables: context, statistic_fmt_fn, warning, error
```

As a result of the common structure between this result and the continuous variable summary above, these two data frames can be combined with `bind_ard()`. The `bind_ard()` function is similar to `dplyr::bind_rows()`, and includes a few structural checks found in our ARD data frames, e.g. if we combine two ARDs with duplicate statistics, the function will notify us that the ARD no longer contains unique statistics.

It is common that errors or warnings may be return by functions performing these calculations. The {cards} package will continue to return a data frame of the expected structure. Where a statistic that could not be calculate would have appeared, we will now see a NULL value and the error will be captured and returned as text in the "error" column.

```
mean_with_error <- function(x) {
  stop("There was an error calculating the mean.")
  mean(x)
}
```

```
ard_with_error <-
  ard_continuous(
    ADSL,
    variables = AGE,
    statistics = ~list(mean = mean_with_error)
  )
ard_with_error

{cards} data frame: 1 x 8

  variable   context stat_name stat_label statistic     error
1      AGE continuo…      mean       Mean      NULL There wa…

ℹ 2 more variables: statistic_fmt_fn, warning
```

The {cards} package exports many utilities for working with ARDs. The example below is a utility to print any errors or warnings that may have occurred while calculating the statistics.

```
print_ard_conditions(ard_with_error)

The following errors were returned while calculating statistics:
✖ For variable `AGE` and "mean" statistic: There was an error calculating the
  mean.
```

EXTRA ARDs WITH {cardx}

As indicated above, the {cards} package exports many utilities for working with ARD objects. Additionally, {cards} exports utilities for creating new `ard_*()` functions. The {cardx} package takes advantage of this infrastructure, and exports many other functions for creating more complex ARD objects.

Utilizing these utilities from {cards}, we can easily create a function to prepare the results from a t-test.

```
ADSL |>
  # keep two treatment arms for the t-test calculation
  dplyr::filter(ARM %in% c("Placebo", "Xanomeline High Dose")) |>
  cardx::ard_ttest(by = ARM, variable = AGE)

{cards} data frame: 14 x 9

   group1 variable context   stat_name stat_label statistic
1     ARM      AGE   ttest    estimate  Mean Dif…     0.828
2     ARM      AGE   ttest   estimate1  Group 1 …    75.209
3     ARM      AGE   ttest   estimate2  Group 2 …    74.381
4     ARM      AGE   ttest   statistic  t Statis…     0.655
5     ARM      AGE   ttest     p.value    p-value     0.513
6     ARM      AGE   ttest   parameter  Degrees …   167.362
7     ARM      AGE   ttest    conf.low  CI Lower…    -1.668
8     ARM      AGE   ttest   conf.high  CI Upper…     3.324
9     ARM      AGE   ttest      method     method Welch Tw…
10    ARM      AGE   ttest alternative  alternat… two.sided
11    ARM      AGE   ttest          mu    H0 Mean         0
12    ARM      AGE   ttest      paired  Paired t…     FALSE
13    ARM      AGE   ttest   var.equal  Equal Va…     FALSE
14    ARM      AGE   ttest  conf.level  CI Confi…      0.95

ℹ 3 more variables: statistic_fmt_fn, warning, error
```

The utilities allow us to return, not only the results of the t-test, but rows for each of the arguments. This allows us to both report the results, and also in a re-use case, know exactly how the results were calculated, e.g. assuming equal variances, the level of the confidence interval, etc.

JSON, YAML, REST API

The default structure of a {cards} object is data frame. But it is often needed to represent the results in other formats. The {cards} objects can easily be expressed in YAML and JSON formats.

```
ADSL |>
  ard_continuous(by = ARM, variables = c(AGE, BMIBL)) |>
  head(n = 1) |>
  as_nested_list() |>
  jsonlite::toJSON(pretty = TRUE)
```


```
{
  "variable": {
    "AGE": {
      "group1": {
        "ARM": {
          "group1_level": {
            "Placebo": {
              "stat_name": {
                "N": {
                  "statistic": [86],
                  "statistic_fmt": ["86"],
                  "warning": {},
                  "error": {},
                  "context": ["continuous"]
                }
      …
}
```


This JSON representation can easily be used to create a REST API using the {plumber} R package.

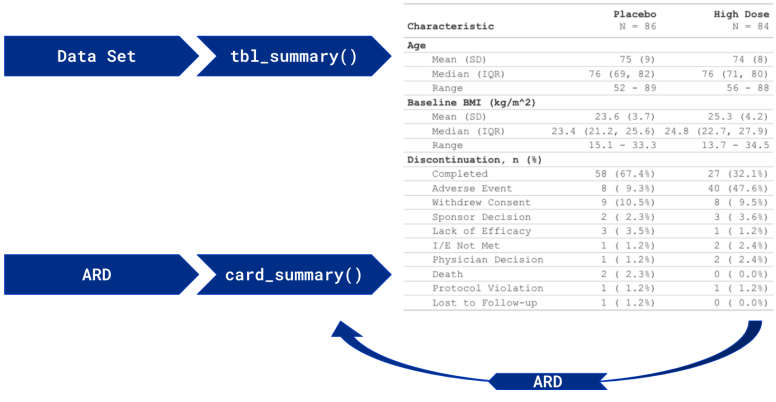
INTEGRATIONS WITH {gtsummary}

The {cards} package does not present results and this is where the {gtsummary} package shines. The {gtsummary} package offers a modular framework to construct summary tables. The {gtsummary} package is the most widely used package for summary tables in the R ecosystem, and won the American Statistical Association's 2021 award for Innovation in Statistical Programming and Analytics and it's currently being refactored with a {cards} backend.

After the update, ARDs will play two important roles in a {gtsummary}.

1. An ARD will be a byproduct of every {gtsummary} tables created.

2. The package will also support an ARD-first approach, ingesting the ARD and returning a {gtsummary} table.



Characteristic	Placebo N = 86	High Dose N = 84
Age		
Mean (SD)	75 (9)	74 (8)
Median (IQR)	76 (69, 82)	76 (71, 80)
Range	52 - 89	56 - 88
Baseline BMI (kg/m^2)		
Mean (SD)	23.6 (3.7)	25.3 (4.2)
Median (IQR)	23.4 (21.2, 25.6)	24.8 (22.7, 27.9)
Range	15.1 - 33.3	13.7 - 34.5
Discontinuation, n (%)		
Completed	58 (67.4%)	27 (32.1%)
Adverse Event	8 (9.3%)	40 (47.6%)
Withdrew Consent	9 (10.5%)	8 (9.5%)
Sponsor Decision	2 (2.3%)	3 (3.6%)
Lack of Efficacy	3 (3.5%)	1 (1.2%)
I/E Not Met	1 (1.2%)	2 (2.4%)
Physician Decision	1 (1.2%)	2 (2.4%)
Death	2 (2.3%)	0 (0.0%)
Protocol Violation	1 (1.2%)	1 (1.2%)
Lost to Follow-up	1 (1.2%)	0 (0.0%)

CONCLUSION

In conclusion, the CDISC Analysis Results Standard Model, with its focus on a standardized Analysis Results Dataset, promises to transform the way we handle and understand clinical trial results. By embracing this innovative approach, we

can unlock a future of efficient, transparent, and data-driven research, ultimately benefiting patients and accelerating the development of life-saving therapies.

{cards} https://github.com/insightsengineering/cards

{cardx} https://github.com/insightsengineering/cardx

{gtsummary} https://github.com/ddsjoberg/gtsummary

CONTACT INFORMATION

Daniel D. Sjoberg

Genentech, Member of the Roche Family

1 DNA Way, South San Francisco, CA 94080 USA

sjobergd@gene.com

www.danieldsjoberg.com