PharmaSUG 2023 - Paper SD-325

admiralonco

- the cross-company R package for Oncology admirers

Neharika Sharma, GlaxoSmithKline, Collegeville, PA, USA
Matthew Marino, GlaxoSmithKline, Collegeville, PA, USA

ABSTRACT

As companies across pharmaceutical industry are focusing on adopting R language for the creation of submission packages and data analysis, there becomes a growing requirement for coming out of silos and working collectively and efficiently towards developing functions and utilities that cater to the common industry-wide need. Through a collaborative effort between pharmaceutical companies like GlaxoSmithKline, Roche, Amgen, and Bristol Myers Squibb, Admiralonco was developed as an extension package to the Admiral package to assist in the creation of oncology specific CDISC compliant ADaM datasets like ADRS, ADTTE etc. In this presentation, we will elaborate the contents and usability of the package which is readily available to all statisticians and programmers and which we believe which could help one and all to efficiently increase the usage of R while analyzing clinical data. We will provide an overview of the capabilities of this project and demonstrate the application of some of the functions within the package. Briefly taking audience through the concepts of WRAP and Github, we plan to showcase examples of utilizing these functions in any project. We will also discuss the utilization of Agile working in a collaborative setup between different companies and the process that was followed in the development of this package. Finally, we will share our personal experience working on Admiralonco and discuss our learnings and hardships. Through this presentation, we hope the audience will learn about this package and consider benefiting from it.

INTRODUCTION

In the past, as drug development was still evolving, clinical trials data was analyzed and reported in conventional ways. The regulatory asks were limited, and clinical trial budgets allowed flexibility. However, the pharmaceutical industry, being heavily reliant on science and technology, is currently witnessing a fast change in the landscape of Clinical Reporting and seeing a trending organizational goal of using open-source languages like R. With this industry-wide interest, there are efforts being made at individual organizations to prepare for the shift in paradigm and gear up for the change. As companies in the pharmaceutical industry focus on adopting R for submission package creation and data analysis, there is a growing need to break out of silos and work collectively and effectively to develop functions and utilities that meet the common needs of the entire industry.

In this direction, admiral came into existence as the first cross-company collaboration to develop ADaM in R Asset library. It is an open-source collaboration initially between GSK and Roche to create ADaMs in R. Later around eighteen more companies joined the testing. The aim was to foster cross-industry crowdsourcing by promoting collaboration among individuals from different organizations, rather than working in isolated groups.

Building on the above, admiralonco is developed as the first Therapeutic area specific extension to the package of admiral family. It is a complementary open-source toolbox made of re-usable functions and utilities with individual purpose, wrapped together in a modular way. This modular nature lets you customize the creation of some of the most complicated oncology specific CDISC compliant ADaMs. Inheriting the modular nature from admiral has also helped in reusing some of the functions from there too. admiralonco has been developed in collaboration with four companies GSK, Roche, Amgen, and Bristol-Myers Squibb. The aim is to develop a package for universal adoption within the industry, thereby avoiding the need to duplicate the work at each individual company in solving the problem at hand. This paper is for all those admirers of the core conventions of Oncology therapeutic area, who are interested in enhancing the usage of R. The purpose of this paper is to provide an overview of the

package admiralonco, including its benefits and brief walkthrough of the available scripts. Our goal is to encourage testing and invite feedback to increase the usability and enhance the package. Embracing the changing trend of using the open-source languages while dynamically endorsing recent paradigm shift and upsurging organizational objectives, through this read, we hope to raise awareness of an industry wide collaboration opportunity of demystifying the use of R, jointly and unitedly.

MAIN GOAL OF ADMIRALONCO

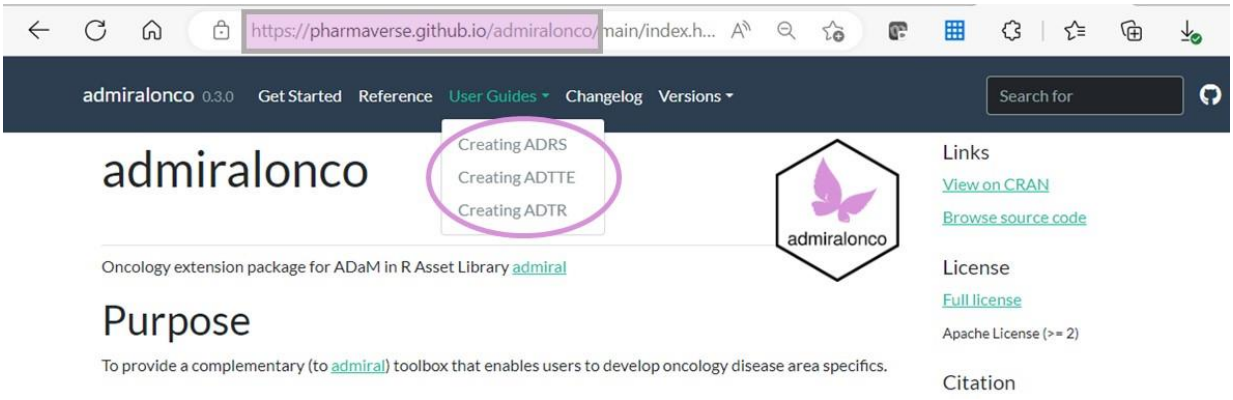
To provide an open source, modularized and complementary (to admiral) toolbox that enables users to develop oncology disease area specific ADaM datasets in R. One of the key aspects here is its development by the users for the users. It gives an entry point for all to collaborate, co-create and contribute to a harmonized approach of developing ADaMs in R across the pharmaceutical industry. The main idea is that an ADaM dataset is built by a simple to adopt sequence of derivations in a form of readable and easily constructible ADaM program scripts. Each derivation adds one or more variables or parameters to the processed dataset. This modular approach makes it easy to adjust code by adding, removing, or modifying derivations. Each derivation is a function call.

SCOPE

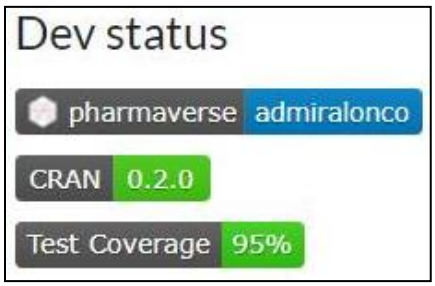
- Build a toolbox of re-usable functions and utilities to create oncology specific ADaM datasets in R in a modular manner.
- All functions are created based upon the ADaM Implementation Guide and aim to facilitate the programming of ADaM dataset standards.
- Initially the package will focus on the most common efficacy endpoint needs for solid tumor (using RECIST v1.1 response criteria), but over time we will look to add extra areas such as: endpoints for targeted indications, baseline disease characteristics, common sensitivity analyses and questionnaires.

ADMIRALONCO AT PHARMAVERSE GITHUB

To make usage of admiralonco easier and to facilitate learning, there is enormous documentation available around the package. A screenshot is shown here from the page on pharmaverse at github. The landing page helps you understand the scope of the package and how to go about installing the package. If you navigate to the get started page, it will show you how to start a script and use the admiralonco templates. The Reference tab has the details of all the functions and utilities that has been added as part of the package. There are also user guides available separately for each domain as highlighted in the screenshot, if you go there, you will see vignettes for step-by-step creation of an admiralonco based ADaM dataset, with some example data to demonstrate and make it easy to follow. Overall, with the information available on the page you should have all you need to get started and you should be all set!



TEST COVERAGE



Whenever an open-source package is talked about, we encounter a common question which is around the validation of the package. Hence the introduction to admiralonco wouldn't be complete without the mention of the test coverage of the package which is 95%. With a thorough testing, functions are setup to be follow CDISC and not violate any rules, ensuring quality of the package. Having said that, ultimately, it falls on the users/individual organizations to adjudicate whether that is sufficient or, further testing is required to ensure CDISC compliance.

BENEFITS OF USING ADMIRALONCO

Having introduced to admiralonco, it is of utmost importance to understand the benefits the package is bringing before we dive deep into the technical details of the package. To cater to this, lets understand the scope of area of utility and what new this package is bringing into our current world of clinical programming.

- In contrast to the traditional data analysis and reporting conventions, the pharmaceutical industry is now growing an interest towards the use of open-source languages such as R. In this area, admiralonco comes as an open-source R package, which is freely available to all organizations and individuals to use.
- The difficulty with most new techniques is usually the need for training. This package, admiralonco, comes with a huge amount of documentation available on the sites (illustrated ahead and detailed in the following sections), which makes it quite easy to follow and use without the need for formal training.
- In each organization we have a mixed pool of programmers, ranging from experienced programmers well versed in oncology conventions to new and curious minds willing to learn and work in oncology TA. As admiralonco is based on global guidelines such as RECIST 1.1, it is usable by programmers who are new to the therapeutic area of oncology and who do not yet have a thorough understanding of the oncology efficacy requirements. These scripts are written to follow all the standard guidelines and, with detailed documentation, provide an easier way to work with some of the complex conventions of oncology.
- As mentioned previously, this is an effective way to increase the use of R in any project and to save a lot of time in generating the most complicated ADaMs in oncology, which could have taken even longer in other programming languages like SAS.
- Finally, in different organizations we follow standard guidelines, but we still have many company-specific standards or methods for collecting and processing efficacy endpoints. It may therefore be intuitive to question the scope and usability of the package in all cases. The answer lies in the foundations of the package. As mentioned earlier, this package is developed using a modular approach, which means that the intention is not to automate the creation of ADaMs, but to provide users with a set of reusable and modifiable functions and utilities that allow them to feed into their own company-specific conventions or practices. Later in this document we will see how these scripts can handle the pre/post processing of data where any necessary changes can be made easily.

CONTENTS OF THE PACKAGE

This package is developed in an Agile way. This indicates that we have addressed the issue in hand by catering to the common needs first and will continue adding to the package over time. We have started with RECIST 1.1 which involves the most common efficacy endpoints in oncology. The package will grow over time.

CURRENT CONTENTS

The currently available scripts are:

- ADRS (ADaM dataset for Tumor Response Analysis)
- ADTTE (ADaM dataset for Time to Event Analysis)
- ADTR (ADaM dataset for Tumor Results Analysis)

In this section we will illustrate various functions that are developed as part of these scripts:

Functions related to derivations for adding Parameters:

1. derive_param_bor()

Calculates and adds a parameter for Best Overall Response (BOR) parameter without confirmation, as detailed below.

Records after PD can be removed using the source_pd and source_datasets arguments.

- All CR, PR and PD response records are considered for Best Overall Response.
- All SD or NON-CR/NON-PD records where ADT >= reference_date + ref_start_window are also considered for Best Overall Response.
- Subjects with ONLY an SD or NON-CR/NON-PD records where ADT < reference_date + ref_start_window are assigned a Best Overall Response of NE.
- The Best Response, from the records in steps 1 to 3, is then selected in the following order of preference: CR, PR, SD, NON-CR/NON-PD, PD, NE, MISSING
- The AVAL column is added and set using the aval_fun(AVALC) function
- The columns specified by the set_values_to parameter and records are added to the dataframe passed into the dataset argument

Note: Any responses of SD or NON-CR/NON-PD that occur before reference_date + ref_start_window are ignored in the calculation of BOR. All other responses are included in the calculation of BOR, irrespective of the number of days from the reference date.

Also Note: All columns from the input dataset are kept. For subjects with no records in the input dataset (after the filter is applied) all columns are kept from ADSL which are also in the input dataset. Columns which are not to be populated for the new parameter or populated differently (e.g., RSSTRESC, VISIT, PARCATy, ANLzzFL, ...) should be overwritten using the set_values_to parameter.

2. derive_param_clinbenefit()

Calculates and adds a parameter for Clinical Benefit. Clinical benefit/disease control is first identified by looking for subjects having response status, and then derived for subjects that have at least one evaluable non-PD response assessment prior to first PD (Progressive Disease) (i.e., responses inclusive of CR, PR, SD, and NON-CR/NON-PD) and after a specified amount of time from a reference date (ref_start_window).

Note: The user input values they wish to include when determining clinical benefit using the argument clinben_vals. The default values for this are CR, PR, SD, and NON-CR/NON-PD, as listed above. In the below example, eligible values be limited to CR and PR.

Example: clinben_vals <- c("CR", "PR")

- The input dataset (dataset) is restricted to the observations matching filter_source and to observations before or at the date specified by source_pd.

- This dataset is further restricted to include user-generated response assessments from clinben_vals or include response assessments of CR, PR, SD, and NON-CR/NON-PD, exclude missing response assessments, and exclude those less than ref_start_window after reference_date. The earliest assessment by ADT is then selected.
- The dataset identified by dataset in source_resp is restricted according to its filter argument. The variable corresponding to the date parameter of source_resp is considered together with ADT from the previous step.
- For the observations being added to dataset, ADT is set to the earlier of the first assessment date representing an evaluable non-PD assessment prior to first PD, or the date representing the start of response.
- For the observations being added to dataset, AVALC is set to
    - Y for those subjects in the dataset meeting the criteria for clinical benefit above
    - N for subjects not meeting the clinical benefit criteria in dataset or the dataset identified in source_resp
    - N for subjects present in dataset_adsl but not present in dataset or the dataset identified in source_resp.
- AVAL is derived using AVALC as input to the function specified in aval_fun.
- The variables specified by set_values_to are added to the new observations with values equal to the values specified in the same.
- The new observations are added to dataset. Variables held in common between dataset and dataset_adsl are kept for the new observations and are populated with their values from dataset_adsl.

3. derive_param_confirmed_bor()

Adds a Parameter for Confirmed Best Overall Response.

- The input dataset (dataset) is restricted to the observations matching filter_source and to observations before or at the date specified by source_pd.
- The following potential confirmed responses are selected from the restricted input dataset:
    - "CR": An assessment is considered as complete response (CR) if
        - AVALC == "CR",
        - there is a confirmatory assessment with AVALC == "CR" at least ref_confirm days after the assessment,
        - all assessments between the assessment and the confirmatory assessment are "CR" or "NE", and
        - there are at most max_nr_ne "NE" assessments between the assessment and the confirmatory assessment.
    - "PR": An assessment is considered as partial response (PR) if
        - AVALC == "PR",
        - there is a confirmatory assessment with AVALC %in% c("CR", "PR") at least ref_confirm days after the assessment,
        - all assessments between the assessment and the confirmatory assessment are "CR", "PR", "SD", or "NE",

        - there is no "PR" assessment after a "CR" assessment in the confirmation period,
        - there are at most max_nr_ne "NE" assessments between the assessment and the confirmatory assessment, and
        - if the accept_sd argument is set to TRUE, one "SD" assessment in the confirmation period is accepted. Otherwise, no "SD" assessment must occur within the confirmation period.
    - "SD": An assessment is considered as stable disease (SD) if
        - AVALC %in% c("CR", "PR", "SD") and
        - the assessment is at least ref_start_window days after reference_date.
    - "NON-CR/NON-PD": An assessment is considered as NON-CR/NON-PD if
        - AVALC = "NON-CR/NON-PD" and
        - the assessment is at least ref_start_window days after reference_date.
    - "PD": An assessment is considered as progressive disease (PD) if AVALC == "PD".
    - "NE": An assessment is considered as not estimable (NE) if
        - AVALC == "NE" or
        - AVALC %in% c("CR", "PR", "SD", "NON-CR/NON-PD") and the assessment is less than ref_start_window days after reference_date.
    - "ND": An assessment is considered as not done (ND) if AVALC == "ND".
    - "MISSING": An assessment is considered as missing (MISSING) if a subject has no observation in the input dataset.

      If the missing_as_ne argument is set to TRUE, AVALC is set to "NE" for these subjects.
- For each subject, the best response as derived in the previous step is selected, where "CR" is best and "MISSING" is worst in the order above. If the best response is not unique, the first one (with respect to ADT) is selected. If the selected record is from the input dataset, all variables are kept. If the selected record is from dataset_adsl, all variables which are in both dataset and dataset_adsl are kept.
- The AVAL variable is added and set to aval_fun(AVALC).
- The variables specified by the set_values_to parameter are added to the new observations.
- The new observations are added to input dataset.

4. derive_param_confirmed_resp() - Adds a Parameter for Confirmed Response

Adds a Parameter for Confirmed Response.

- The input dataset (dataset) is restricted to the observations matching filter_source and to observations before or at the date specified by source_pd.
- A subject is considered as responder if there is at least one observation in the restricted dataset with
    - AVALC == "CR",
    - there is a confirmatory assessment with AVALC == "CR" at least ref_confirm days after the assessment,

    - all assessments between the assessment and the confirmatory assessment are "CR" or "NE", and
    - there are at most max_nr_ne "NE" assessments between the assessment and the confirmatory assessment.
    - or at least one observation with
    - AVALC == "PR",
    - there is a confirmatory assessment with AVALC %in% c("CR", "PR") at least ref_confirm days after the assessment,
    - all assessments between the assessment and the confirmatory assessment are "CR", "PR", "SD", or "NE",
    - there is no "PR" assessment after a "CR" assessment in the confirmation period,
    - there are at most max_nr_ne "NE" assessments between the assessment and the confirmatory assessment,
    - if the accept_sd argument is set to TRUE, one "SD" assessment in the confirmation period is accepted. Otherwise, no "SD" assessment must occur within the confirmation period.
- For responders AVALC is set to "Y" and ADT to the first date where the response criteria are fulfilled. For all other subjects in dataset_adsl AVALC is set to "N" and ADT to NA.
- The AVAL variable is added and set to aval_fun(AVALC).
- The variables specified by the set_values_to parameter are added to the new observations.
- The new observations are added to input dataset.

5. derive_param_response()

Adds a Parameter Indicating If a Subject Had a Response before Progressive Disease

- The Date of the end of the assessment period (e.g., Progressive disease, as defined by pd_source) is added to the response dataset.
- The response dataset is restricted to observations occurring before ** or on ** the date of progressive disease.
- For each subject (with respect to the variables specified for the subject_keys parameter), the first observation (with respect to ADT) where the response condition (filter_source parameter) is fulfilled is selected.
- For each observation in dataset_adsl a new observation is created.
    - For subjects with a response AVALC is set to "Y", AVAL to 1, and ADT to the first date (ADT) where the response condition is fulfilled.
    - For all other subjects AVALC is set to "N", AVAL to 0 and ADT to NA.
- The variables specified by the set_values_to parameter are added to the new observations.
- The new observations are added to input dataset.

Advanced Functions:

Pre-Defined Time-to-Event Sources

tte_source objects defined by {admiralonco} that can be used as input for admiral::derive_param_tte()

- death_event
- lastalive_censor
- pd_event
- lasta_censor
- rand_censor
- trts_censor

To see the definition of the various objects simply print the object in the R console, e.g., print(death_event). For details of how to use these objects please refer to admiral::derive_param_tte().

Printing an object will display input dataset_name, filter (if applicable), date variable, and appropriate values for EVNTDESC, CNSDTDSC, SRCDOM, SRCVAR, and SRCSEQ.

Utility Functions:

Utilities for Formatting Observations

- aval_resp(): Map Character Response Values to Numeric Values

Utilities for Dataset Checking

- get_crpr_dataset() : Get CR Records Followed by PR That Led to a Prior Error

  Some admiralonco function check that the in the source records CR is not followed by PR and throw an error otherwise. The get_crpr_dataset() function allows one to retrieve the duplicate records that lead to an error.

  Note that the function always returns the dataset of duplicates from the last error that has been thrown in the current R session. Thus, after restarting the R sessions get_crpr_dataset() will return NULL and after a second error has been thrown, the dataset of the first error can no longer be accessed (unless it has been saved in a variable).

- signal_crpr() : Signal CR Records Followed by PR

Utilities for Filtering Observations

- filter_pd() : Filter up to First PD (Progressive Disease) Date

  The input dataset (dataset) is restricted by filter.

  For each subject, the first PD date is derived as the first date (source_pd$date) in the source pd dataset (source_datasets[[source_pd$dataset_name]]) restricted by source_pd$filter.

  The restricted input dataset is restricted to records up to first PD date. Records matching first PD date are included. For subject without any first PD date, all records are included.

Utilities for Catching Errors

- call_aval_fun() : Creates AVAL from AVALC by Calling User Function

  The new variable AVAL is set to aval_fun(AVALC).

FUTURE PLANS

As mentioned earlier, this package is being developed using an agile methodology, which means that it will continue to grow in the coming years. Over time, we plan to add additional areas to the scope, such as endpoints for targeted indications like iRECIST (immune RECIST) and the International Myeloma Working Group (IMWG) for response assessment in multiple myeloma. We have also started to explore the possibility of adding the derivation of the first anti-cancer therapy date. There are many other items on our list, such as the handling of common sensitivity analysis, etc. The order of implementation of these planned elements may change depending on our strategy and alignment with the subsequent admiral releases.

HOW TO USE THE PACKAGE

Having understood the contents of the package, next is to learn how to use it. To understand how to benefit from this package, lets dive deep into the coding. This section will have a step-by-step guide to create ADaM datasets using the scripts available as part of admiralonco. At every stage, a sample data is provided to illustrate the concept and implementation better.

INSTALLATION

The package is available from CRAN and can be installed by running

```
install.packages("admiralonco").
```

To install the latest development version of the package directly from GitHub use the following code:

```
if (!requireNamespace("remotes", quietly = TRUE)) {
  install.packages("remotes")
}
remotes::install_github("pharmaverse/admiralonco", ref = "devel")
```

STARTING A SCRIPT

For the oncology ADaM data structures, an overview of the flow and example function calls for the most common steps are provided by the following vignettes:

- Creating ADRS (ADaM dataset for Tumor Response Analysis)
- Creating ADTTE (ADaM dataset for Time to Event Analysis)
- Creating ADTR (ADaM dataset for Tumor Results Analysis)

{admiralonco} also provides template R scripts as a starting point. They can be created by calling use_ad_template() from {admiral}, e.g.,

```
library(admiral)
  use_ad_template(
    adam_name = "adrs",
    save_path = "./ad_adrs.R",
    package = "admiralonco"
  )
```

A list of all available templates can be obtained by list_all_templates() from {admiral}:

```
list_all_templates(package = "admiralonco")
#> Existing ADaM templates in package 'admiralonco':
#> • ADRS
#> • ADTR
#> • ADTTE
```

CREATING ADRS

Source: https://github.com/pharmaverse/admiralonco/blob/main/vignettes/adrs.Rmd

Example Script: https://github.com/pharmaverse/admiralonco/blob/main/inst/templates/ad_adrs.R

Introduction

This article describes creating an ADRS ADaM with common oncology endpoint parameters based on RECIST v1.1. Therefore, response values are expected as either CR, PR, SD, NON-CR/NON-PD, PD or NE. Examples are currently presented and tested using ADSL (ADaM) and RS, TU (SDTM) inputs. However, other domains could be used. The functions and workflow could similarly be used to create an intermediary ADEVENT ADaM.

Note: All examples assume CDISC SDTM and/or ADaM format as input unless otherwise specified.

Read in Data

To start, all data frames needed for the creation of ADRS should be read into the environment. This will be a company specific process. Some of the data frames needed may be ADSL, RS and TU.

For example, purpose, the SDTM and ADaM datasets (based on CDISC Pilot test data)—which are included in {admiral.test}—are used.

```
library(admiral)
library(admiralonco)
library(dplyr)
library(admiral.test)
library(lubridate)
library(stringr)

data("admiral_adsl")
data("admiral_rs")
data("admiral_tu")

adsl <- admiral_adsl
rs <- admiral_rs
tu <- admiral_tu
rs <- convert_blanks_to_na(rs)
tu <- convert_blanks_to_na(tu)
```

At this step, it may be useful to join ADSL to your RS domain. Only the ADSL variables used for derivations are selected at this step. The rest of the relevant ADSL would be added later.

```
adsl_vars <- exprs(RANDDT)
adrs <- derive_vars_merged(
  rs,
  dataset_add = adsl,
  new_vars = adsl_vars,
  by_vars = exprs(STUDYID, USUBJID)
)
```

USUBJID	RSTESTCD	VISIT	RSDTC	RANDDT
01-701-1015	OVRLRESP	WEEK 6	2014-02-12	2014-01-02
01-701-1015	OVRLRESP	WEEK 6	2014-02-12	2014-01-02
01-701-1015	OVRLRESP	WEEK 6	2014-02-12	2014-01-02
01-701-1015	OVRLRESP	WEEK 12	2014-03-26	2014-01-02
01-701-1015	OVRLRESP	WEEK 12	2014-03-26	2014-01-02

Pre-processing of Input Records

The first step involves company-specific pre-processing of records for the required input to the downstream parameter functions. Note that this could be needed multiple times (e.g., once for investigator and once for Independent Review Facility (IRF)/Blinded Independent Central Review (BICR) records). It could even involve merging input data from other sources besides RS, such as ADTR.

This step would include any required selection/derivation of ADT or applying any necessary partial date imputations, updating AVAL (e.g. this should be ordered from best to worst response), and setting analysis flag ANL01FL. Common options for ANL01FL would be to set null for invalid assessments or those occurring after new anti-cancer therapy, or to only flag assessments on or after date of first treatment/randomization, or rules to cover the case when a patient has multiple observations per visit (e.g. by selecting worst value). Another consideration could be extra potential protocol-specific sources of Progressive Disease such as radiological assessments, which could be pre-processed here to create a PD record to feed downstream derivations.

For the derivation of the parameters, it is expected that the subject identifier variables (usually STUDYID and USUBJID) and ADT are a unique key. This can be achieved by deriving an analysis flag (ANLzzFL). See Derive ANL01FL for an example.

The below shows an example of a possible company-specific implementation of this step.

Select Overall Response Records and Set Parameter Details

In this case we use the overall response records from RS from the investigator as our starting point. The parameter details such as PARAMCD, PARAM etc. will always be company-specific, but an example is shown below so that you can trace through how these records feed into the other parameter derivations.

```
adrs <- adrs %>%
  filter(RSEVAL == "INVESTIGATOR" & RSTESTCD == "OVRLRESP") %>%
  mutate(
    PARAMCD = "OVR",
    PARAM = "Overall Response by Investigator",
    PARCAT1 = "Tumor Response",
    PARCAT2 = "Investigator",
    PARCAT3 = "Recist 1.1"
  )
```

USUBJID	RSTESTCD	RSEVAL	VISIT	PARAMCD	PARAM	PARCAT1	PARCAT2	PARCAT3
01-716-1024	OVRLRESP	INVESTIGATOR	WEEK 6	OVR	Overall Response by Investigator	Tumor Response	Investigator	Recist 1.1
01-716-1024	OVRLRESP	INVESTIGATOR	WEEK 12	OVR	Overall Response by Investigator	Tumor Response	Investigator	Recist 1.1
01-716-1024	OVRLRESP	INVESTIGATOR	WEEK 18 (T)	OVR	Overall Response by Investigator	Tumor Response	Investigator	Recist 1.1
01-716-1024	OVRLRESP	INVESTIGATOR	WEEK 24	OVR	Overall Response by Investigator	Tumor Response	Investigator	Recist 1.1

Partial Date Imputation and Deriving ADT, ADTF, AVISIT etc.

If your data collection allows for partial dates, you could apply a company-specific imputation rule at this stage when deriving ADT. For this example, here we impute missing day to last possible date.

```
adrs <- adrs %>%
  derive_vars_dt(
    dtc = RSDTC,
    new_vars_prefix = "A",
```

```
    highest_imputation = "D",
    date_imputation = "last"
  ) %>%
  mutate(AVISIT = VISIT)
```

USUBJID	RSSTRESC	RSDTC	PARAMCD	PARAM	ADT	ADTF	AVISIT
01-701-1015	PD	2014-02-12	OVR	Overall Response by Investigator	2014-02-12		WEEK 6
01-701-1015	CR	2014-03-26	OVR	Overall Response by Investigator	2014-03-26		WEEK 12
01-701-1015	SD	2014-06-18	OVR	Overall Response by Investigator	2014-06-18		WEEK 24
01-703-1086	PD	2012-10-13	OVR	Overall Response by Investigator	2012-10-13		WEEK 6
01-703-1086	PD	2012-11-27	OVR	Overall Response by Investigator	2012-11-27		WEEK 12

Derive AVALC and AVAL

Here we populate AVALC and create the numeric version as AVAL (ordered from best to worst response). This ordering is already covered within our RECIST v1.1 parameter derivation functions, and so changing AVAL here would not change the result of those derivations.

```
adrs <- adrs %>%
  mutate(
    AVALC = RSSTRESC,
    AVAL = aval_resp(AVALC)
  )
```

USUBJID	RSSTRESC	PARAMCD	PARAM	AVISIT	AVALC	AVAL
01-701-1015	PD	OVR	Overall Response by Investigator	WEEK 6	PD	5
01-701-1015	CR	OVR	Overall Response by Investigator	WEEK 12	CR	1
01-701-1015	SD	OVR	Overall Response by Investigator	WEEK 24	SD	3
01-703-1086	PD	OVR	Overall Response by Investigator	WEEK 6	PD	5
01-703-1086	PD	OVR	Overall Response by Investigator	WEEK 12	PD	5

Derive ANL01FL

When deriving ANL01FL this is an opportunity to exclude any records that should not contribute to any downstream parameter derivations. In the below example this includes only selecting valid assessments and those occurring on or after randomization date. If there is more than one assessment at a date, the worst one is flagged.

```
adrs <- adrs %>%
  restrict_derivation(
    derivation = derive_var_extreme_flag,
    args = params(
      by_vars = exprs(STUDYID, USUBJID, ADT),
      order = exprs(AVAL, RSSEQ),
      new_var = ANL01FL,
      mode = "last"
    ),
    filter = !is.na(AVAL) & ADT >= RANDDT
  )
```

USUBJID	RANDDT	PARAMCD	PARAM	ADT	AVISIT	AVALC	ANL01FL
01-701-1015	2014-01-02	OVR	Overall Response by Investigator	2014-02-12	WEEK 6	PD	Y
01-701-1015	2014-01-02	OVR	Overall Response by Investigator	2014-03-26	WEEK 12	CR	Y
01-701-1015	2014-01-02	OVR	Overall Response by Investigator	2014-06-18	WEEK 24	SD	Y
01-703-1086	2012-09-02	OVR	Overall Response by Investigator	2012-10-13	WEEK 6	PD	Y
01-703-1086	2012-09-02	OVR	Overall Response by Investigator	2012-11-27	WEEK 12	PD	Y

Here is an alternative example where those records occurring after new anti-cancer therapy are additionally excluded (where NACTDT would be pre-derived as first date of new anti-cancer therapy).

```
adrs <- adrs %>%
  mutate(
    ANL01FL = case_when(
      !is.na(AVAL) & ADT >= RANDDT & ADT < NACTDT ~ "Y",
      TRUE ~ NA_character_
    )
  )
```

Note here that we do not filter out records after first PD at this stage, as that is specifically catered for in the {admiralonco} parameter derivation functions in the below steps, via source_pd arguments.

Derive ANL02FL

However, if you prefer not to rely on source_pd arguments, then the user is free to filter out records after first PD at this stage in a similar way via a ANLzzFL flag, and then you could leave source_pd as null in all downstream parameter derivation function calls. So, for example the user could create ANL02FL flag to subset the post-baseline response data up to and including first reported progressive disease. This would be an alternative and transparent method to the use of source_pd argument approach to create ADRS parameters below. Using admiral function admiral::derive_var_relative_flag() we could create ANL02FL as below.

```
adrs <- adrs %>%
  derive_var_relative_flag(
    by_vars = exprs(USUBJID),
    order = exprs(ADT, AVISITN),
    new_var = ANL02FL,
    condition = AVALC == "PD",
    mode = "first",
    selection = "before",
    inclusive = TRUE
  )
```

Derive Progressive Disease Parameter

Now that we have the input records prepared above with any company-specific requirements, we can start to derive new parameter records. For the parameter derivations, all values except those overwritten by set_values_to argument are kept from the earliest occurring input record fulfilling the required criteria.

The function admiral::derive_param_extreme_event() can be used to find the date of first PD.

```
adrs <- adrs %>%
  derive_param_extreme_event(
    dataset_adsl = adsl,
    dataset_source = adrs,
```

```
    filter_source = PARAMCD == "OVR" & AVALC == "PD" & ANL01FL == "Y",
    order = exprs(ADT, RSSEQ),
    set_values_to = exprs(
      PARAMCD = "PD",
      PARAM = "Disease Progression by Investigator",
      PARCAT1 = "Tumor Response",
      PARCAT2 = "Investigator",
      PARCAT3 = "Recist 1.1",
      ANL01FL = "Y"
    )
  )
```

USUBJID	PARAMCD	PARAM	ADT	AVISIT	AVALC	ANL01FL
01-701-1015	PD	Disease Progression by Investigator	2014-02-12	WEEK 6	PD	Y
01-703-1086	PD	Disease Progression by Investigator	2012-10-13	WEEK 6	PD	Y
01-716-1024	PD	Disease Progression by Investigator	2012-08-24	WEEK 6	PD	Y
01-701-1023	PD	Disease Progression by Investigator				Y
01-703-1096	PD	Disease Progression by Investigator				Y

For progressive disease, response and death parameters shown in steps here and below, in our examples we show these as ADRS parameters, but they could equally be achieved via ADSL dates or ADEVENT parameters. If you prefer to store as an ADSL date, then the function admiral::derive_var_extreme_dt() could be used to find the date of first PD as a variable, rather than as a new parameter record. All the parameter derivation functions that use these dates are flexible to allow sourcing these from any input source using admiral::date_source(). See examples below.

Derive Response Parameter

The next required step is to define the source location for this newly derived PD date.

```
pd <- date_source(
  dataset_name = "adrs",
  date = ADT,
  filter = PARAMCD == "PD" & AVALC == "Y"
)
```

An equivalent example if using ADSL instead could be as follows (where PDDT would be pre-derived as first date of progressive disease).

```
pd <- date_source(
  dataset_name = "adsl",
  date = PDDT
)
```

The function derive_param_response() can then be used to find the date of first response. This differs from the admiral::derive_param_extreme_event() function in that it only looks for events occurring prior to first PD. In the below example, the response condition has been defined as CR or PR.

```
adrs <- adrs %>%
  derive_param_response(
    dataset_adsl = adsl,
    filter_source = PARAMCD == "OVR" & AVALC %in% c("CR", "PR") &
ANL01FL == "Y",
    source_pd = pd,
    source_datasets = list(adrs = adrs),
```

```
    set_values_to = exprs(
      PARAMCD = "RSP",
      PARAM = "Response by Investigator (confirmation not required)",
      PARCAT1 = "Tumor Response",
      PARCAT2 = "Investigator",
      PARCAT3 = "Recist 1.1",
      ANL01FL = "Y"
    )
  )
```

USUBJID	PARAMCD	PARAM	ADT	AVISIT	AVALC	ANL01FL
01-701-1015	RSP	Response by Investigator (confirmation not required)	2014-03-26	WEEK 12	Y	Y
01-701-1023	RSP	Response by Investigator (confirmation not required)			N	Y
01-703-1086	RSP	Response by Investigator (confirmation not required)			N	Y
01-703-1096	RSP	Response by Investigator (confirmation not required)			N	Y
01-707-1037	RSP	Response by Investigator (confirmation not required)			N	Y

Derive Clinical Benefit Parameter

Similarly, we now define the source location for this newly derived first response date.

```
resp <- date_source(
  dataset_name = "adrs",
  date = ADT,
  filter = PARAMCD == "RSP" & AVALC == "Y"
)
```

The function derive_param_clinbenefit() can then be used to derive the clinical benefit parameter, which we define as a patient having had a response or a sustained period of time before first PD. This could also be known as disease control. In this example the “sustained period” has been defined as 42 days after randomization date, using the ref_start_window argument.

```
adrs <- adrs %>%
  derive_param_clinbenefit(
    dataset_adsl = adsl,
    filter_source = PARAMCD == "OVR" & ANL01FL == "Y",
    source_resp = resp,
 source_pd = pd,
 source_datasets = list(adrs = adrs),
 reference_date = RANDDT,
 ref_start_window = 42,
 set_values_to = exprs(
   PARAMCD = "CB",
   PARAM = "Clinical Benefit by Investigator (confirmation for response
not required)",
   PARCAT1 = "Tumor Response",
   PARCAT2 = "Investigator",
   PARCAT3 = "Recist 1.1",
   ANL01FL = "Y"
 )
)
```

USUBJID	RANDDT	PARAMCD	PARAM	ADT	AVISIT	AVALC	ANL01FL
01-701-1015	2014-01-02	CB	Clinical Benefit by Investigator (confirmation for response not required)	2014-03-26	WEEK 12	Y	Y
01-701-1023	2012-08-05	CB	Clinical Benefit by Investigator (confirmation for response not required)			N	Y
01-703-1086	2012-09-02	CB	Clinical Benefit by Investigator (confirmation for response not required)			N	Y
01-703-1096	2013-01-25	CB	Clinical Benefit by Investigator (confirmation for response not required)			N	Y
01-707-1037	2013-12-20	CB	Clinical Benefit by Investigator (confirmation for response not required)			N	Y

Derive Best Overall Response Parameter

The function derive_param_bor() can be used to derive the best overall response (without confirmation required) parameter. Similar to the above function you can optionally decide what period would you consider a SD or NON-CR/NON-PD as being eligible from. In this example, 42 days after randomization date has been used again.

```
adrs <- adrs %>%
  derive_param_bor(
    dataset_adsl = adsl,
    filter_source = PARAMCD == "OVR" & ANL01FL == "Y",
    source_pd = pd,
    source_datasets = list(adrs = adrs),
    reference_date = RANDDT,
    ref_start_window = 42,
    set_values_to = exprs(
      PARAMCD = "BOR",
      PARAM = "Best Overall Response by Investigator (confirmation not
required)",
      PARCAT1 = "Tumor Response",
      PARCAT2 = "Investigator",
      PARCAT3 = "Recist 1.1",
      ANL01FL = "Y"
    )
  )
```

USUBJID	RANDDT	PARAMCD	PARAM	ADT	AVISIT	AVALC	ANL01FL
01-701-1015	2014-01-02	BOR	Best Overall Response by Investigator (confirmation not required)	2014-03-26	WEEK 12	CR	Y
01-701-1023	2012-08-05	BOR	Best Overall Response by Investigator (confirmation not required)			MISSING	Y
01-703-1086	2012-09-02	BOR	Best Overall Response by Investigator (confirmation not required)	2012-10-13	WEEK 6	PD	Y
01-703-1096	2013-01-25	BOR	Best Overall Response by Investigator (confirmation not required)			MISSING	Y
01-707-1037	2013-12-20	BOR	Best Overall Response by Investigator (confirmation not required)			MISSING	Y

Note that the above gives pre-defined AVAL values of: "CR" ~ 1, "PR" ~ 2, "SD" ~ 3, "NON-CR/NON-PD" ~ 4, "PD" ~ 5, "NE" ~ 6, "MISSING" ~ 7.

If you would like to provide your own company-specific ordering here you could do this as follows:

```
aval_resp_new <- function(arg) {
  case_when(
    arg == "CR" ~ 7,
    arg == "PR" ~ 6,
    arg == "SD" ~ 5,
    arg == "NON-CR/NON-PD" ~ 4,
    arg == "PD" ~ 3,
    arg == "NE" ~ 2,
    arg == "MISSING" ~ 1,
    TRUE ~ NA_real_
  )
}
```

Then add the additional argument aval_fun = aval_resp_new to the above derive_param_bor() call. Be aware that this will only impact the AVAL mapping, not the derivation of BOR in any way - as the function derivation relies only on AVALC here.

Derive Best Overall Response of CR/PR Parameter

The function admiral::derive_param_extreme_event() can be used to check if a patient had a response for BOR.

```
adrs <- adrs %>%
  derive_param_extreme_event(
    dataset_adsl = adsl,
    dataset_source = adrs,
    filter_source = PARAMCD == "BOR" & AVALC %in% c("CR", "PR") &
ANL01FL == "Y",
    order = exprs(ADT, RSSEQ),
    set_values_to = exprs(
      PARAMCD = "BCP",
      PARAM = "Best Overall Response of CR/PR by Investigator
(confirmation not required)",
      PARCAT1 = "Tumor Response",
      PARCAT2 = "Investigator",
      PARCAT3 = "Recist 1.1",
      ANL01FL = "Y"
    )
  )
```

USUBJID	PARAMCD	PARAM	ADT	AVISIT	AVALC	ANL01FL
01-701-1015	BCP	Best Overall Response of CR/PR by Investigator (confirmation not required)	2014-03-26	WEEK 12	CR	Y
01-701-1023	BCP	Best Overall Response of CR/PR by Investigator (confirmation not required)				Y
01-703-1086	BCP	Best Overall Response of CR/PR by Investigator (confirmation not required)				Y
01-703-1096	BCP	Best Overall Response of CR/PR by Investigator (confirmation not required)				Y
01-707-1037	BCP	Best Overall Response of CR/PR by Investigator (confirmation not required)				Y

Derive Response Parameters requiring Confirmation

Any of the above response parameters can be repeated for “confirmed” responses only. For these the functions derive_param_confirmed_resp() and derive_param_confirmed_bor() can be used. Some of the other functions from above can then be re-used passing in these confirmed response records. See the examples below of derived parameters requiring confirmation. The assessment and the confirmatory assessment here need to occur at least 28 days apart (without any +1 applied to this calculation of days between visits), using the ref_confirm argument.

```
adrs <- adrs %>%
  derive_param_confirmed_resp(
    dataset_adsl = adsl,
    filter_source = PARAMCD == "OVR" & AVALC %in% c("CR", "PR") &
ANL01FL == "Y",
    source_pd = pd,
    source_datasets = list(adrs = adrs),
    ref_confirm = 28,
    set_values_to = exprs(
      PARAMCD = "CRSP",
      PARAM = "Confirmed Response by Investigator",
      PARCAT1 = "Tumor Response",
      PARCAT2 = "Investigator",
      PARCAT3 = "Recist 1.1",
      ANL01FL = "Y"
    )
  )

confirmed_resp <- date_source(
  dataset_name = "adrs",
  date = ADT,
  filter = PARAMCD == "CRSP" & AVALC == "Y"
)

adrs <- adrs %>%
  derive_param_clinbenefit(
    dataset_adsl = adsl,
    filter_source = PARAMCD == "OVR" & ANL01FL == "Y",
    source_resp = confirmed_resp,
    source_pd = pd,
    source_datasets = list(adrs = adrs),
    reference_date = RANDDT,
    ref_start_window = 42,
    set_values_to = exprs(
      PARAMCD = "CCB",
      PARAM = "Confirmed Clinical Benefit by Investigator",
      PARCAT1 = "Tumor Response",
      PARCAT2 = "Investigator",
      PARCAT3 = "Recist 1.1",
      ANL01FL = "Y"
    )
  ) %>%
  derive_param_confirmed_bor(
    dataset_adsl = adsl,
    filter_source = PARAMCD == "OVR" & ANL01FL == "Y",
    source_pd = pd,
    source_datasets = list(adrs = adrs),
    reference_date = RANDDT,
```

```
    ref_start_window = 42,
    ref_confirm = 28,
    set_values_to = exprs(
      PARAMCD = "CBOR",
      PARAM = "Best Confirmed Overall Response by Investigator",
      PARCAT1 = "Tumor Response",
      PARCAT2 = "Investigator",
      PARCAT3 = "Recist 1.1",
      ANL01FL = "Y"
    )
  ) %>%
  derive_param_extreme_event(
    dataset_adsl = adsl,
    dataset_source = adrs,
    filter_source = PARAMCD == "CBOR" & AVALC %in% c("CR", "PR") &
ANL01FL == "Y",
    order = exprs(ADT, RSSEQ),
    set_values_to = exprs(
      PARAMCD = "CBCP",
      PARAM = "Best Confirmed Overall Response of CR/PR by
Investigator",
      PARCAT1 = "Tumor Response",
      PARCAT2 = "Investigator",
      PARCAT3 = "Recist 1.1",
      ANL01FL = "Y"
    )
  )
```

USUBJID	RANDDT	PARAMCD	PARAM	ADT	AVISIT	AVALC	ANL01FL
01-701-1015	2014-01-02	CRSP	Confirmed Response by Investigator			N	Y
01-701-1023	2012-08-05	CRSP	Confirmed Response by Investigator			N	Y
01-703-1086	2012-09-02	CRSP	Confirmed Response by Investigator			N	Y
01-703-1096	2013-01-25	CRSP	Confirmed Response by Investigator			N	Y
01-707-1037	2013-12-20	CRSP	Confirmed Response by Investigator			N	Y

Derive Parameters using Independent Review Facility (IRF)/Blinded Independent Central Review (BICR) responses

All of the above steps can be repeated for different sets of records, such as now using assessments from the IRF/BICR instead of investigator. For this you would just need to replace the first steps with selecting the required records, and then feed these as input to the downstream parameter functions.

Remember that a new progressive disease and response source object would be required for passing to source_pd and source_resp, respectively.

```
adrsirf <- rs %>%
  filter(RSEVAL == "INDEPENDENT ASSESSOR" & RSEVALID == "RADIOLOGIST 1"
& RSTESTCD == "OVRLRESP") %>%
  mutate(
    PARAMCD = "OVRR1",
    PARAM = "Overall Response by Radiologist 1",
    PARCAT1 = "Tumor Response",
    PARCAT2 = "Radiologist",
    PARCAT3 = "Recist 1.1"
  )
```

USUBJID	RSTESTCD	RSEVAL	VISIT	PARAMCD	PARAM	PARCAT1	PARCAT2	PARCAT3
01-701-1015	OVRLRESP	INDEPENDENT ASSESSOR	WEEK 6	OVRR1	Overall Response by Radiologist 1	Tumor Response	Radiologist	Recist 1.1
01-701-1015	OVRLRESP	INDEPENDENT ASSESSOR	WEEK 12	OVRR1	Overall Response by Radiologist 1	Tumor Response	Radiologist	Recist 1.1
01-701-1015	OVRLRESP	INDEPENDENT ASSESSOR	WEEK 24	OVRR1	Overall Response by Radiologist 1	Tumor Response	Radiologist	Recist 1.1
01-703-1086	OVRLRESP	INDEPENDENT ASSESSOR	WEEK 6	OVRR1	Overall Response by Radiologist 1	Tumor Response	Radiologist	Recist 1.1
01-703-1086	OVRLRESP	INDEPENDENT ASSESSOR	WEEK 12	OVRR1	Overall Response by Radiologist 1	Tumor Response	Radiologist	Recist 1.1

Then in all the calls to the parameter derivation functions you would replace the PARAMCD == "OVR" source with PARAMCD == "OVRR1".

Derive Death Parameter

The function admiral::derive_param_extreme_event() can be used to create a new death parameter using death date from ADSL. We need to restrict the columns from ADSL as we will merge all required variables later across all our ADRS records.

```
adsldth <- adsl %>%
  select(STUDYID, USUBJID, DTHDT, !!!adsl_vars)

adrs <- adrs %>%
  derive_param_extreme_event(
    dataset_adsl = adsldth,
    dataset_source = adsldth,
    filter_source = !is.na(DTHDT),
    set_values_to = exprs(
      PARAMCD = "DEATH",
      PARAM = "Death",
      PARCAT1 = "Reference Event",
      ANL01FL = "Y",
      ADT = DTHDT
    )
  ) %>%
  select(-DTHDT)
```

USUBJID	PARAMCD	PARAM	ADT	AVISIT	AVALC	ANL01FL
01-701-1015	DEATH	Death				Y
01-701-1023	DEATH	Death				Y
01-703-1086	DEATH	Death				Y
01-703-1096	DEATH	Death				Y
01-707-1037	DEATH	Death				Y

Derive Last Disease Assessment Parameters

The function admiral::derive_param_extreme_event() can be used to create a parameter for last disease assessment. We need to set new_var to a dummy variable here, or otherwise the original AVALC value from the record would be overwritten with "Y".

```
adrs <- adrs %>%
  derive_param_extreme_event(
    dataset_adsl = adsl,
    dataset_source = adrs,
    filter_source = PARAMCD == "OVR" & ANL01FL == "Y",
    order = exprs(ADT, RSSEQ),
    mode = "last",
    new_var = dummy,
    set_values_to = exprs(
      PARAMCD = "LSTA",
      PARAM = "Last Disease Assessment by Investigator",
      PARCAT1 = "Tumor Response",
      PARCAT2 = "Investigator",
      PARCAT3 = "Recist 1.1",
      ANL01FL = "Y"
    )
  ) %>%
  select(-dummy)
```

USUBJID	PARAMCD	PARAM	ADT	AVISIT	AVALC	ANL01FL
01-701-1015	LSTA	Last Disease Assessment by Investigator	2014-06-18	WEEK 24	SD	Y
01-703-1086	LSTA	Last Disease Assessment by Investigator	2012-11-27	WEEK 12	PD	Y
01-716-1024	LSTA	Last Disease Assessment by Investigator	2012-12-30	WEEK 24	PD	Y
01-701-1023	LSTA	Last Disease Assessment by Investigator				Y
01-703-1096	LSTA	Last Disease Assessment by Investigator				Y

Derive Measurable Disease at Baseline Parameter

The function admiral::derive_param_exist_flag() can be used to check whether a patient has measurable disease at baseline, according to a company-specific condition. In this example we check TU for target lesions during the baseline visit. We need to restrict the columns from ADSL as we will merge all required variables later across all our ADRS records.

```
adslmdis <- adsl %>%
  select(STUDYID, USUBJID, !!!adsl_vars)

adrs <- adrs %>%
  derive_param_exist_flag(
    dataset_adsl = adslmdis,
    dataset_add = tu,
    condition = TUEVAL == "INVESTIGATOR" & TUSTRESC == "TARGET" & VISIT
== "BASELINE",
    false_value = "N",
    missing_value = "N",
    set_values_to = exprs(
      PARAMCD = "MDIS",
      PARAM = "Measurable Disease at Baseline by Investigator",
      PARCAT2 = "Investigator",
      PARCAT3 = "Recist 1.1",
      ANL01FL = "Y"
    )
  )
```

USUBJID	PARAMCD	PARAM	ADT	AVISIT	AVALC	ANL01FL
01-701-1015	MDIS	Measurable Disease at Baseline by Investigator			Y	Y
01-701-1023	MDIS	Measurable Disease at Baseline by Investigator			Y	Y
01-703-1086	MDIS	Measurable Disease at Baseline by Investigator			Y	Y
01-703-1096	MDIS	Measurable Disease at Baseline by Investigator			Y	Y
01-707-1037	MDIS	Measurable Disease at Baseline by Investigator			Y	Y

Derive AVAL for New Parameters

For cases where AVALC has been derived for new parameters above as "Y" or "N", we need to set AVAL to numeric versions such as 1/0.

```
adrs <- adrs %>%
  mutate(
    AVAL = case_when(
      AVALC == "Y" ~ 1,
      AVALC == "N" ~ 0,
      TRUE ~ AVAL
    )
  )
```

USUBJID	PARAMCD	AVALC	AVAL
01-701-1015	RSP	Y	1
01-701-1015	CB	Y	1
01-701-1015	CRSP	N	0
01-701-1015	CCB	Y	1
01-701-1015	MDIS	Y	1

Assign ASEQ

The function admiral::derive_var_obs_number() can be used to derive ASEQ. An example call is:

```
adrs <- adrs %>%
  derive_var_obs_number(
    by_vars = exprs(STUDYID, USUBJID),
    order = exprs(PARAMCD, ADT, VISITNUM, RSSEQ),
    check_type = "error"
  )
```

USUBJID	VISITNUM	PARAMCD	ADT	AVISIT	ASEQ
01-701-1015	9	BCP	2014-03-26	WEEK 12	1
01-701-1015	9	BOR	2014-03-26	WEEK 12	2
01-701-1015	9	CB	2014-03-26	WEEK 12	3
01-701-1015		CBCP			4
01-701-1015	9	CBOR	2014-03-26	WEEK 12	5

Add ADSL variables

If needed, the other ADSL variables can now be added. List of ADSL variables already merged held in vector adsl_vars.

```
adrs <- adrs %>%
  derive_vars_merged(
    dataset_add = select(adsl, !!!negate_vars(adsl_vars)),
    by_vars = exprs(STUDYID, USUBJID)
  )
```

USUBJID	RFSTDTC	RFENDTC	DTHDTC	DTHFL	AGE	AGEU
01-701-1015	2014-01-02	2014-07-02			63	YEARS
01-701-1015	2014-01-02	2014-07-02			63	YEARS
01-701-1015	2014-01-02	2014-07-02			63	YEARS
01-701-1015	2014-01-02	2014-07-02			63	YEARS
01-701-1015	2014-01-02	2014-07-02			63	YEARS

CREATING ADTTE

Source: https://github.com/pharmaverse/admiralonco/blob/main/vignettes/adtte.Rmd

Example Script: https://github.com/pharmaverse/admiralonco/blob/main/inst/templates/ad_adtte.R

Similar to how ADRS is developed above, the detailed guide for creating ADTTE is available with common oncology endpoint parameters. The main part in programming a time-to-event dataset is the definition of the events and censoring times. admiral/{admiralonco} supports single events like death (Overall Survival) or composite events like disease progression or death (Progression Free Survival). More than one source dataset can be used for the definition of the event and censoring times.

The majority of the functions used here exist from admiral, except for the tte_sources helper object, provided as an example from {admiralonco}. In practice, each company would create their own version of this, as likely the exact specifications such as filtering condition or description metadata will vary.

A step-by-step guide to create admiralonco inspired ADTTE is available on pharmaverse at the link https://pharmaverse.github.io/admiralonco/main/articles/adtte.html

CREATING ADTR

Source: https://github.com/pharmaverse/admiralonco/blob/main/vignettes/adtr.Rmd

Example Script: https://github.com/pharmaverse/admiralonco/blob/main/inst/templates/ad_adtr.R

Similarly, oncology specific ADTR guide is available with common oncology parameters based on RECIST 1.1. The main part in programming a tumor results dataset is the calculation of the sum of diameters of all target lesions (lymph nodes & non-lymph nodes), the calculation of nadir, change & percentage change from baseline, and the analysis flags that could be required for reporting. The tumor results data could be set up for investigator and/or Independent review facility (IRF)/Blinded Independent Central Review (BICR) data. The below sample code would need to be updated (for example, update the Evaluator TR.TREVAL, TU.TUEVAL, and the applicable parameter details PARAM, PARAMCD, PARCATy) in order to create the acquired data for Independent review facility (IRF)/Blinded Independent Central Review (BICR).

The source dataset used will depend on each company, this could be solely the TR domain, or you may merge TU with TR (SDTM inputs) to get additional data variables or to ensure that you are processing the same lesions as collected.

Individual lesion diameters for each target lesion are required to calculate the sum of diameters for all target lesions, this data could be taken directly from TR or additional parameters could be created in ADTR (or similar) depending on the additional processing required (e.g., imputation of dates, re-labeling of visits) and your company specifications. The majority of the functions used here exist from admiral.

A step-by-step guide to create admiralonco inspired ADTR is available on pharmaverse at the link https://pharmaverse.github.io/admiralonco/main/articles/adtr.html

CHALLENGES WHILE DEVELOPING ADMIRALONCO

No journey from concept to reality is complete without an understanding of the challenges and obstacles encountered. Before concluding this paper, we would like to share with you some of the challenges we have faced in the development and implementation of the admiralonco package.

- First of all, we have coded in SAS for years and if you ask an experienced statistical programmer to switch to a new coding language, it is often not easy, and one has to get out of the so-called comfort zone. That is why developing this package in R was a real challenge. It was indeed a paradigm shift.
- Oncology is a broad therapeutic area with many oncology-specific conventions, standards, and requirements. It is difficult to cover all the requirements in one package. Therefore, we have taken a tailored approach by starting with RECIST 1.1 (which is the most common evaluation criterion in oncology) and expanding our horizon as each new CRAN is released.
- Also, due to the unique nature of oncology, we often found ourselves in a situation where there was not enough data to test all possible scenarios that were required or mentioned in the standard guidelines, despite the fact that the package was being tested on different data by different companies. In these cases, we had to develop dummy data to encompass some unusual scenarios to ensure that the package could handle what was required by the guidelines.
- All cross-company initiatives are great learnings and require a very high level of collaboration. In developing this package, we encountered a similar situation and found ourselves at a crossroads where the conventions were different from organization to organization and each company had different work standards. So, it took a lot of discussion to come to a consensus and make sure that the final package took into account the different conventions while still meeting the standards.
- As mentioned in the development process, to make the creation of this package highly effective and efficient, we reuse Admiral functions whenever possible and available. We also have dependencies with other established packages. Therefore, it is essential to monitor updates and decommissioning of dependent functions.
- Finally, as this package is developed in the Agile framework, it is important to maintain the correct versions when using on an ongoing project for the same reason. Since we have CRAN releases every quarter, depending on feedback or expansion plans, improvisations can be introduced. Therefore, when using this package, please make sure to use the latest version available.

CONCLUSION

This paper is an attempt to raise awareness of the availability of the admiralonco package and the benefits it brings. If you are interested, join our slack channel and user community for discussions and updates. The goal of this industry-wide community effort is also for the package to improve with use and feedback. Through this, we would also like to encourage you to test this package on your studies and projects. Feel free to give us feedback, there might be derivations that we probably have not considered yet or there might be more flexibility or customization that you think would be good to add to increase usability. You can give us feedback, any experiences with bugs, or suggestions for improvement directly on Github.

Lastly, as this is envisioned as an industry wide collaboration, if you would like to be more than a consumer this product and would like to contribute by join hands with us, please contact us.

- Slack (https://app.slack.com/client/T028PB489D3/C02M8KN8269) - for informal discussions, Q&A and building our user community. If you don't have access, use this link (https://join.slack.com/t/pharmaverse/shared_invite/zt-yv5atkr4-Np2ytJ6W_QKz_4Olo7Jo9A) to join the pharmaverse Slack workspace
- GitHub Issues (https://github.com/pharmaverse/admiralonco/issues) - for direct feedback, enhancement requests or raising bugs

REFERENCES

- Oncology Extension Package for ADaM in R Asset Library • admiralonco (pharmaverse.github.io) (https://pharmaverse.github.io/admiralonco)
- ADaM in R Asset Library • admiral (pharmaverse.github.io) (https://pharmaverse.github.io/admiral)
- CRAN - Package admiralonco (rstudio.com) (https://cran.rstudio.com/web/packages/admiralonco)
- GitHub - cran/admiralonco: This is a read-only mirror of the CRAN R package repository. admiralonco — Oncology Extension Package for ADaM in 'R' Asset Library (https://github.com/cran/admiralonco)
- ADMIRAL (ADaM in R Asset Library) – Can we Collaborate to Lessen the Burden of ADaM? (https://phuse.s3.eu-central-1.amazonaws.com/Archive/2021/Connect/EU/Virtual/VID_HoW_ADMIRAL.mp4)

ACKNOWLEDGMENTS

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Neharika Sharma
GlaxoSmithKline
1250 S. Collegeville Road,
Collegeville, Pennsylvania, US, 19426-0989
Email: neharika.x.sharma@gsk.com; neharikaa.sharma@gmail.com