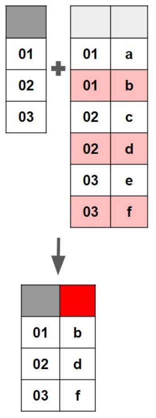
admiral :: CHEAT SHEET

What you need to know

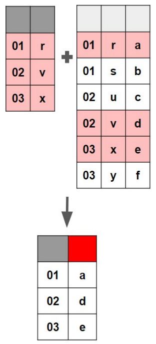
{admiral} is an open-source, modularized toolbox that enables the development of ADaM datasets in R. {admiral} code is comprised of interchangeable blocks, i.e. function calls, that sequentially derive new variables or parameters to help construct an ADaM dataset.

Generic Variable-Adding Functions



derive_vars_merged(dataset, dataset_add, new_vars, filter_add, order, mode…)
Add new variable(s) to the input dataset based on variables from another dataset. Merged observations can be selected by a condition and/or selecting the first/last observation for each by group.

```
derive_vars_merged(
  dataset = adsl,
  dataset_add = vs,
  by_vars = exprs(STUDYID, USUBJID),
  order = exprs(convert_dtc_to_dtm(VSDTC)),
  mode = "last",
  new_vars = exprs(LASTWGT = VSSTRESN),
  filter_add = VSTESTCD == "WEIGHT"
)
```

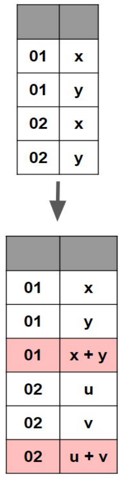


derive_vars_joined(dataset, dataset_add, new_vars, join_type, filter_add, order, mode…)
Add variables from an additional dataset to the input dataset. The selection of the observations from the additional dataset can depend on variables from both datasets.

```
derive_vars_joined(
  dataset = adae, dataset_add = period_ref,
  by_vars = exprs(STUDYID, USUBJID),
  join_vars = exprs(APERSDT, APEREDT),
  join_type = "all",
  filter_join = APERSDT <= ASTDT & [...]
)
```

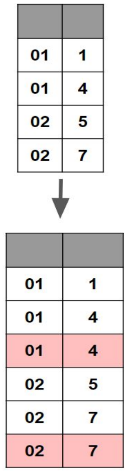
Notable others
derive_vars_extreme_event() derive_vars_merged_lookup()
derive_vars_transposed(), derive_var_merged_ef_msrc()
derive_vars_computed(), derive_var_merged_summary()

Generic Parameter-Adding Functions



derive_param_computed(dataset, dataset_add = NULL, by_vars, parameters, set_values_to, …)
Add a parameter computed from the analysis value of other parameters.

```
derive_param_computed(
 dataset = advs,
 by_vars = exprs(USUBJID, VISIT),
 parameters = c("SYSBP", "DIABP"),
 set_values_to = exprs(
   AVAL = (AVAL.SYSBP+2*AVAL.DIABP)/3,
   PARAMCD = "MAP",
   PARAM = "Mean Arterial Pressure",
   AVALU = "mmHg"
 )
)
```



derive_extreme_records(dataset, dataset_add, dataset_ref, by_vars, order, mode, keep_source_vars, set_values_to, …)
Add the first or last observation for each by group as new observations. The new observations can be selected from the input dataset or an additional dataset.

```
derive_extreme_records(
 dataset = adlb,  dataset_add = adlb,
 by_vars = exprs(USUBJID),
 order = exprs(AVAL, AVISITN),
 mode = "first", filter_add = !is.na(AVAL),
 keep_source_vars = exprs(AVAL),
 set_values_to = exprs(DTYPE = MIN"))
```

Notable others:
derive_expected_records(), derive_locf_records()
derive_extreme_event(), derive_param_exposure(),
derive_summary_records()

Note: These functions are just some examples of the many generic variable/parameter-adding functions in {admiral}. Check the reference page for all of them!

Functions Treating Days/Dates/Datetimes

derive_vars_(dt/dtm)(dataset, new_vars_prefix, …)
Derive or impute a date/datetime from a date character Vector.

derive_vars_dt(admh, new_vars_prefix = "AST", dtc = MHSTDTC)

derive_vars_dy(dataset, reference_date, source_vars)
Adds relative day variables (--DY).

```
derive_vars_dy(
  dataset = adsl, reference_date = TRTSDTM,
  source_vars = exprs(TRTSDTM, ASTDTM, AENDT)
)
```

derive_vars_dtm_to_(dt/tm)(dataset, source_vars,…)
Derive date/time variables from datetime variables.

```
derive_vars_dtm_to_tm(
   dataset = adcm, source_var = exprs(TRTSDTM)
)
```

derive_vars_duration(dataset, new_var, new_var_unit, start_date, end_date).
Derive duration between two dates.

```
derive_vars_duration(
  dataset = adsl, new_var = AAGE, new_var_unit = AAGEU,
  start_date = BRTHDT, end_date = RANDDT,
  out_unit =  years"
)
```

Computation Functions for Vectors

These functions do what their names suggest and can be used inside dplyr:: mutate() or other {admiral} functions.

compute_age_years()	**convert_dtc_to_dt()**
compute_dtf()	**convert_dtc_to_dtm()**
compute_duration()	**impute_dtc_dt()**
compute_tmf()	**impute_dtc_dtm()**
convert_date_to_dtm()	

Links: Github Repo - Documentation - Join the Pharmaverse Slack

Special Variable-Adding Functions

derive_var_age_years(dataset, age_var, age_unit, new_var)
Derive age in years.

derive_vars_period(dataset, dataset_ref, new_vars)
Add subperiod, period, or phase variables.

derive_var_anrind(dataset, use_a1h1lo)
Derive analysis reference range indicator (ANRIND)

derive_var_atoxgr(dataset, lotox_description_var, hitox_description_var)
Derive character lab grade based on high and low severity/toxicity grade(s).

derive_var_base/chg/pchg(dataset, …)
Derive baseline/change/percent change variables.

derive_var_ontrtfl(dataset, start_date, ref_start_date, ref_end_date, ref_end_window …)
Derive on-treatment flag (ONTRTFL) with a single assessment date (e.g ADT) or event start and end dates (e.g. ASTDT/AENDT).

derive_var_trtemfl(dataset, new_var, start_date, end_date, trt_start_date, trt_end_date, end_window, …)
Derive treatment emergent analysis flag (TRTEMFL).

derive_vars_query(dataset, dataset_queries)
Derive query variables.

derive_vars_atc(dataset, dataset_facm, by_vars, value_var)
Derive ATC class variables from FACM to ADCM..

Special Parameter-Adding Functions

***derive_param_bmi**(dataset, by_vars, set_values_to, …)
Derive BMI parameter.

***derive_param_bsa**(dataset, by_vars, set_values_to, …)
Derive body surface area parameter (multiple methods).

***derive_param_map**(dataset, by_vars, set_values_to, …)
Derive mean arterial pressure parameter.

derive_param_doseint(dataset, by_vars, set_values_to, …)
Derive dose intensity parameter.

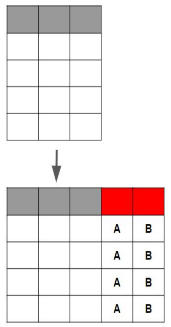
derive_param_tte(dataset, dataset_adsl, source_datasets, by_vars, start_date, event_conditions, censor_conditions, …)
Derive time-to-event parameter.

* wrapper of derive_param_computed().

Note: These functions are just some examples of the many special variable/parameter-adding functions in {admiral}. Check the reference page for all of them!

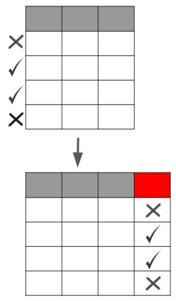
Higher Order Functions

Meta-functions that take {admiral} functions as input and facilitate their execution.



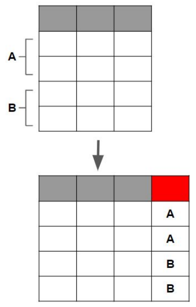
call_derivation(dataset, derivation, variable_params, …)
Call a single derivation multiple times with some parameters/arguments fixed across calls and others varying.

call_derivation(
  dataset = adae,
  derivation = derive_vars_dt,
  variable_params = list(
    params([...]),
    params([...])
))



restrict_derivation(dataset, derivation, args, filter)
Execute a derivation on a subset of the input dataset.

restrict_derivation(
  dataset = adlb,
  derivation = derive_vars_merged,
  args = params([...]),
  filter = AVISITN > 0
)



slice_derivation(dataset, derivation, args, …)
The input dataset is split into slices (subsets) and for each slice the derivation is called separately. Some or all arguments of the derivation may vary depending on the slice.

slice_derivation(
  dataset = advs,
  derivation = derive_vars_dtm,
  args = params([...]),
  derivation_slice(filter = [...], args = [...]),
  derivation_slice(filter = [...], args = [...]),
)

Links: Github Repo - Documentation - Join the Pharmaverse Slack

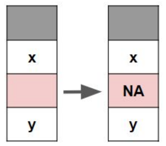
Templates

Example scripts to be used as a starting point for ADaM creation.

list_all_templates(package)
List all available ADaM templates in {admiral} (or another package).

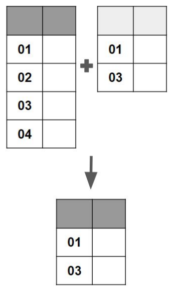
use_ad_template(adam_name, package, overwrite, open)
Open an ADaM template script.   *use_ad_template("adsl")*

Utilities



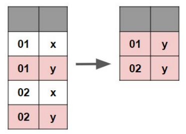
convert_blanks_to_na()
Turn SAS blank strings into R NAs.

convert_blanks_to_na(c("a", "", "b"))



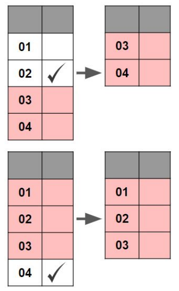
filter_exist(dataset, dataset_add, by_vars, filter_add)
Returns all records in the input dataset belonging to by groups present in a (possibly filtered) source dataset.

filter_exist(
  dataset = adsl, dataset_add = adae,
  by_vars = exprs(USUBJID),
  filter_add = AEDECOD == "FATIGUE")
)



filter_extreme(dataset, by_vars, order, mode, check_type = "warning")
Filters the first/last record in by group.

filter_extreme(by_vars = exprs(USUBJID),
  order = exprs(EXSEQ), mode = "first")



filter_relative(dataset, by_vars, order, condition, mode, selection, inclusive…)
Filters the observations before or after the observation where a specified condition is fulfilled for each by group.

filter_relative(
  response,
  by_vars = exprs(USUBJID),
  order = exprs(AVISITN),
  condition = AVALC == "PD",
  mode = "first", selection = "before",
  inclusive = TRUE
)