

# 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

01	01	a
02	01	b
03	02	c
	02	d
	03	e
	03	f

`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.

01	r	a
02	v	b
03	x	c
	02	d
	03	e
	03	f

`derive_vars_merged(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_merged(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_merged(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.

## 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!

Links: [Github Repo](#) - [Documentation](#) - [Join the Pharmaverse Slack](#)

## Functions Treating Days/Dates/Datetimes

`derive_vars_dt(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 = MHSTDTTC)`

`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/dm(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_date_to_dtm()`  
`compute_dtf()` `transform_range()`  
`compute_duration()` `convert_dtc_to_dt()`  
`compute_tmf()` `convert_dtc_to_dtm()`  
`compute_scale()`

## 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\_dir**(dataset, lotox\_description\_var, hitox\_description\_var)  
Derive character lab grade based on severity or toxicity criteria.

**derive\_var\_(base/chg/pchg)**(dataset, ...)  
Derive baseline/change/percent change variables.

**derive\_vars\_crit\_flag**(dataset, condition, description, ...)  
Derive criterion flag variables (CRITy, CRITyFL(N)).

**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\_var\_trtemfl**(dataset, new\_var, start\_date, end\_date, trt\_start\_date, trt\_end\_date, end\_window, ...)  
Derive treatment emergent analysis flag (TRTEMFL).

**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\_query**(dataset, dataset\_queries)  
Derive query variables.

**derive\_vars\_atc**(dataset, dataset\_facm, by\_vars, id\_vars, value\_var)  
Derive ATC class variables from FACM to ADCM.

**derive\_vars\_atc**(dataset, dataset\_facm, by\_vars, id\_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.


			A	B
			A	B
			A	B
			A	B

**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([...])
  )
)
```

x			
✓			
✓			
✓			
x			

				x
				✓
				✓
				x

**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
)
```

A			
B			

				A
				A
				B
				B

**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 = [...])
)
```

## 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

x	
y	

**convert\_blanks\_to\_na**()  
Turn SAS blank strings into R NAs.

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

01	
02	
03	
04	

**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")`

01	
03	

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

01	x
01	y
02	x
02	y

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

01	
02	✓
03	
04	

**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.

01	
02	
03	
04	✓

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

