

# 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.

**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"  
)

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

**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

01	x
01	y
02	x
02	y

**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"  
)

01	x
01	y
01	x + y
02	u
02	v
02	u + v

**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"))

01	1
01	4
01	4
02	5
02	7
02	7

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 = MHSTDTCT)

**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.

<b>compute_age_years()</b>	<b>convert_dtc_to_dt()</b>
<b>compute_dtf()</b>	<b>convert_dtc_to_dtm()</b>
<b>compute_duration()</b>	<b>impute_dtc_dt()</b>
<b>compute_tmf()</b>	<b>impute_dtc_dtm()</b>
<b>convert_date_to_dtm()</b>	

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

**derive\_var\_onttrtfl**(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.**


			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.

01	
03	

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

01	x
01	y
02	x
02	y

**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")
```

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
)
```

