More Science, Less Bookkeeping: Our Path to Hyper-Efficient ADaM Program Generation via AI, Process Improvements, and Good Old-Fashioned Programming

SM04
PHUSE EU Connect 2025

Matthew Phelps
Novo Nordisk A/S
&
Nicolai Skov Johnsen

Views and opinions expressed are those of the speakers and not necessarily Novo Nordisk

The **mighty** team

Nicolai Skov Jensen

Michael Hedegaard Thomsen

Matthew Phelps

Aksel Thomsen

What's the problem?

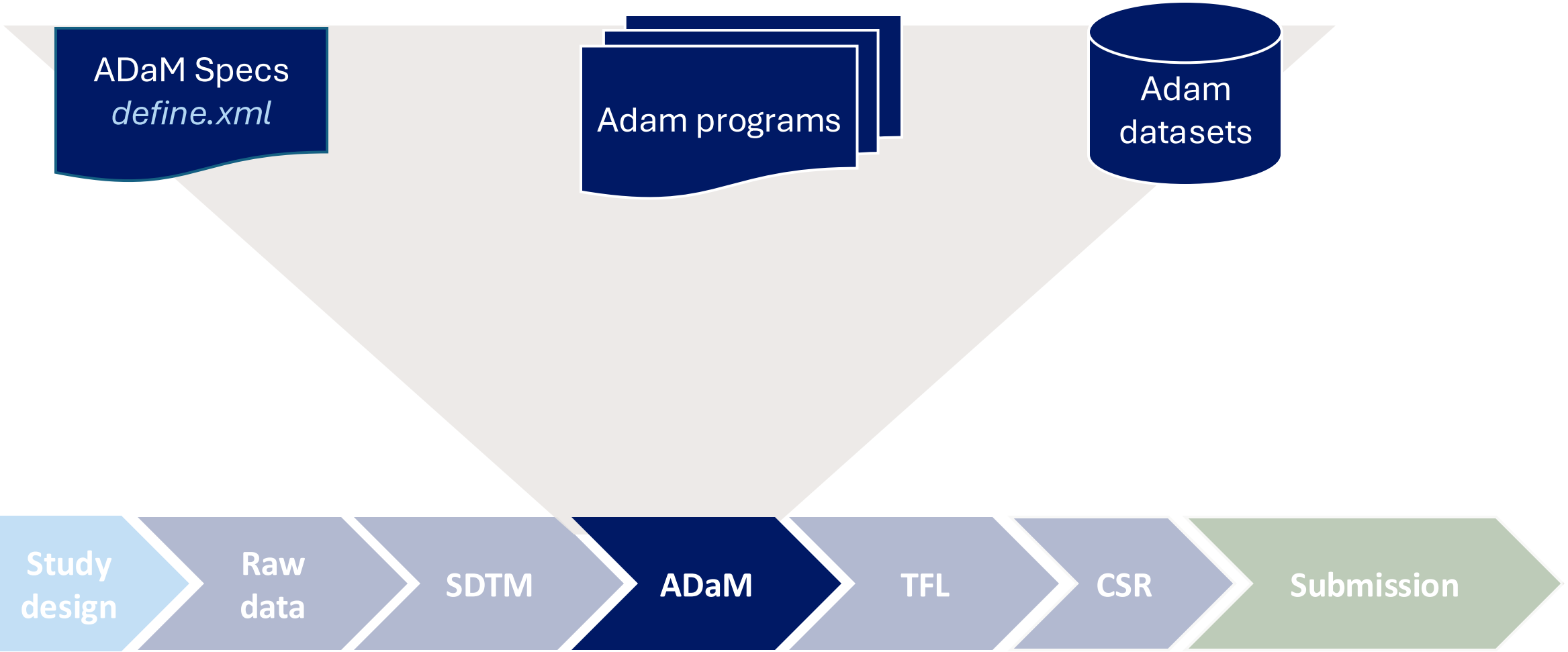
What's the problem?

Study design → Raw data → CSR → Submission

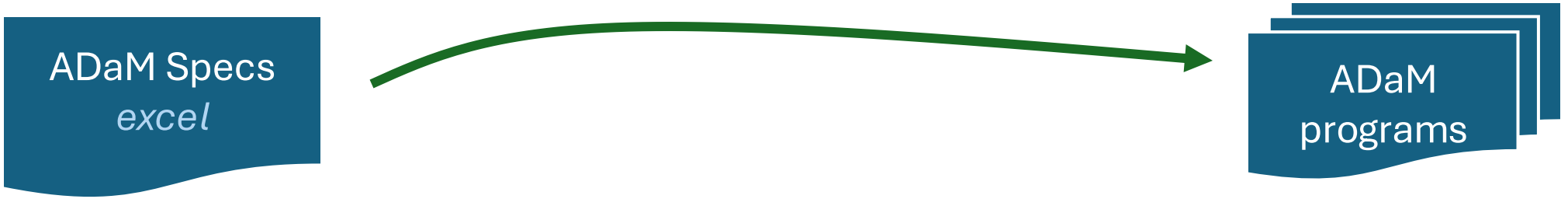
What's the problem?


Study design
Raw data
SDTM
ADaM
TFL
CSR
Submission

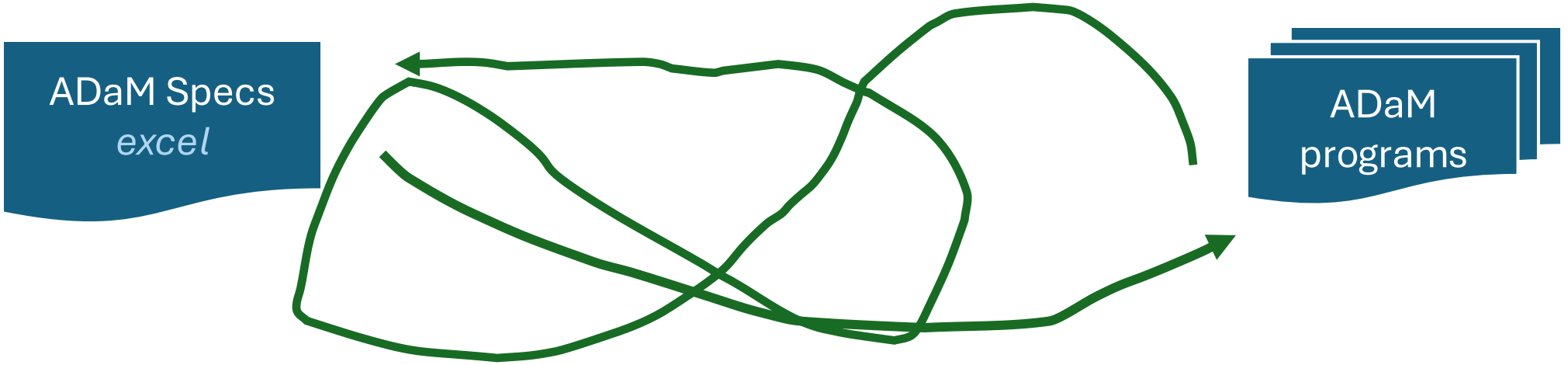
~1.7 → ~1.0


ADaM Specs
define.xml
Adam programs
Adam datasets
Study design
Raw data
SDTM
ADaM
TFL
CSR
Submission

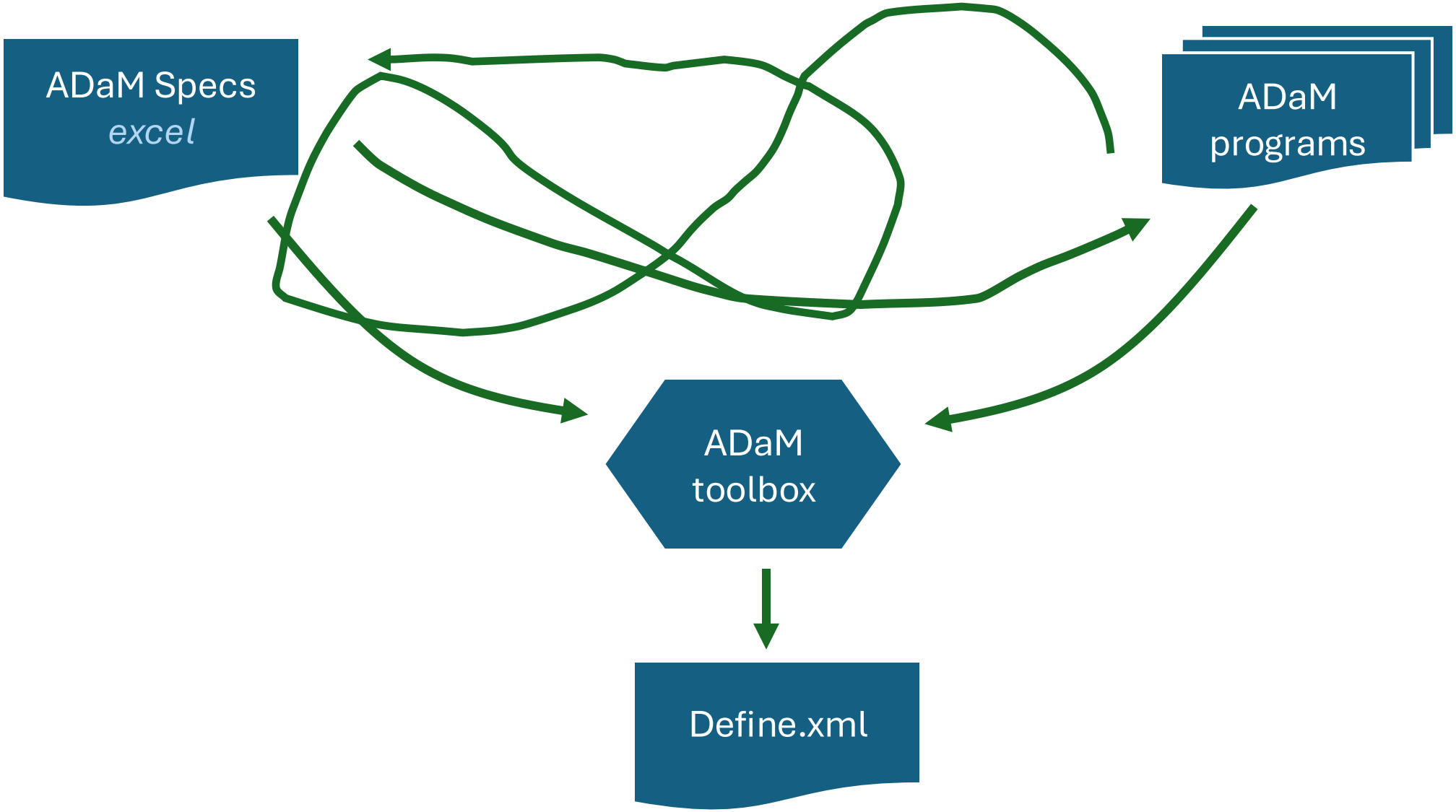
Current state of ADaM programming



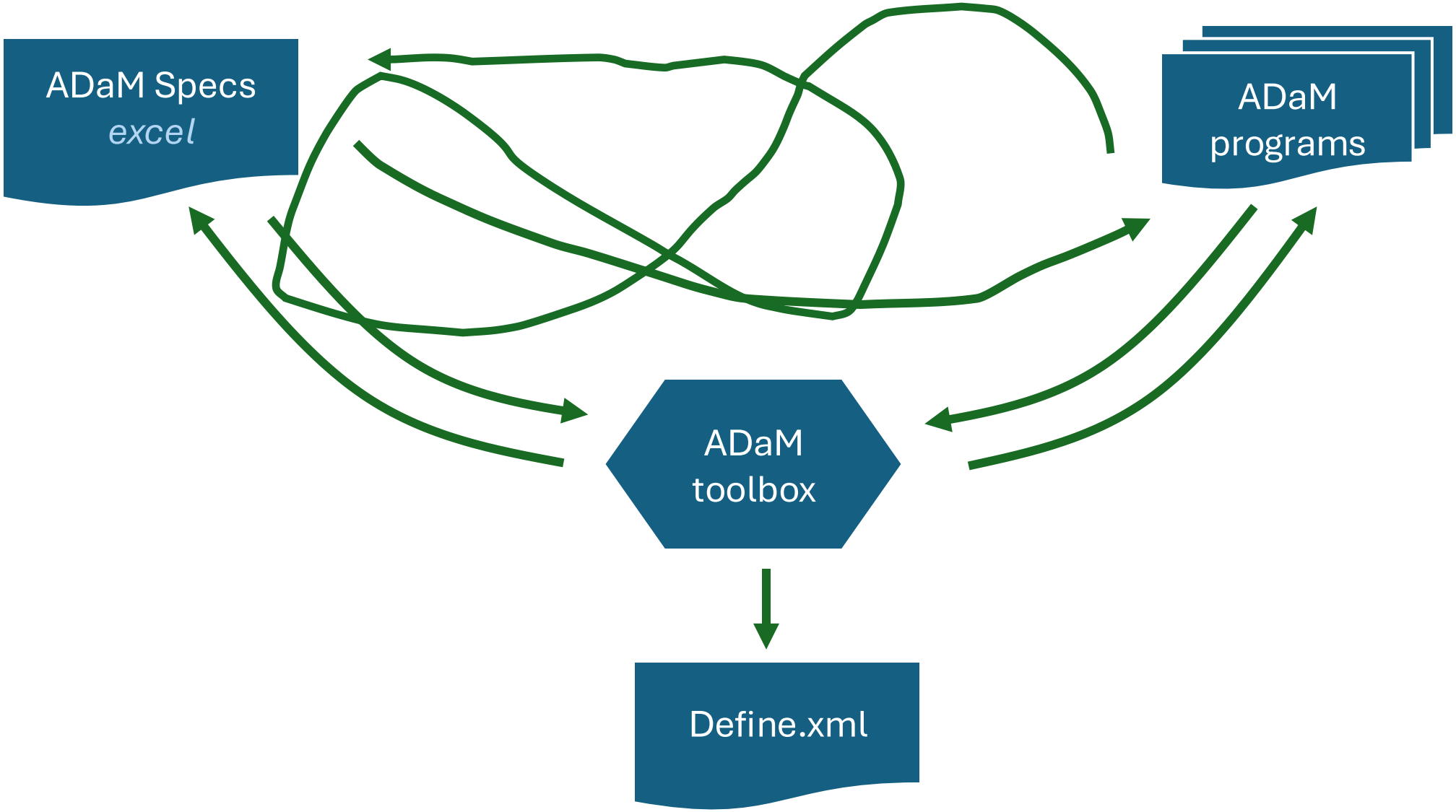
Current state of ADaM programming



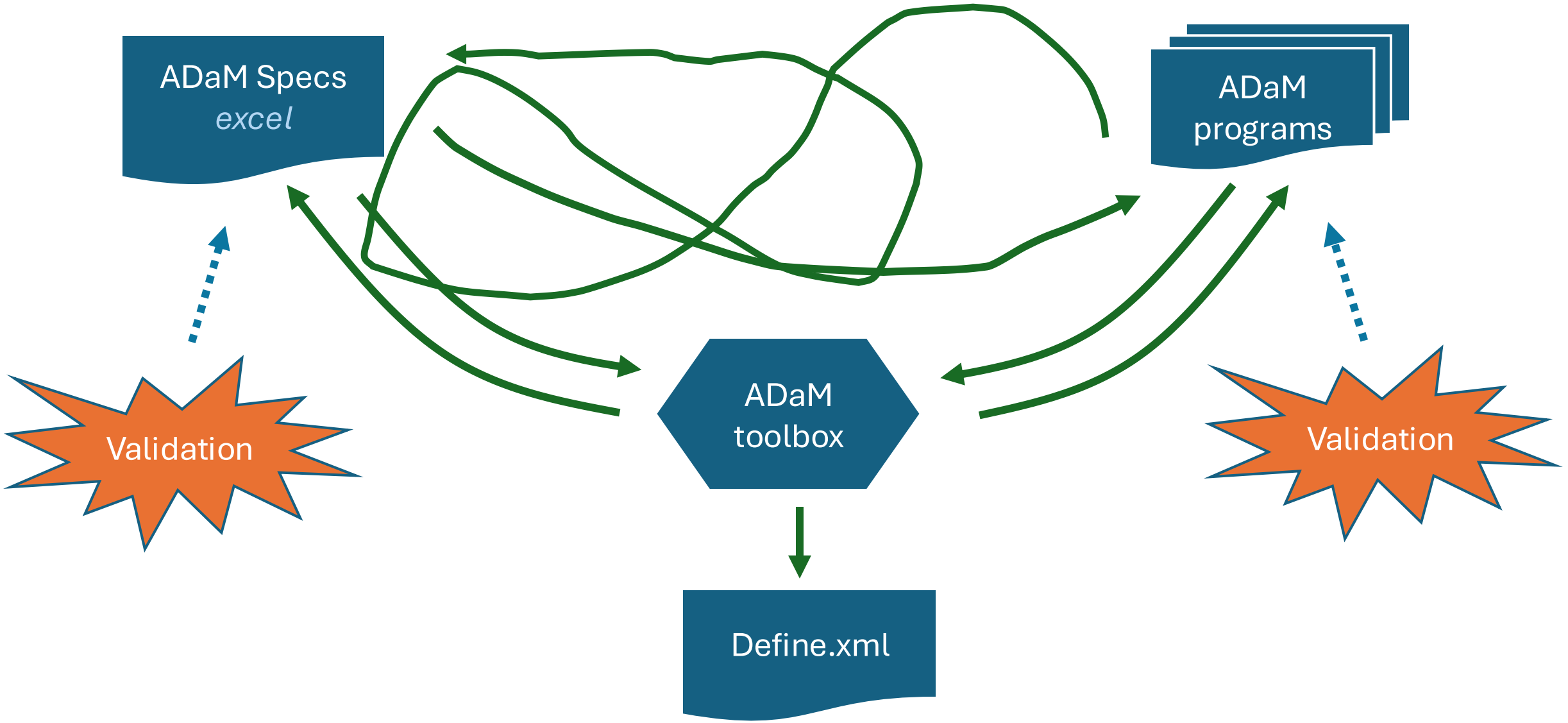
Current state of ADaM programming



Current state of ADaM programming



Current state of ADaM programming



Summary of problems

- Multiple sources of truth

Summary of problems

- Multiple sources of truth
- Inefficient validation

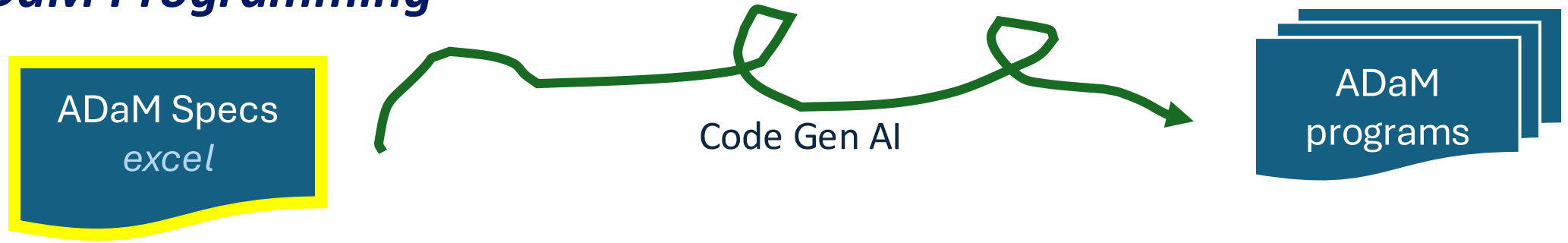
Summary of problems

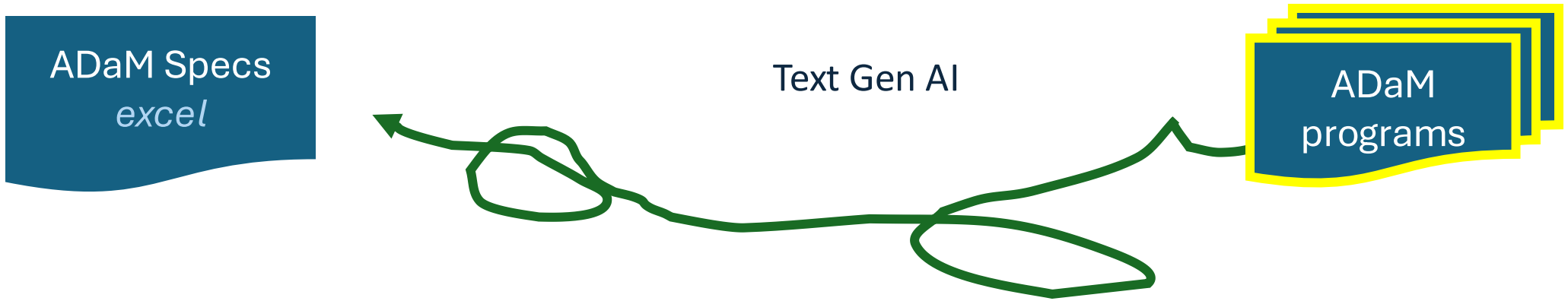
- Multiple sources of truth
- Inefficient validation
- Burdensome bookkeeping

Summary of problems

- Multiple sources of truth
- Inefficient validation
- Burdensome bookkeeping
- Lack of standards

ADaM Programming



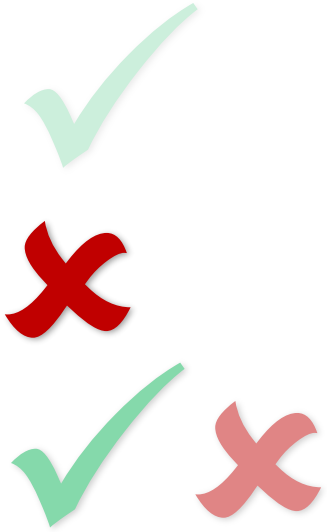


Summary of LLM-based approach

- Multiple sources of truth

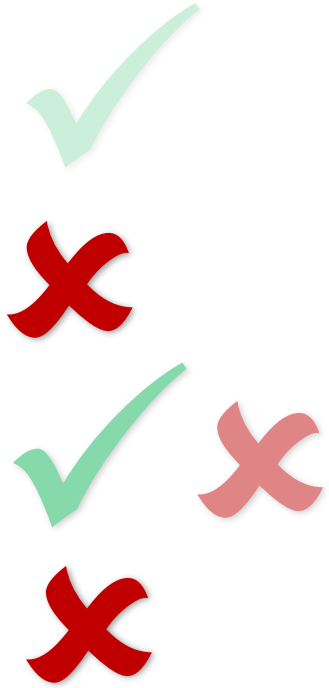
Summary of LLM-based approach

- Multiple sources of truth
- Inefficient validation
- Burdensome bookkeeping



Summary of LLM-based approach

- Multiple sources of truth
- Inefficient validation
- Burdensome bookkeeping
- Lack of standards



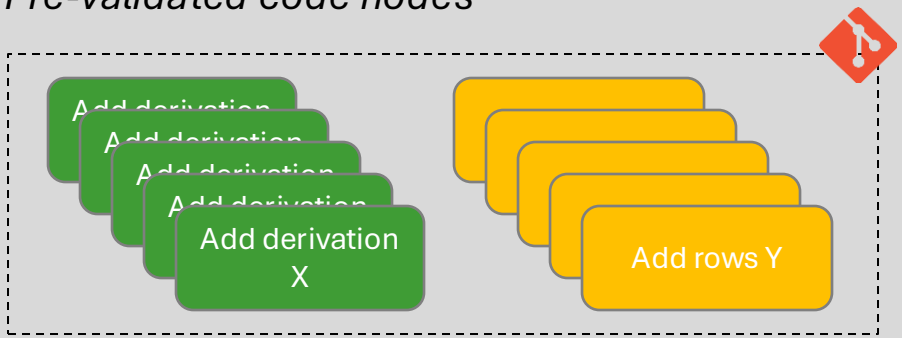
ADaM specification

- ✓ Predecessors
- ✓ Derivations
- ✓ Imputations
- ✓ Derived parameters
- ✓ …

ADaM specification

- ✓ Predecessors
- ✓ Derivations
- ✓ Imputations
- ✓ Derived parameters
- ✓ …

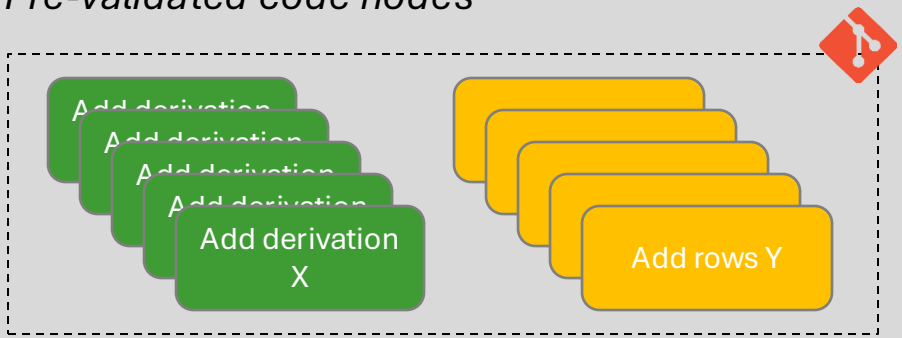
Pre-validated code nodes



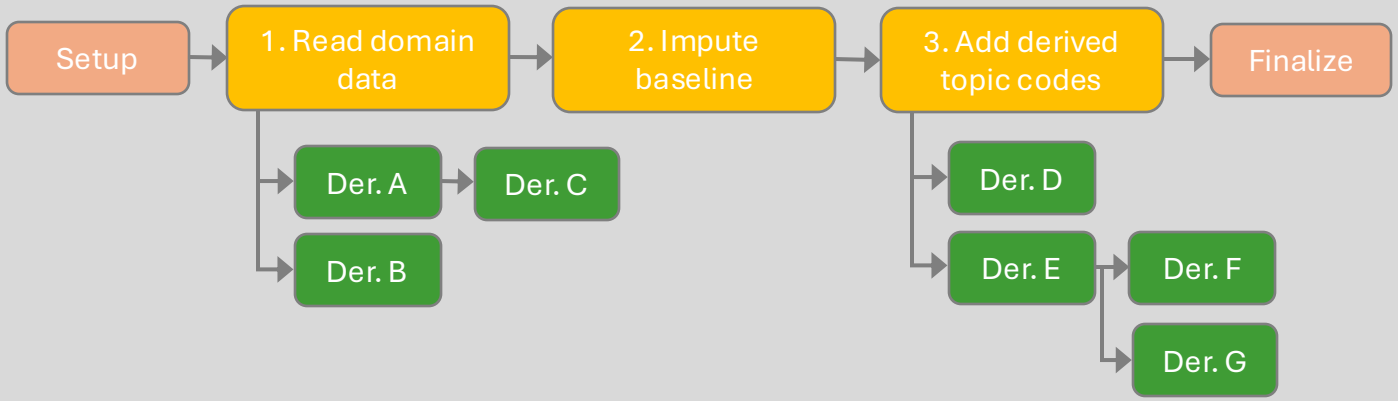
ADaM specification

- ✓ Predecessors
- ✓ Derivations
- ✓ Imputations
- ✓ Derived parameters
- ✓ ...

Pre-validated code nodes



Code dependency graph (example)



ADaM specification

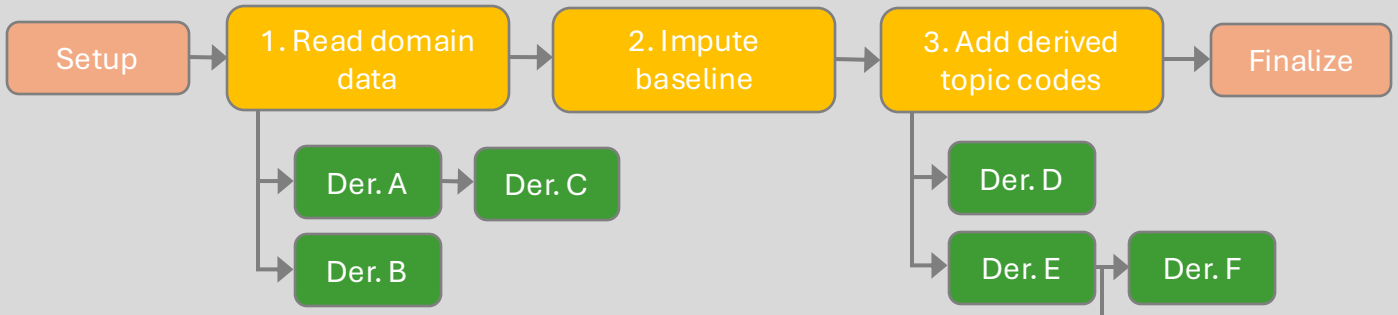
- ✓ Predecessors
- ✓ Derivations
- ✓ Imputations
- ✓ Derived parameters
- ✓ ...

Pre-validated code nodes

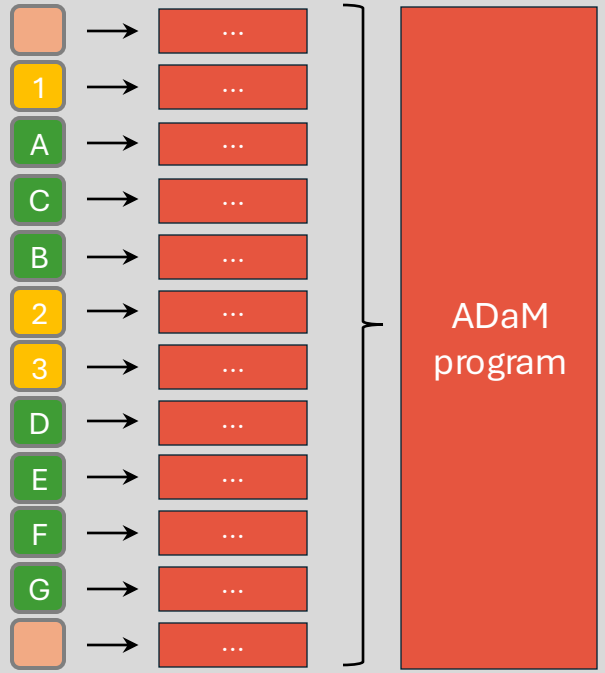
Add derivation X

Add rows Y

Code dependency graph (example)



Generate program



ADaM specification

- ✓ Predecessors
- ✓ Derivations
- ✓ Imputations
- ✓ Derived parameters
- ✓ …

Pre-validated code nodes

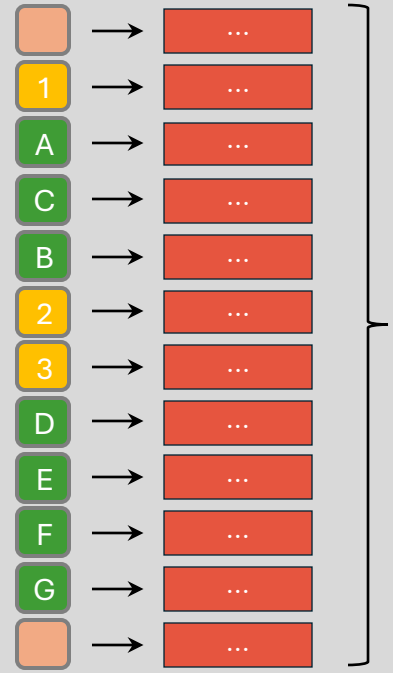
Add derivation X

Add rows Y

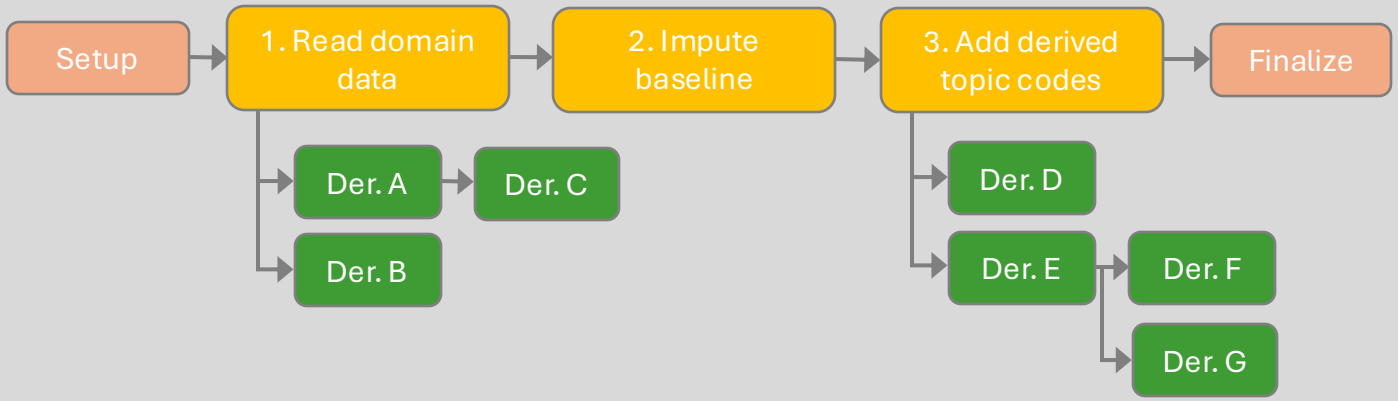
define.xml

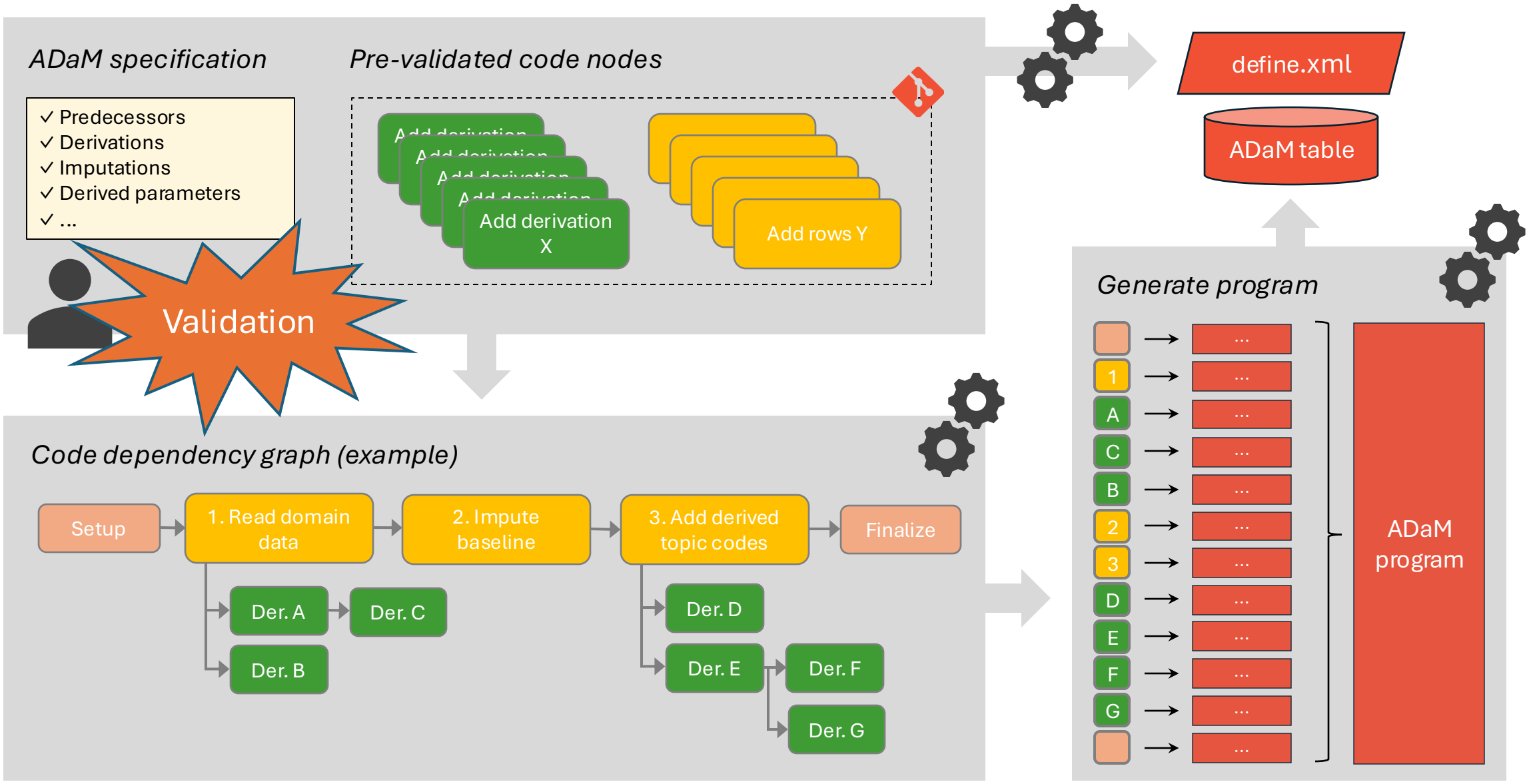
ADaM table

Generate program



Code dependency graph (example)




ADaM specification
✓ Predecessors
✓ Derivations
✓ Imputations
✓ Derived parameters
✓ ...
Validation
Pre-validated code nodes
Add derivation X
Add rows Y
define.xml
ADaM table
Generate program
ADaM program
Code dependency graph (example)
Setup
1. Read domain data
2. Impute baseline
3. Add derived topic codes
Finalize
Der. A
Der. C
Der. B
Der. D
Der. E
Der. F
Der. G

```
{{self}} <- .self |>
  dplyr::mutate(
    {{variable}} = admiral::compute_duration(
      start_date = TRTSDT,
      end_date = {{date}},
      in_unit = "days",
      out_unit = "days",
      add_one = TRUE
    )
  )
```

ADY

```
adlb <- exmapleFunction(
d_datain = bind_rows(
adlb_11_homair,
adlb_11_leptin
), v_dtvar = "LBDTC",
v_dtout = "ADT")
```

LBDTC

```
ADLB <- ADLB |>
dplyr::mutate(
ADT = admiral::convert_dtc_to_dt(
dtc = LBDTC, highest_imputation = "M",
date_imputation = "mid"),
ADTF = admiral::compute_dtf(
dtc = LBDTC, dt = ADT)
```

ADT

```
ex %>%
 mutate(EXSTD = ymd(EXSTDTC)) %>%
 filter(!is.na(EXSTD)) %>%
 group_by(USUBJID) %>%
 summarise(TRTSDT = min(EXSTD), .groups = "drop")
```

TRTSDT

```
lower <- baseline_window[1]
upper <- baseline_window[2]
bds2 <- bds %>% mutate(.row_id = row_number())
base_candidates <- bds2 %>%
filter(!is.na(AVAL), !is.na(ADT), !is.na(TRTSDT)) %>%
mutate(rel_day = as.integer(ADT - TRTSDT)) %>%
filter(rel_day >= lower, rel_day <= upper)
base_pick <- base_candidates %>%
group_by(USUBJID, PARAMCD) %>%
filter(ADT == max(ADT, na.rm = TRUE)) %>%
{ if ("ADTM" %in% names(.)) {
arrange(., ADT, ADTM, .row_id, .by_group = TRUE) %>%
 slice_tail(n = 1)
} else {
arrange(., ADT, .row_id, .by_group = TRUE) %>% slice_tail(n = 1)}
} %>% ungroup() %>%
transmute(USUBJID, PARAMCD, BASE = AVAL, BASEDT = ADT)
```

BASE

ADY

```
{{self}} <- .self |>
  dplyr::mutate(
    {{variable}} = admiral::compute_duration(
      start_date = TRTSDT,
      end_date = {{date}},
      in_unit = "days",
      out_unit = "days",
      add_one = TRUE
    )
  )
```

LBDTC

```
adlb <- exmapleFunction(
d_datain = bind_rows(
adlb_11_homair,
adlb_11_leptin
), v_dtvar = "LBDTC",
v_dtout = "ADT")
```

ADT

```
ADLB <- ADLB |>
dplyr::mutate(
ADT = admiral::convert_dtc_to_dt(
dtc = LBDTC, highest_imputation = "M",
date_imputation = "mid"),
ADTF = admiral::compute_dtf(
dtc = LBDTC, dt = ADT)
```

TRTSDT

```
ex %>%
 mutate(EXSTD = ymd(EXSTDTC)) %>%
 filter(!is.na(EXSTD)) %>%
 group_by(USUBJID) %>%
 summarise(TRTSDT = min(EXSTD), .groups = "drop")
```

BASE

```
lower <- baseline_window[1]
upper <- baseline_window[2]
bds2 <- bds %>% mutate(.row_id = row_number())
base_candidates <- bds2 %>%
filter(!is.na(AVAL), !is.na(ADT), !is.na(TRTSDT)) %>%
mutate(rel_day = as.integer(ADT - TRTSDT)) %>%
filter(rel_day >= lower, rel_day <= upper)
base_pick <- base_candidates %>%
group_by(USUBJID, PARAMCD) %>%
filter(ADT == max(ADT, na.rm = TRUE)) %>%
{ if ("ADTM" %in% names(.)) {
arrange(., ADT, ADTM, .row_id, .by_group = TRUE) %>%
 slice_tail(n = 1)
} else {
arrange(., ADT, .row_id, .by_group = TRUE) %>% slice_tail(n = 1)}
} %>% ungroup() %>%
transmute(USUBJID, PARAMCD, BASE = AVAL, BASEDT = ADT)
```

Standards repo

ADY

```
{{self}} <- .self |>
 dplyr::mutate(
  {{variable}} = admiral::compute_duration(
   start_date = TRTSDT,
   end_date = {{date}},
   in_unit = "days",
   out_unit = "days",
   add_one = TRUE
  )
 )
```

LBDTC

```
adlb <- exmapleFunction(
d_datain = bind_rows(
adlb_11_homair,
adlb_11_leptin
), v_dtvar = "LBDTC",
v_dtout = "ADT")
```

ADT

```
ADLB <- ADLB |>
dplyr::mutate(
ADT = admiral::convert_dtc_to_dt(
dtc = LBDTC, highest_imputation = "M",
date_imputation = "mid"),
ADTF = admiral::compute_dtf(
dtc = LBDTC, dt = ADT)
```

TRTSDT

```
ex %>%
 mutate(EXSTD = ymd(EXSTDTC)) %>%
 filter(!is.na(EXSTD)) %>%
 group_by(USUBJID) %>%
 summarise(TRTSDT = min(EXSTD), .groups = "drop")
```

BASE

```
lower <- baseline_window[1]
upper <- baseline_window[2]
bds2 <- bds %>% mutate(.row_id = row_number())
base_candidates <- bds2 %>%
filter(!is.na(AVAL), !is.na(ADT), !is.na(TRTSDT)) %>%
mutate(rel_day = as.integer(ADT - TRTSDT)) %>%
filter(rel_day >= lower, rel_day <= upper)
base_pick <- base_candidates %>%
group_by(USUBJID, PARAMCD) %>%
filter(ADT == max(ADT, na.rm = TRUE)) %>%
{ if ("ADTM" %in% names(.)) {
arrange(., ADT, ADTM, .row_id, .by_group = TRUE) %>%
 slice_tail(n = 1)
} else {
arrange(., ADT, .row_id, .by_group = TRUE) %>% slice_tail(n = 1)}
} %>% ungroup() %>%
transmute(USUBJID, PARAMCD, BASE = AVAL, BASEDT = ADT)
```

Standards repo

Study repo

```
{{self}} <- .self |>
 dplyr::mutate(
  {{variable}} = admiral::compute_duration(
   start_date = TRTSDT,
   end_date = {{date}},
   in_unit = "days",
   out_unit = "days",
   add_one = TRUE
  )
 )
```

ADY

```
adlb <- exmapleFunction(
d_datain = bind_rows(
adlb_11_homair,
adlb_11_leptin
), v_dtvar = "LBDTC",
v_dtout = "ADT")
```

LBDTC

```
ADLB <- ADLB |>
dplyr::mutate(
ADT = admiral::convert_dtc_to_dt(
dtc = LBDTC, highest_imputation = "M",
date_imputation = "mid"),
ADTF = admiral::compute_dtf(
dtc = LBDTC, dt = ADT)
```

ADT

```
ex %>%
 mutate(EXSTD = ymd(EXSTDTC)) %>%
 filter(!is.na(EXSTD)) %>%
 group_by(USUBJID) %>%
 summarise(TRTSDT = min(EXSTD), .groups = "drop")
```

TRTSDT

```
lower <- baseline_window[1]
upper <- baseline_window[2]
bds2 <- bds %>% mutate(.row_id = row_number())
base_candidates <- bds2 %>%
filter(!is.na(AVAL), !is.na(ADT), !is.na(TRTSDT)) %>%
mutate(rel_day = as.integer(ADT - TRTSDT)) %>%
filter(rel_day >= lower, rel_day <= upper)
base_pick <- base_candidates %>%
group_by(USUBJID, PARAMCD) %>%
filter(ADT == max(ADT, na.rm = TRUE)) %>%
{ if ("ADTM" %in% names(.)) {
arrange(., ADT, ADTM, .row_id, .by_group = TRUE) %>%
 slice_tail(n = 1)
} else {
arrange(., ADT, .row_id, .by_group = TRUE) %>% slice_tail(n = 1)}
} %>% ungroup() %>%
transmute(USUBJID, PARAMCD, BASE = AVAL, BASEDT = ADT)
```

BASE

Standards repo

```
column:
  ADT:
    code_id: adt_std

  BASE:
    code_id: base_std

  ADY:
    code_id: ady_std
```

Study repo

```
{{self}} <- .self |>
 dplyr::mutate(
  {{variable}} = admiral::compute_duration(
   start_date = TRTSDT,
   end_date = {{date}},
   in_unit = "days",
   out_unit = "days",
   add_one = TRUE
  )
 )
```

ADY

```
adlb <- exmapleFunction(
d_datain = bind_rows(
adlb_11_homair,
adlb_11_leptin
), v_dtvar = "LBDTC",
v_dtout = "ADT")
```

LBDTC

```
ADLB <- ADLB |>
dplyr::mutate(
ADT = admiral::convert_dtc_to_dt(
dtc = LBDTC, highest_imputation = "M",
date_imputation = "mid"),
ADTF = admiral::compute_dtf(
dtc = LBDTC, dt = ADT)
```

ADT

```
ex %>%
 mutate(EXSTD = ymd(EXSTDTC)) %>%
 filter(!is.na(EXSTD)) %>%
 group_by(USUBJID) %>%
 summarise(TRTSDT = min(EXSTD), .groups = "drop")
```

TRTSDT

```
lower <- baseline_window[1]
upper <- baseline_window[2]
bds2 <- bds %>% mutate(.row_id = row_number())
base_candidates <- bds2 %>%
filter(!is.na(AVAL), !is.na(ADT), !is.na(TRTSDT)) %>%
mutate(rel_day = as.integer(ADT - TRTSDT)) %>%
filter(rel_day >= lower, rel_day <= upper)
base_pick <- base_candidates %>%
group_by(USUBJID, PARAMCD) %>%
filter(ADT == max(ADT, na.rm = TRUE)) %>%
{ if ("ADTM" %in% names(.)) {
arrange(., ADT, ADTM, .row_id, .by_group = TRUE) %>%
 slice_tail(n = 1)
} else {
arrange(., ADT, .row_id, .by_group = TRUE) %>% slice_tail(n = 1)}
} %>% ungroup() %>%
transmute(USUBJID, PARAMCD, BASE = AVAL, BASEDT = ADT)
```

BASE

```
column:
  ADT:
    code_id: adt_std
  BASE:
    code_id: base_std
  ADY:
    code_id: ady_std
```

ADY

```
{{self}} <- .self |>
 dplyr::mutate(
  {{variable}} = admiral::compute_duration(
   start_date = TRTSDT,
   end_date = {{date}},
   in_unit = "days",
   out_unit = "days",
   add_one = TRUE
  )
 )
```

LBDTC

```
adlb <- exmapleFunction(
d_datain = bind_rows(
adlb_11_homair,
adlb_11_leptin
), v_dtvar = "LBDTC",
v_dtout = "ADT")
```

ADT

```
ADLB <- ADLB |>
dplyr::mutate(
ADT = admiral::convert_dtc_to_dt(
dtc = LBDTC, highest_imputation = "M",
date_imputation = "mid"),
ADTF = admiral::compute_dtf(
dtc = LBDTC, dt = ADT)
```

TRTSDT

```
ex %>%
 mutate(EXSTD = ymd(EXSTDTC)) %>%
 filter(!is.na(EXSTD)) %>%
 group_by(USUBJID) %>%
 summarise(TRTSDT = min(EXSTD), .groups = "drop")
```

BASE

```
lower <- baseline_window[1]
upper <- baseline_window[2]
bds2 <- bds %>% mutate(.row_id = row_number())
base_candidates <- bds2 %>%
filter(!is.na(AVAL), !is.na(ADT), !is.na(TRTSDT)) %>%
mutate(rel_day = as.integer(ADT - TRTSDT)) %>%
filter(rel_day >= lower, rel_day <= upper)
base_pick <- base_candidates %>%
group_by(USUBJID, PARAMCD) %>%
filter(ADT == max(ADT, na.rm = TRUE)) %>%
{ if ("ADTM" %in% names(.)) {
arrange(., ADT, ADTM, .row_id, .by_group = TRUE) %>%
 slice_tail(n = 1)
} else {
arrange(., ADT, .row_id, .by_group = TRUE) %>% slice_tail(n = 1)}
} %>% ungroup() %>%
transmute(USUBJID, PARAMCD, BASE = AVAL, BASEDT = ADT)
```


```
column:
  ADT:
    code_id: adt_std
  BASE:
    code_id: base_std
  ADY:
    code_id: ady_std
```

```
{{self}} <- .self |>
 dplyr::mutate(
  {{variable}} = admiral::compute_duration(
   start_date = TRTSDT,
   end_date = {{date}},
   in_unit = "days",
   out_unit = "days",
   add_one = TRUE
  )
 )
```

ADY

```
adlb <- exmapleFunction(
d_datain = bind_rows(
adlb_11_homair,
adlb_11_leptin
), v_dtvar = "LBDTC",
v_dtout = "ADT")
```

LBDTC

```
ADLB <- ADLB |>
dplyr::mutate(
ADT = admiral::convert_dtc_to_dt(
dtc = LBDTC, highest_imputation = "M",
date_imputation = "mid"),
ADTF = admiral::compute_dtf(
dtc = LBDTC, dt = ADT)
```

ADT

```
ex %>%
 mutate(EXSTD = ymd(EXSTDTC)) %>%
 filter(!is.na(EXSTD)) %>%
 group_by(USUBJID) %>%
 summarise(TRTSDT = min(EXSTD), .groups = "drop")
```

TRTSDT

```
lower <- baseline_window[1]
upper <- baseline_window[2]
bds2 <- bds %>% mutate(.row_id = row_number())
base_candidates <- bds2 %>%
filter(!is.na(AVAL), !is.na(ADT), !is.na(TRTSDT)) %>%
mutate(rel_day = as.integer(ADT - TRTSDT)) %>%
filter(rel_day >= lower, rel_day <= upper)
base_pick <- base_candidates %>%
group_by(USUBJID, PARAMCD) %>%
filter(ADT == max(ADT, na.rm = TRUE)) %>%
{ if ("ADTM" %in% names(.)) {
arrange(., ADT, ADTM, .row_id, .by_group = TRUE) %>%
 slice_tail(n = 1)
} else {
arrange(., ADT, .row_id, .by_group = TRUE) %>% slice_tail(n = 1)}
} %>% ungroup() %>%
transmute(USUBJID, PARAMCD, BASE = AVAL, BASEDT = ADT)
```

BASE


```
column:
  ADT:
    code_id: adt_std
  BASE:
    code_id: base_std
  ADY:
    code_ide: ady_std
```

```
.self <- .self |>
 dplyr::mutate(
  {{variable}} = admiral::compute_duration(
   start_date = TRTSDT,
   end_date = {{date}},
   in_unit = "days",
   out_unit = "days",
   add_one = TRUE
  )
 )
```

ADY

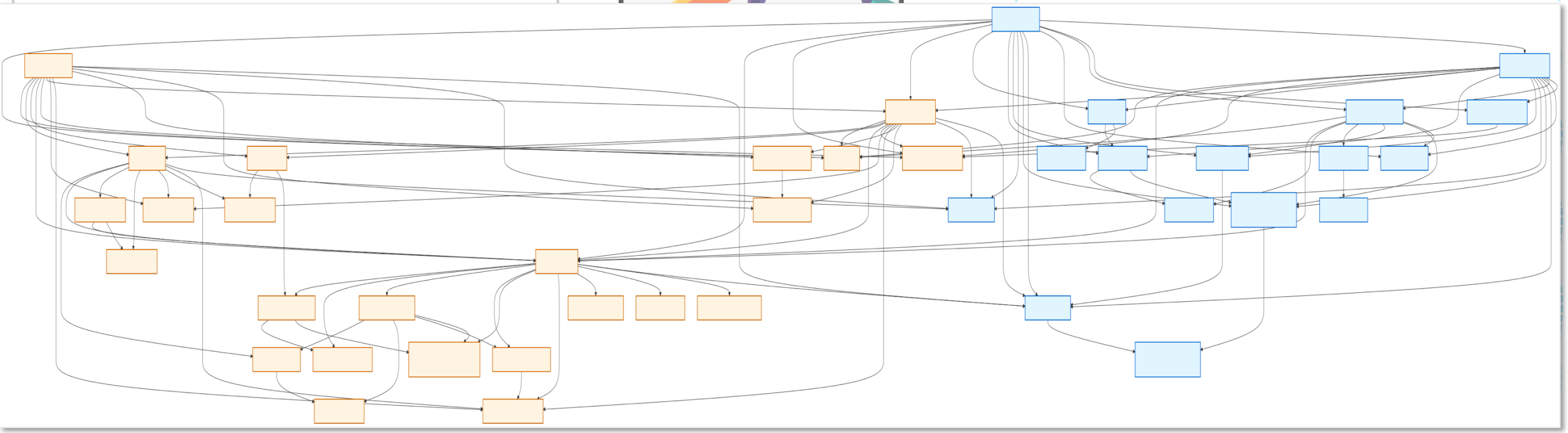
```
adlb <- exmapleFunction(
d_datain = bind_rows(
adlb_11_homair,
adlb_11_leptin
), v_dtvar = "LBDTC",
v_dtout = "ADT")
```

LBDTC

```
ADLB <- ADLB |>
dplyr::mutate(
ADT = admiral::convert_dtc_to_dt(
dtc = LBDTC, highest_imputation = "M",
date_imputation = "mid"),
ADTF = admiral::compute_dtf(
dtc = LBDTC, dt = ADT)
```

ADT

```
column:
  ADT:
    code_id: adt_std
  BASE:
    code_id: base_std
  ADY:
    code_ide: ady_std
```



```
.self <- .self |>
 dplyr::mutate(
  {{variable}} = admiral::compute_duration(
   start_date = TRTSDT,
   end_date = {{date}},
   in_unit = "days",
   out_unit = "days",
   add_one = TRUE
  )
 )
```

ADY

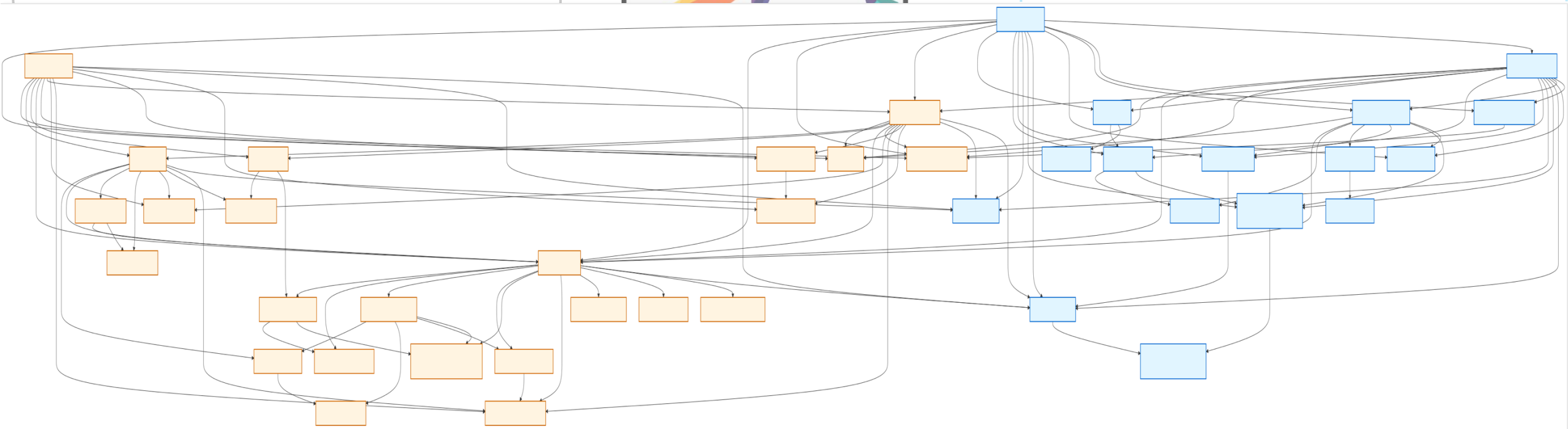
```
adlb <- exmapleFunction(
d_datain = bind_rows(
adlb_11_homair,
adlb_11_leptin
), v_dtvar = "LBDTC",
v_dtout = "ADT")
```

LBDTC

```
ADLB <- ADLB |>
dplyr::mutate(
ADT = admiral::convert_dtc_to_dt(
dtc = LBDTC, highest_imputation = "M",
date_imputation = "mid"),
ADTF = admiral::compute_dtf(
dtc = LBDTC, dt = ADT)
```

ADT

```
column:
  ADT:
    code_id: adt_std
  BASE:
    code_id: base_std
  ADY:
    code_ide: ady_std
```



```
#' @title Analysis relative day
#' @description
#' Derives the relative day compared to the treatment start date.
#' @type derivation
#' @depends ADLB ADT
#' @depends ADLB TRTSDT
#' @outputs ADY
#' @code

ADLB <- ADLB |>
  dplyr::mutate(
    ADY = admiral::compute_duration(
      start_date = TRTSDT,
      end_date = ADT,
      in_unit = "days",
      out_unit = "days",
      add_one = TRUE
    )
  )
```

```
#' @title Analysis relative day
#' @description
#' Derives the relative day compared to the treatment start date.
#' @type derivation
#' @depends ADLB ADT
#' @depends ADLB TRTSDT
#' @outputs ADY
#' @code
```

```
ADLB <- ADLB |>
  dplyr::mutate(
    ADY = admiral::compute_duration(
      start_date = TRTSDT,
      end_date = ADT,
      in_unit = "days",
      out_unit = "days",
      add_one = TRUE
    )
  )
```

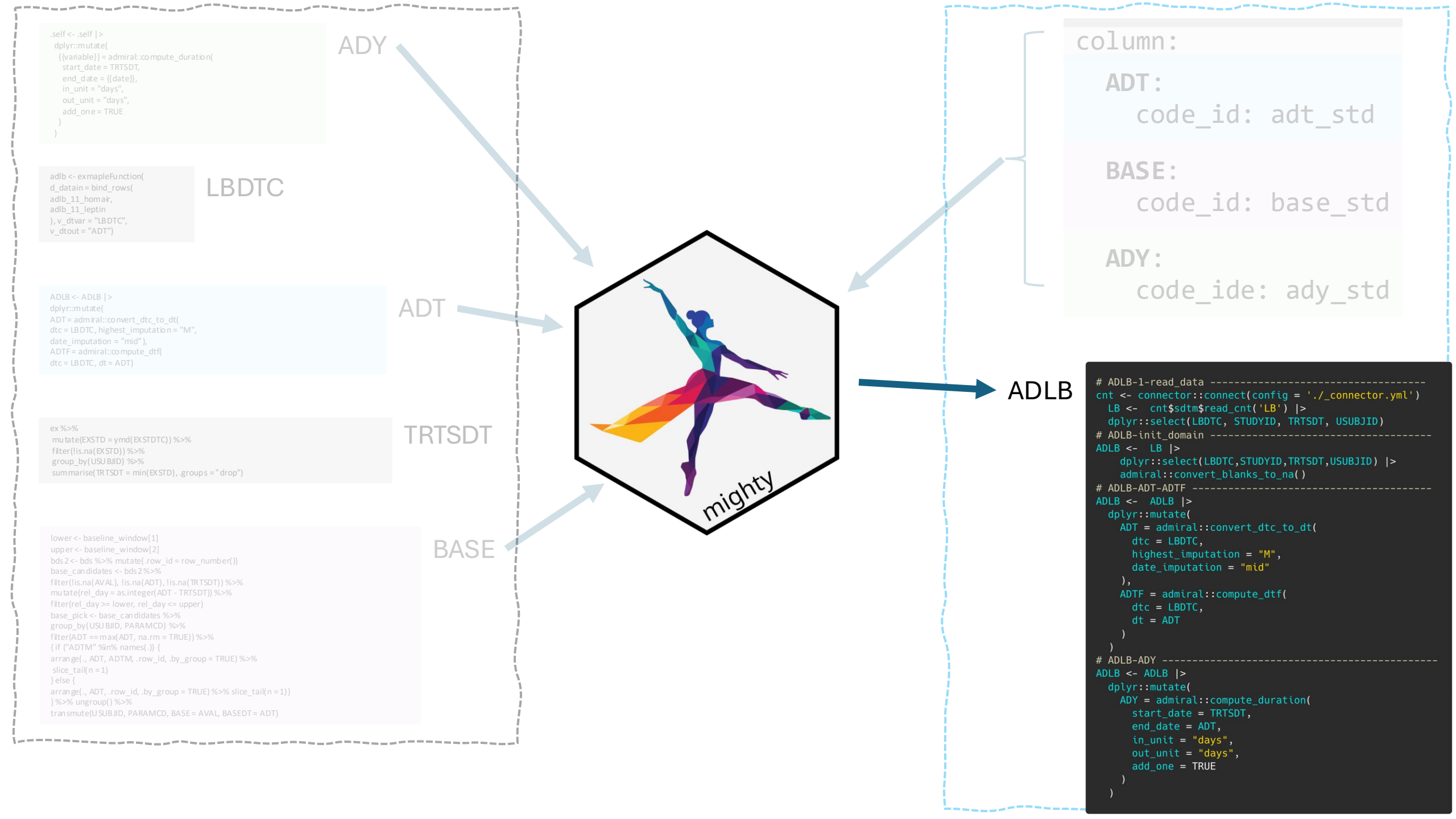
```
#' @title Analysis relative day
#' @description
#' Derives the relative day compared to the treatment start date.
#' @type derivation
#' @depends ADLB ADT
#' @depends ADLB TRTSDT
#' @outputs ADY
#' @code
```

```
ADLB <- ADLB |>
  dplyr::mutate(
    ADY = admiral::compute_duration(
      start_date = TRTSDT,
      end_date = ADT,
      in_unit = "days",
      out_unit = "days",
      add_one = TRUE
    )
  )
```

```
#' @title Analysis relative day
#' @description
#' Derives the relative day compared to the treatment start date.
#' @type derivation
#' @depends ADLB ADT
#' @depends ADLB TRTSDT
#' @outputs ADY
#' @code
```

```
ADLB <- ADLB |>
  dplyr::mutate(
    ADY = admiral::compute_duration(
      start_date = TRTSDT,
      end_date = ADT,
      in_unit = "days",
      out_unit = "days",
      add_one = TRUE
    )
  )
```

```
#' @title Analysis relative day
#' @description
#' Derives the relative day compared to the treatment start date.
#' @param variable `character` Name of new variable to create
#' @param date `character` Name of date variable to use
#' @type derivation
#' @depends {{domain}} {{date}}
#' @depends {{domain}} TRTSDT
#' @outputs {{variable}}
#' @code
{{domain}} <- {{domain}} |>
  dplyr::mutate(
    {{variable}} = admiral::compute_duration(
      start_date = TRTSDT,
      end_date = {{date}},
      in_unit = 'days',
      out_unit = 'days',
      add_one = TRUE
    )
  )
```


.self <- .self |>
dplyr::mutate(
{{variable}} = admiral::compute_duration(
start_date = TRTSDT,
end_date = {{date}},
in_unit = "days",
out_unit = "days",
add_one = TRUE
)
)
ADY
adlb <- exmapleFunction(
d_datain = bind_rows(
adlb_11_homair,
adlb_11_leptin
), v_dtvar = "LBDTC",
v_dtout = "ADT")
LBDTC
ADLB <- ADLB |>
dplyr::mutate(
ADT = admiral::convert_dtc_to_dt(
dtc = LBDTC, highest_imputation = "M",
date_imputation = "mid"),
ADTF = admiral::compute_dtf(
dtc = LBDTC, dt = ADT)
ADT
ex %>%
mutate(EXSTD = ymd(EXSTDTC)) %>%
filter(!is.na(EXSTD)) %>%
group_by(USUBJID) %>%
summarise(TRTSDT = min(EXSTD), .groups = "drop")
TRTSDT
lower <- baseline_window[1]
upper <- baseline_window[2]
bds2 <- bds %>% mutate(.row_id = row_number())
base_candidates <- bds2 %>%
filter(!is.na(AVAL), !is.na(ADT), !is.na(TRTSDT)) %>%
mutate(rel_day = as.integer(ADT - TRTSDT)) %>%
filter(rel_day >= lower, rel_day <= upper)
base_pick <- base_candidates %>%
group_by(USUBJID, PARAMCD) %>%
filter(ADT == max(ADT, na.rm = TRUE)) %>%
{ if ("ADTM" %in% names(.)) {
arrange(., ADT, ADTM, .row_id, .by_group = TRUE) %>%
slice_tail(n = 1)
} else {
arrange(., ADT, .row_id, .by_group = TRUE) %>% slice_tail(n = 1)}
} %>% ungroup() %>%
transmute(USUBJID, PARAMCD, BASE = AVAL, BASEDT = ADT)
BASE
mighty
column:
ADT:
code_id: adt_std
BASE:
code_id: base_std
ADY:
code_ide: ady_std
ADLB
# ADLB-1-read_data ----------
cnt <- connector::connect(config = './_connector.yml')
LB <- cnt$sdtm$read_cnt('LB') |>
dplyr::select(LBDTC, STUDYID, TRTSDT, USUBJID)
# ADLB-init_domain ----------
ADLB <- LB |>
dplyr::select(LBDTC,STUDYID,TRTSDT,USUBJID) |>
admiral::convert_blanks_to_na()
# ADLB-ADT-ADTF ----------
ADLB <- ADLB |>
dplyr::mutate(
ADT = admiral::convert_dtc_to_dt(
dtc = LBDTC,
highest_imputation = "M",
date_imputation = "mid"
),
ADTF = admiral::compute_dtf(
dtc = LBDTC,
dt = ADT
)
)
# ADLB-ADY ----------
ADLB <- ADLB |>
dplyr::mutate(
ADY = admiral::compute_duration(
start_date = TRTSDT,
end_date = ADT,
in_unit = "days",
out_unit = "days",
add_one = TRUE
)
)

```r
# ADLB-1-read_data -----------------------------------
cnt <- connector::connect(config = './_connector.yml')
  LB <-  cnt$sdtm$read_cnt('LB') |>
  dplyr::select(LBDTC, STUDYID, TRTSDT, USUBJID)
# ADLB-init_domain -----------------------------------
ADLB <-  LB |>
    dplyr::select(LBDTC,STUDYID,TRTSDT,USUBJID) |>
    admiral::convert_blanks_to_na()
# ADLB-ADT-ADTF --------------------------------------
ADLB <-  ADLB |>
  dplyr::mutate(
    ADT = admiral::convert_dtc_to_dt(
      dtc = LBDTC,
      highest_imputation = "M",
      date_imputation = "mid"
    ),
    ADTF = admiral::compute_dtf(
      dtc = LBDTC,
      dt = ADT
    )
  )
# ADLB-ADY -------------------------------------------
ADLB <- ADLB |>
  dplyr::mutate(
    ADY = admiral::compute_duration(
      start_date = TRTSDT,
      end_date = ADT,
      in_unit = "days",
      out_unit = "days",
      add_one = TRUE
    )
  )
```

```r
# ADLB-1-read_data -----------------------------------
cnt <- connector::connect(config = './_connector.yml')
  LB <-  cnt$sdtm$read_cnt('LB') |>
  dplyr::select(LBDTC, STUDYID, TRTSDT, USUBJID)
# ADLB-init_domain -----------------------------------
ADLB <-  LB |>
    dplyr::select(LBDTC,STUDYID,TRTSDT,USUBJID) |>
    admiral::convert_blanks_to_na()
# ADLB-ADT-ADTF --------------------------------------
ADLB <-  ADLB |>
  dplyr::mutate(
    ADT = admiral::convert_dtc_to_dt(
      dtc = LBDTC,
      highest_imputation = "M",
      date_imputation = "mid"
    ),
    ADTF = admiral::compute_dtf(
      dtc = LBDTC,
      dt = ADT
    )
  )
# ADLB-ADY -------------------------------------------
ADLB <- ADLB |>
  dplyr::mutate(
    ADY = admiral::compute_duration(
      start_date = TRTSDT,
      end_date = ADT,
      in_unit = "days",
      out_unit = "days",
      add_one = TRUE
    )
  )
```

```
# ADLB-1 read_data ------------------------------------------
cnt <- connector::connect(config = './_connector.yml')
  LB <-  cnt$sdtm$read_cnt('LB') |>
  dplyr::select(LBDTC, STUDYID, TRTSDT, USUBJID)
# ADLB-2 all_domain -----------------------------------------
ADLB <-  LB |>
    dplyr::select(LBDTC,STUDYID,TRTSDT,USUBJID) |>
    admiral::convert_blanks_to_na()
# ADLB-ADT-ADTF ---------------------------------------------
ADLB <-  ADLB |>
  dplyr::mutate(
    ADT = admiral::convert_dtc_to_dt(
      dtc = LBDTC,
      highest_imputation = "M",
      date_imputation = "mid"
    ),
    ADTF = admiral::compute_dtf(
      dtc = LBDTC,
      dt = ADT
    )
  )
# ADLB-ADY --------------------------------------------------
ADLB <- ADLB |>
  dplyr::mutate(
    ADY = admiral::compute_duration(
      start_date = TRTSDT,
      end_date = ADT,
      in_unit = "days",
      out_unit = "days",
      add_one = TRUE
    )
  )
```

```r
# ADLB-1 read_data ------------------------------------------------
cnt <- connector::connect(config = './_connector.yml')
  LB <-  cnt$sdtm$read_cnt('LB') |>
  dplyr::select(LBDTC, STUDYID, TRTSDT, USUBJID)
# ADLB-2 full_domain ----------------------------------------------
ADLB <-  LB |>
    dplyr::select(LBDTC,STUDYID,TRTSDT,USUBJID) |>
    admiral::convert_blanks_to_na()
# ADLB-ADT-ADTF ---------------------------------------------------
ADLB <-  ADLB |>
  dplyr::mutate(
    ADT = admiral::convert_dtc_to_dt(
      dtc = LBDTC,
      highest_imputation = "M",
      date_imputation = "mid"
    ),
    ADTF = admiral::compute_dtf(
      dtc = LBDTC,
      dt = ADT
    )
  )
# ADLB-ADY --------------------------------------------------------
ADLB <- ADLB |>
  dplyr::mutate(
    ADY = admiral::compute_duration(
      start_date = TRTSDT,
      end_date = ADT,
      in_unit = "days",
      out_unit = "days",
      add_one = TRUE
    )
  )
```

Cervan Girard

Vladimir Obucina

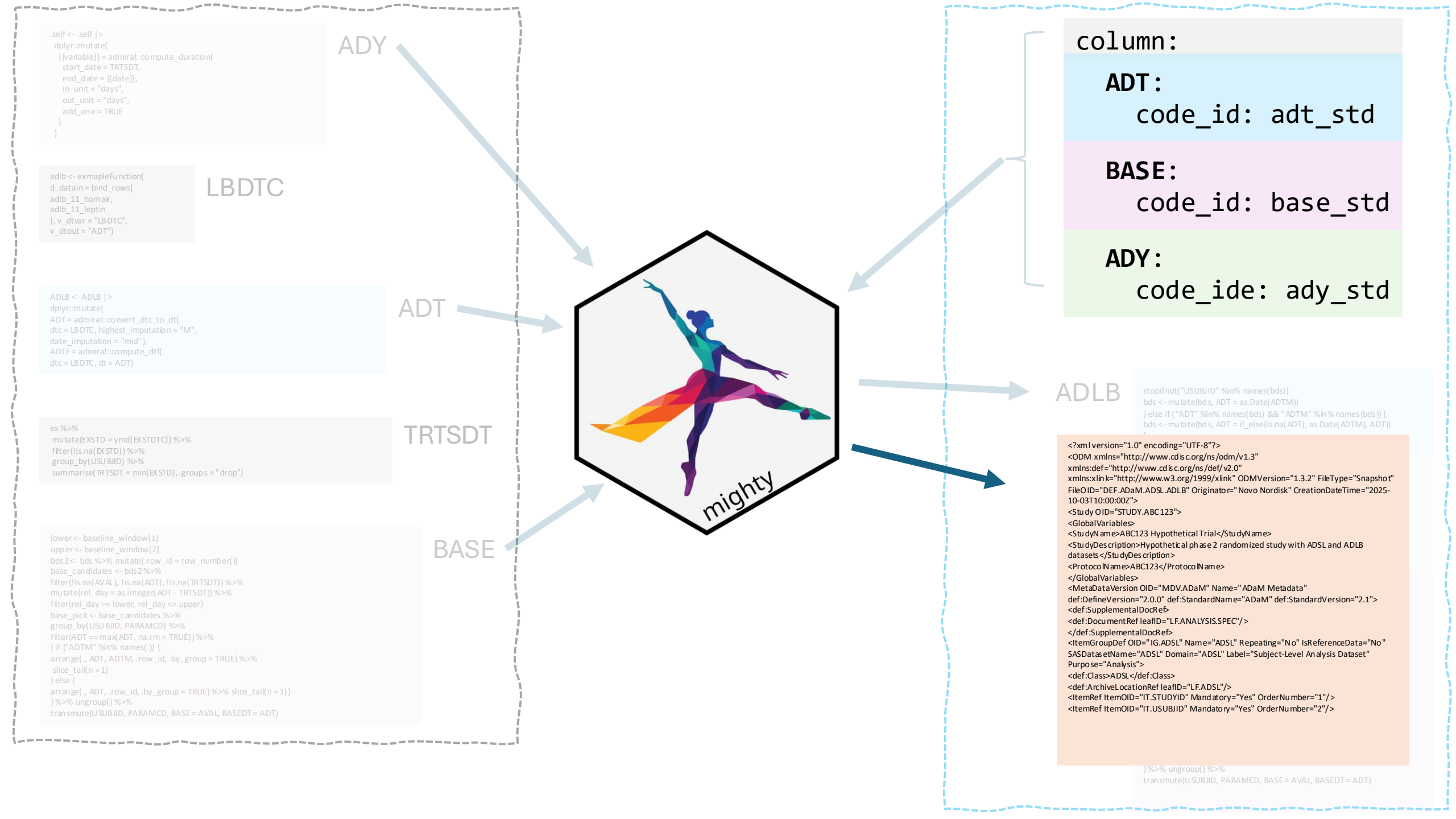
Aksel Thomsen

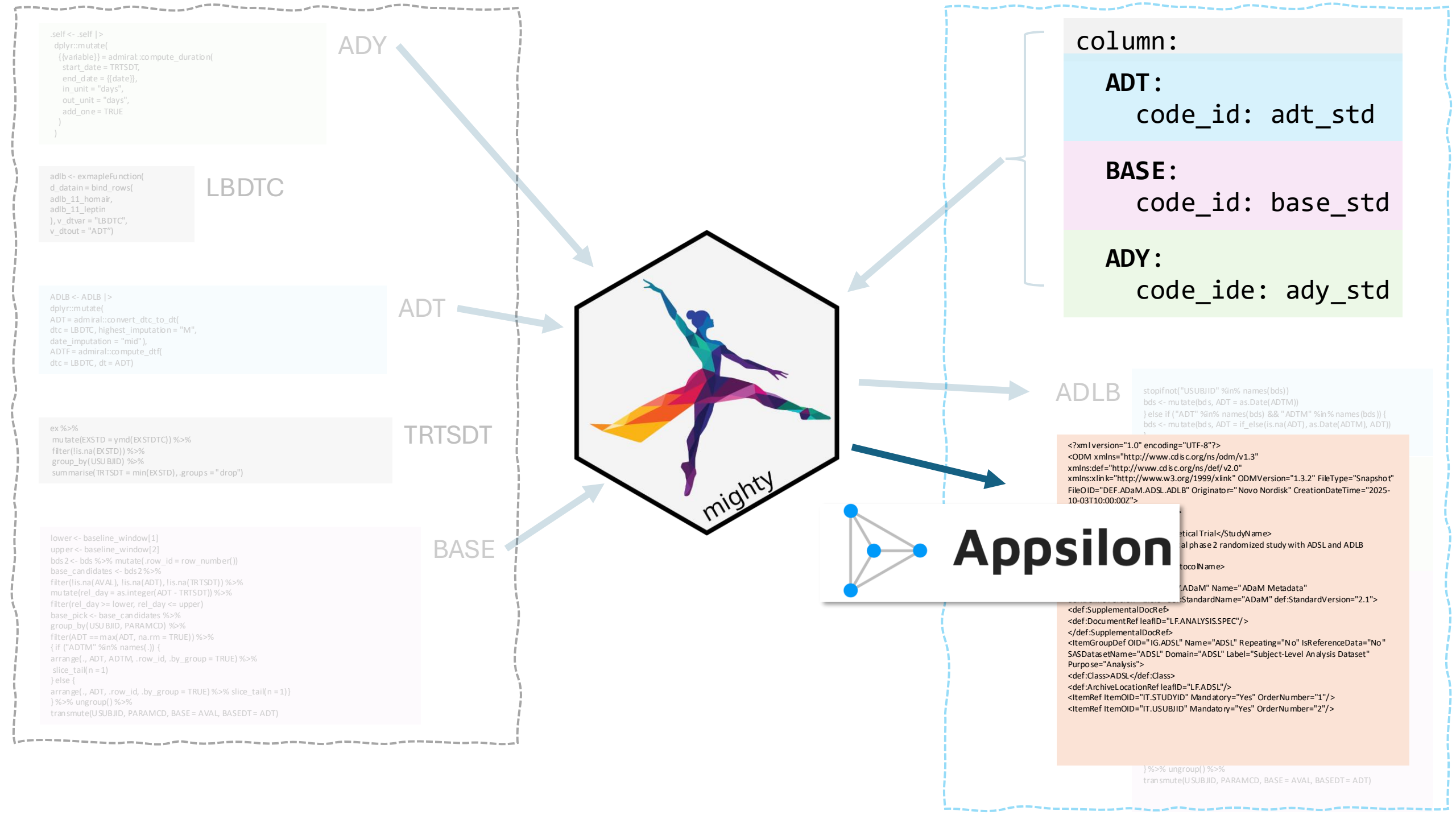
```r
# ADLB-1-read_data -----------------------------------
cnt <- connector::connect(config = './_connector.yml')
  LB <-  cnt$sdtm$read_cnt('LB') |>
  dplyr::select(LBDTC, STUDYID, TRTSDT, USUBJID)
# ADLB-init_domain -----------------------------------
ADLB <-  LB |>
    dplyr::select(LBDTC,STUDYID,TRTSDT,USUBJID) |>
    admiral::convert_blanks_to_na()
# ADLB-ADT-ADTF --------------------------------------
ADLB <-  ADLB |>
  dplyr::mutate(
    ADT = admiral::convert_dtc_to_dt(
      dtc = LBDTC,
      highest_imputation = "M",
      date_imputation = "mid"
    ),
    ADTF = admiral::compute_dtf(
      dtc = LBDTC,
      dt = ADT
    )
  )
# ADLB-ADY -------------------------------------------
ADLB <- ADLB |>
  dplyr::mutate(
    ADY = admiral::compute_duration(
      start_date = TRTSDT,
      end_date = ADT,
      in_unit = "days",
      out_unit = "days",
      add_one = TRUE
    )
  )
```

```r
# ADLB-1-read_data -----------------------------------
cnt <- connector::connect(config = './_connector.yml')
  LB <-  cnt$sdtm$read_cnt('LB') |>
  dplyr::select(LBDTC, STUDYID, TRTSDT, USUBJID)
# ADLB-init_domain -----------------------------------
ADLB <-  LB |>
    dplyr::select(LBDTC,STUDYID,TRTSDT,USUBJID) |>
    admiral::convert_blanks_to_na()
# ADLB-ADT-ADTF --------------------------------------
ADLB <-  ADLB |>
  dplyr::mutate(
    ADT = admiral::convert_dtc_to_dt(
      dtc = LBDTC,
      highest_imputation = "M",
      date_imputation = "mid"
    ),
    ADTF = admiral::compute_dtf(
      dtc = LBDTC,
      dt = ADT
    )
  )
# ADLB-ADY -------------------------------------------
ADLB <- ADLB |>
  dplyr::mutate(
    ADY = admiral::compute_duration(
      start_date = TRTSDT,
      end_date = ADT,
      in_unit = "days",
      out_unit = "days",
      add_one = TRUE
    )
  )
```

```r
# ADLB-1-read_data ------------------------------------
cnt <- connector::connect(config = './_connector.yml')
  LB <-  cnt$sdtm$read_cnt('LB') |>
  dplyr::select(LBDTC, STUDYID, TRTSDT, USUBJID)
# ADLB-init_domain ------------------------------------
ADLB <-  LB |>
    dplyr::select(LBDTC,STUDYID,TRTSDT,USUBJID) |>
    admiral::convert_blanks_to_na()
# ADLB-ADT-ADTF ---------------------------------------
ADLB <-  ADLB |>
  dplyr::mutate(
    ADT = admiral::convert_dtc_to_dt(
      dtc = LBDTC,
      highest_imputation = "M",
      date_imputation = "mid"
    ),
    ADTF = admiral::compute_dtf(
      dtc = LBDTC,
      dt = ADT
    )
  )
# ADLB-ADY --------------------------------------------
ADLB <- ADLB |>
  dplyr::mutate(
    ADY = admiral::compute_duration(
      start_date = TRTSDT,
      end_date = ADT,
      in_unit = "days",
      out_unit = "days",
      add_one = TRUE
    )
  )
```

```r
# ADLB-1-read_data ---------------------------------
cnt <- connector::connect(config = './_connector.yml')
  LB <-  cnt$sdtm$read_cnt('LB') |>
  dplyr::select(LBDTC, STUDYID, TRTSDT, USUBJID)
# ADLB-init_domain ---------------------------------
ADLB <-  LB |>
    dplyr::select(LBDTC,STUDYID,TRTSDT,USUBJID) |>
    admiral::convert_blanks_to_na()
# ADLB-ADT-ADTF ------------------------------------
ADLB <-  ADLB |>
  dplyr::mutate(
    ADT = admiral::convert_dtc_to_dt(
      dtc = LBDTC,
      highest_imputation = "M",
      date_imputation = "mid"
    ),
    ADTF = admiral::compute_dtf(
      dtc = LBDTC,
      dt = ADT
    )
  )
# ADLB-ADY -----------------------------------------
ADLB <- ADLB |>
  dplyr::mutate(
    ADY = admiral::compute_duration(
      start_date = TRTSDT,
      end_date = ADT,
      in_unit = "days",
      out_unit = "days",
      add_one = TRUE
    )
  )
```


.self <- .self |>
dplyr::mutate(
{{variable}} = admiral::compute_duration(
start_date = TRTSDT,
end_date = {{date}},
in_unit = "days",
out_unit = "days",
add_one = TRUE
)
)
ADY
adlb <- exmapleFunction(
d_datain = bind_rows(
adlb_11_homair,
adlb_11_leptin
), v_dtvar = "LBDTC",
v_dtout = "ADT")
LBDTC
ADLB <- ADLB |>
dplyr::mutate(
ADT = admiral::convert_dtc_to_dt(
dtc = LBDTC, highest_imputation = "M",
date_imputation = "mid"),
ADTF = admiral::compute_dtf(
dtc = LBDTC, dt = ADT)
ADT
ex %>%
mutate(EXSTD = ymd(EXSTDTC)) %>%
filter(!is.na(EXSTD)) %>%
group_by(USUBJID) %>%
summarise(TRTSDT = min(EXSTD), .groups = "drop")
TRTSDT
lower <- baseline_window[1]
upper <- baseline_window[2]
bds2 <- bds %>% mutate(.row_id = row_number())
base_candidates <- bds2 %>%
filter(!is.na(AVAL), !is.na(ADT), !is.na(TRTSDT)) %>%
mutate(rel_day = as.integer(ADT - TRTSDT)) %>%
filter(rel_day >= lower, rel_day <= upper)
base_pick <- base_candidates %>%
group_by(USUBJID, PARAMCD) %>%
filter(ADT == max(ADT, na.rm = TRUE)) %>%
{ if ("ADTM" %in% names(.)) {
arrange(., ADT, ADTM, .row_id, .by_group = TRUE) %>%
slice_tail(n = 1)
} else {
arrange(., ADT, .row_id, .by_group = TRUE) %>% slice_tail(n = 1)}
} %>% ungroup() %>%
transmute(USUBJID, PARAMCD, BASE = AVAL, BASEDT = ADT)
BASE
mighty
column:
ADT:
code_id: adt_std
BASE:
code_id: base_std
ADY:
code_ide: ady_std
ADLB
stopifnot("USUBJID" %in% names(bds))
bds <- mutate(bds, ADT = as.Date(ADTM))
} else if ("ADT" %in% names(bds) && "ADTM" %in% names(bds)) {
bds <- mutate(bds, ADT = if_else(is.na(ADT), as.Date(ADTM), ADT))
<?xml version="1.0" encoding="UTF-8"?>
<ODM xmlns="http://www.cdisc.org/ns/odm/v1.3"
xmlns:def="http://www.cdisc.org/ns/def/v2.0"
xmlns:xlink="http://www.w3.org/1999/xlink" ODMVersion="1.3.2" FileType="Snapshot"
FileOID="DEF.ADaM.ADSL.ADLB" Originator="Novo Nordisk" CreationDateTime="2025-10-03T10:00:00Z">
<Study OID="STUDY.ABC123">
<GlobalVariables>
<StudyName>ABC123 Hypothetical Trial</StudyName>
<StudyDescription>Hypothetical phase 2 randomized study with ADSL and ADLB datasets</StudyDescription>
<ProtocolName>ABC123</ProtocolName>
</GlobalVariables>
<MetaDataVersion OID="MDV.ADaM" Name="ADaM Metadata"
def:DefineVersion="2.0.0" def:StandardName="ADaM" def:StandardVersion="2.1">
<def:SupplementalDocRef>
<def:DocumentRef leafID="LF.ANALYSIS.SPEC"/>
</def:SupplementalDocRef>
<ItemGroupDef OID="IG.ADSL" Name="ADSL" Repeating="No" IsReferenceData="No"
SASDatasetName="ADSL" Domain="ADSL" Label="Subject-Level Analysis Dataset"
Purpose="Analysis">
<def:Class>ADSL</def:Class>
<def:ArchiveLocationRef leafID="LF.ADSL"/>
<ItemRef ItemOID="IT.STUDYID" Mandatory="Yes" OrderNumber="1"/>
<ItemRef ItemOID="IT.USUBJID" Mandatory="Yes" OrderNumber="2"/>
} %>% ungroup() %>%
transmute(USUBJID, PARAMCD, BASE = AVAL, BASEDT = ADT)


ADY
LBDTC
ADT
TRTSDT
BASE
mighty
column:
ADT:
code_id: adt_std
BASE:
code_id: base_std
ADY:
code_ide: ady_std
ADLB
Appsilon

Did we solve the problem

- Multiple sources of truth

Did we solve the problem

- Multiple sources of truth
- Inefficient validation

Did we solve the problem

- Multiple sources of truth
- Inefficient validation
- Burdensome bookkeeping

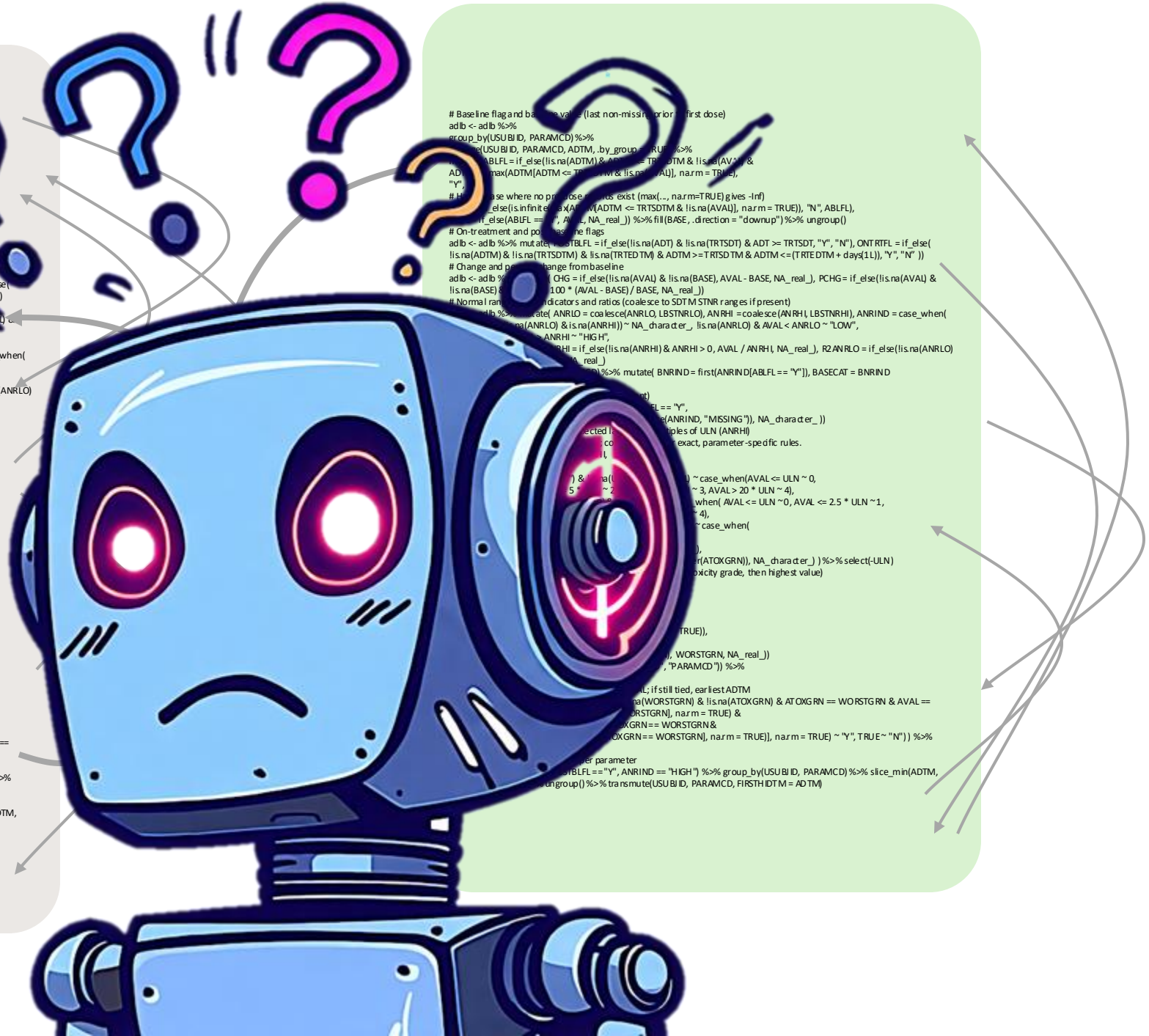
Did we solve the problem

- Multiple sources of truth
- Inefficient validation Burdensome bookkeeping
- Lack of standards

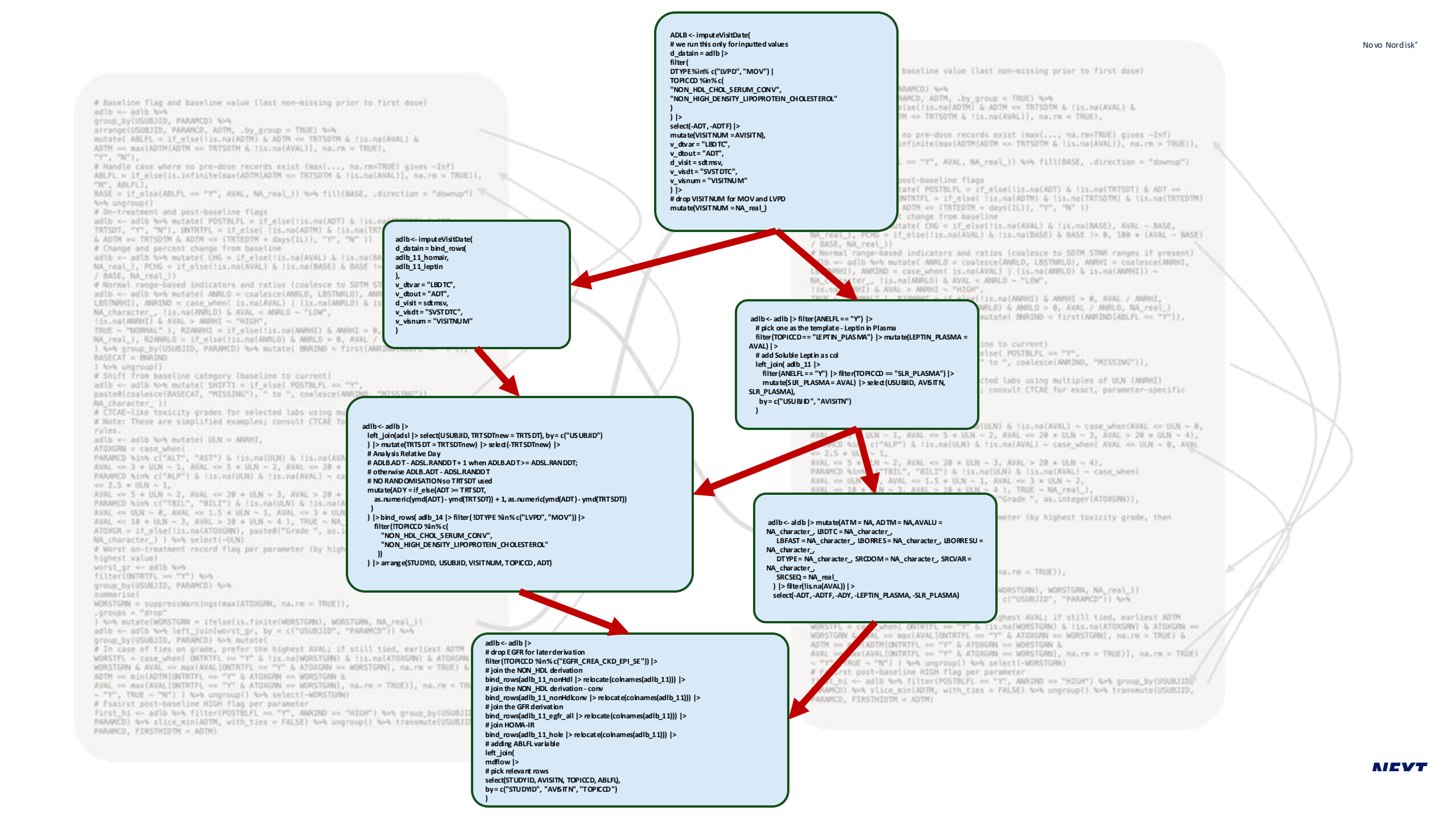
```
# Baseline flag and baseline value (last non-missing prior to first dose)
adlb <- adlb %>%
group_by(USUBJID, PARAMCD) %>%
arrange(USUBJID, PARAMCD, ADTM, .by_group = TRUE) %>%
mutate( ABLFL = if_else(!is.na(ADTM) & ADTM <= TRTSDTM & !is.na(AVAL) &
ADTM == max(ADTM[ADTM <= TRTSDTM & !is.na(AVAL)], na.rm = TRUE),
"Y", "N"),
# Handle case where no pre-dose records exist (max(..., na.rm=TRUE) gives -Inf)
ABLFL = if_else(is.infinite(max(ADTM[ADTM <= TRTSDTM & !is.na(AVAL)], na.rm = TRUE)), "N", ABLFL),
BASE = if_else(ABLFL == "Y", AVAL, NA_real_)) %>% fill(BASE, .direction = "downup") %>% ungroup()
# On-treatment and post-baseline flags
adlb <- adlb %>% mutate( POSTBLFL = if_else(!is.na(ADT) & !is.na(TRTSDT) & ADT >= TRTSDT, "Y", "N"), ONTRTFL = if_else(
!is.na(ADTM) & !is.na(TRTSDTM) & !is.na(TRTEDTM) & ADTM >= TRTSDTM & ADTM <= (TRTEDTM + days(1L)), "Y", "N" ))
# Change and percent change from baseline
adlb <- adlb %>% mutate( CHG = if_else(!is.na(AVAL) & !is.na(BASE), AVAL - BASE, NA_real_), PCHG = if_else(!is.na(AVAL) &
!is.na(BASE) & BASE != 0, 100 * (AVAL - BASE) / BASE, NA_real_))
# Normal range-based indicators and ratios (coalesce to SDTM STNR ranges if present)
adlb <- adlb %>% mutate( ANRLO = coalesce(ANRLO, LBSTNRLO), ANRHI = coalesce(ANRHI, LBSTNRHI), ANRIND = case_when(
is.na(AVAL) | (is.na(ANRLO) & is.na(ANRHI)) ~ NA_character_, !is.na(ANRLO) & AVAL < ANRLO ~ "LOW",
!is.na(ANRHI) & AVAL > ANRHI ~ "HIGH",
TRUE ~ "NORMAL" ), R2ANRHI = if_else(!is.na(ANRHI) & ANRHI > 0, AVAL / ANRHI, NA_real_), R2ANRLO = if_else(!is.na(ANRLO)
& ANRLO > 0, AVAL / ANRLO, NA_real_)
) %>% group_by(USUBJID, PARAMCD) %>% mutate( BNRIND = first(ANRIND[ABLFL == "Y"]), BASECAT = BNRIND
) %>% ungroup()
# Shift from baseline category (baseline to current)
adlb <- adlb %>% mutate( SHIFT1 = if_else( POSTBLFL == "Y",
paste0(coalesce(BASECAT, "MISSING"), " to ", coalesce(ANRIND, "MISSING")), NA_character_ ))
# CTCAE-like toxicity grades for selected labs using multiples of ULN (ANRHI)
# Note: These are simplified examples; consult CTCAE for exact, parameter-specific rules.
adlb <- adlb %>% mutate( ULN = ANRHI,
ATOXGRN = case_when(
PARAMCD %in% c("ALT", "AST") & !is.na(ULN) & !is.na(AVAL) ~ case_when(AVAL <= ULN ~ 0,
AVAL <= 3 * ULN ~ 1, AVAL <= 5 * ULN ~ 2, AVAL <= 20 * ULN ~ 3, AVAL > 20 * ULN ~ 4),
PARAMCD %in% c("ALP") & !is.na(ULN) & !is.na(AVAL) ~ case_when( AVAL <= ULN ~ 0, AVAL <= 2.5 * ULN ~ 1,
AVAL <= 5 * ULN ~ 2, AVAL <= 20 * ULN ~ 3, AVAL > 20 * ULN ~ 4),
PARAMCD %in% c("TBIL", "BILI") & !is.na(ULN) & !is.na(AVAL) ~ case_when(
AVAL <= ULN ~ 0, AVAL <= 1.5 * ULN ~ 1, AVAL <= 3 * ULN ~ 2,
AVAL <= 10 * ULN ~ 3, AVAL > 10 * ULN ~ 4 ), TRUE ~ NA_real_),
ATOXGR = if_else(!is.na(ATOXGRN), paste0("Grade ", as.integer(ATOXGRN)), NA_character_) ) %>% select(-ULN)
# Worst on-treatment record flag per parameter (by highest toxicity grade, then highest value)
worst_gr <- adlb %>%
filter(ONTRTFL == "Y") %>%
group_by(USUBJID, PARAMCD) %>%
summarise(
WORSTGRN = suppressWarnings(max(ATOXGRN, na.rm = TRUE)),
.groups = "drop"
) %>% mutate(WORSTGRN = ifelse(is.finite(WORSTGRN), WORSTGRN, NA_real_))
adlb <- adlb %>% left_join(worst_gr, by = c("USUBJID", "PARAMCD")) %>%
group_by(USUBJID, PARAMCD) %>% mutate(
# In case of ties on grade, prefer the highest AVAL; if still tied, earliest ADTM
WORSTFL = case_when( ONTRTFL == "Y" & !is.na(WORSTGRN) & !is.na(ATOXGRN) & ATOXGRN == WORSTGRN & AVAL ==
max(AVAL[ONTRTFL == "Y" & ATOXGRN == WORSTGRN], na.rm = TRUE) &
ADTM == min(ADTM[ONTRTFL == "Y" & ATOXGRN == WORSTGRN &
AVAL == max(AVAL[ONTRTFL == "Y" & ATOXGRN == WORSTGRN], na.rm = TRUE)], na.rm = TRUE) ~ "Y", TRUE ~ "N") ) %>%
ungroup() %>% select(-WORSTGRN)
# Fsairst post-baseline HIGH flag per parameter
first_hi <- adlb %>% filter(POSTBLFL == "Y", ANRIND == "HIGH") %>% group_by(USUBJID, PARAMCD) %>% slice_min(ADTM,
with_ties = FALSE) %>% ungroup() %>% transmute(USUBJID, PARAMCD, FIRSTHIDTM = ADTM)
```

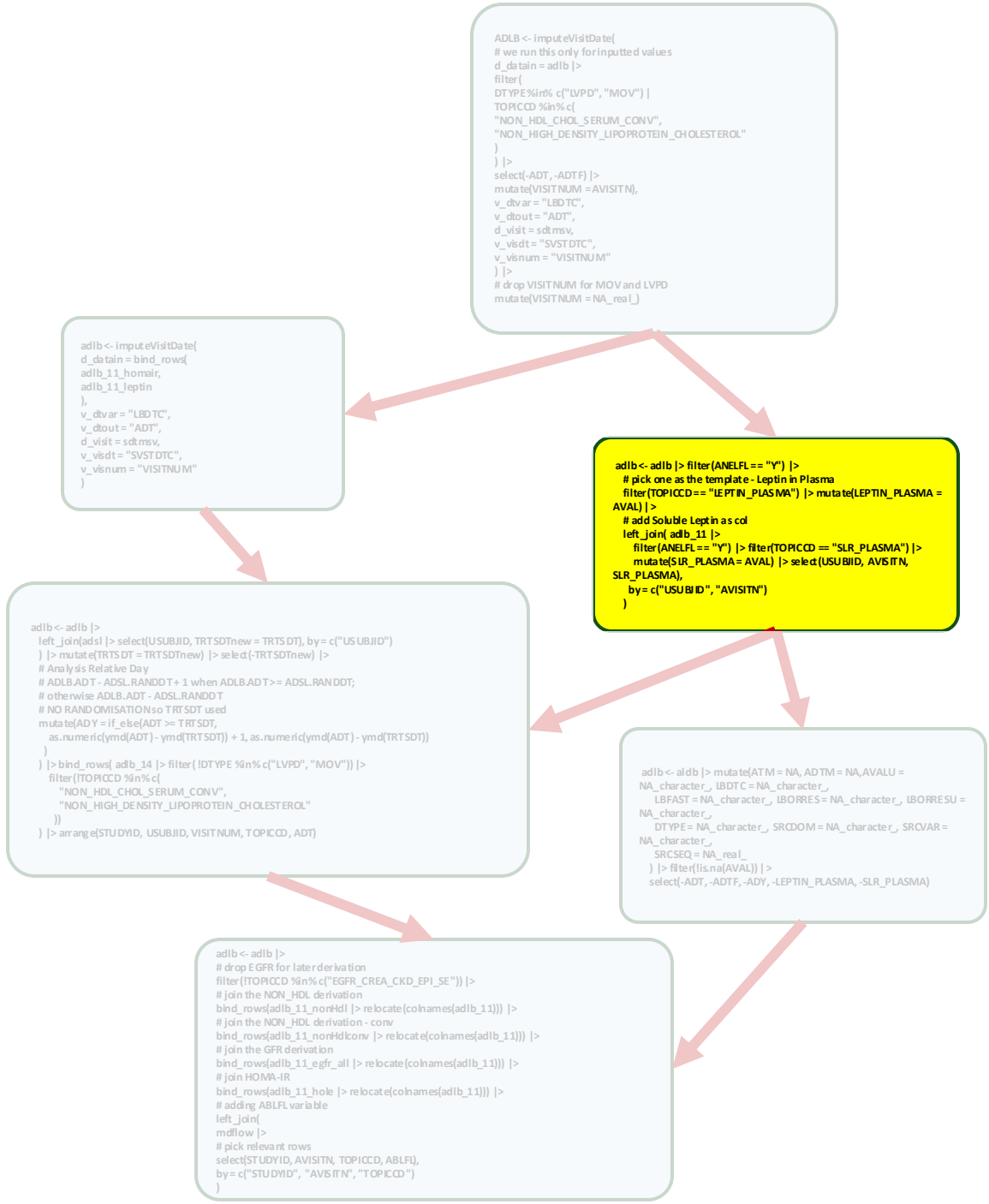

```
# Baseline flag and baseline value (last non-missing prior to first dose)
adlb <- adlb %>%
group_by(USUBJID, PARAMCD) %>%
arrange(USUBJID, PARAMCD, ADTM, .by_group = TRUE) %>%
mutate( ABLFL = if_else(!is.na(ADTM) & ADTM <= TRTSDTM & !is.na(AVAL) &
ADTM == max(ADTM[ADTM <= TRTSDTM & !is.na(AVAL)], na.rm = TRUE),
"Y", "N"),
# Handle case where no pre-dose records exist (max(..., na.rm=TRUE) gives -Inf)
ABLFL = if_else(is.infinite(max(ADTM[ADTM <= TRTSDTM & !is.na(AVAL)], na.rm = TRUE)), "N", ABLFL),
BASE = if_else(ABLFL == "Y", AVAL, NA_real_)) %>% fill(BASE, .direction = "downup") %>% ungroup()
# On-treatment and post-baseline flags
adlb <- adlb %>% mutate( POSTBLFL = if_else(!is.na(ADT) & !is.na(TRTSDT) & ADT >= TRTSDT, "Y", "N"), ONTRTFL = if_else(
!is.na(ADTM) & !is.na(TRTSDTM) & !is.na(TRTEDTM) & ADTM >= TRTSDTM & ADTM <= (TRTEDTM + days(1L)), "Y", "N" ))
# Change and percent change from baseline
adlb <- adlb %>% mutate( CHG = if_else(!is.na(AVAL) & !is.na(BASE), AVAL - BASE, NA_real_), PCHG = if_else(!is.na(AVAL) &
!is.na(BASE) & BASE != 0, 100 * (AVAL - BASE) / BASE, NA_real_))
# Normal range-based indicators and ratios (coalesce to SDTM STNR ranges if present)
adlb <- adlb %>% mutate( ANRLO = coalesce(ANRLO, LBSTNRLO), ANRHI = coalesce(ANRHI, LBSTNRHI), ANRIND = case_when(
is.na(AVAL) | (is.na(ANRLO) & is.na(ANRHI)) ~ NA_character_, !is.na(ANRLO) & AVAL < ANRLO ~ "LOW",
!is.na(ANRHI) & AVAL > ANRHI ~ "HIGH",
TRUE ~ "NORMAL" ), R2ANRHI = if_else(!is.na(ANRHI) & ANRHI > 0, AVAL / ANRHI, NA_real_), R2ANRLO = if_else(!is.na(ANRLO)
& ANRLO > 0, AVAL / ANRLO, NA_real_)
) %>% group_by(USUBJID, PARAMCD) %>% mutate( BNRIND = first(ANRIND[ABLFL == "Y"]), BASECAT = BNRIND
) %>% ungroup()
# Shift from baseline category (baseline to current)
adlb <- adlb %>% mutate( SHIFT1 = if_else( POSTBLFL == "Y",
paste0(coalesce(BASECAT, "MISSING"), " to ", coalesce(ANRIND, "MISSING")), NA_character_ ))
# CTCAE-like toxicity grades for selected labs using multiples of ULN (ANRHI)
# Note: These are simplified examples; consult CTCAE for exact, parameter-specific rules.
adlb <- adlb %>% mutate( ULN = ANRHI,
ATOXGRN = case_when(
PARAMCD %in% c("ALT", "AST") & !is.na(ULN) & !is.na(AVAL) ~ case_when(AVAL <= ULN ~ 0,
AVAL <= 3 * ULN ~ 1, AVAL <= 5 * ULN ~ 2, AVAL <= 20 * ULN ~ 3, AVAL > 20 * ULN ~ 4),
PARAMCD %in% c("ALP") & !is.na(ULN) & !is.na(AVAL) ~ case_when( AVAL <= ULN ~ 0, AVAL <= 2.5 * ULN ~ 1,
AVAL <= 5 * ULN ~ 2, AVAL <= 20 * ULN ~ 3, AVAL > 20 * ULN ~ 4),
PARAMCD %in% c("TBIL", "BILI") & !is.na(ULN) & !is.na(AVAL) ~ case_when(
AVAL <= ULN ~ 0, AVAL <= 1.5 * ULN ~ 1, AVAL <= 3 * ULN ~ 2,
AVAL <= 10 * ULN ~ 3, AVAL > 10 * ULN ~ 4 ), TRUE ~ NA_real_),
ATOXGR = if_else(!is.na(ATOXGRN), paste0("Grade ", as.integer(ATOXGRN)), NA_character_) ) %>% select(-ULN)
# Worst on-treatment record flag per parameter (by highest toxicity grade, then highest value)
worst_gr <- adlb %>%
filter(ONTRTFL == "Y") %>%
group_by(USUBJID, PARAMCD) %>%
summarise(
WORSTGRN = suppressWarnings(max(ATOXGRN, na.rm = TRUE)),
.groups = "drop"
) %>% mutate(WORSTGRN = ifelse(is.finite(WORSTGRN), WORSTGRN, NA_real_))
adlb <- adlb %>% left_join(worst_gr, by = c("USUBJID", "PARAMCD")) %>%
group_by(USUBJID, PARAMCD) %>% mutate(
# In case of ties on grade, prefer the highest AVAL; if still tied, earliest ADTM
WORSTFL = case_when( ONTRTFL == "Y" & !is.na(WORSTGRN) & !is.na(ATOXGRN) & ATOXGRN == WORSTGRN & AVAL ==
max(AVAL[ONTRTFL == "Y" & ATOXGRN == WORSTGRN], na.rm = TRUE) &
ADTM == min(ADTM[ONTRTFL == "Y" & ATOXGRN == WORSTGRN &
AVAL == max(AVAL[ONTRTFL == "Y" & ATOXGRN == WORSTGRN], na.rm = TRUE)], na.rm = TRUE) ~ "Y", TRUE ~ "N") ) %>%
ungroup() %>% select(-WORSTGRN)
# Fsairst post-baseline HIGH flag per parameter
first_hi <- adlb %>% filter(POSTBLFL == "Y", ANRIND == "HIGH") %>% group_by(USUBJID, PARAMCD) %>% slice_min(ADTM,
with_ties = FALSE) %>% ungroup() %>% transmute(USUBJID, PARAMCD, FIRSTHIDTM = ADTM)
```

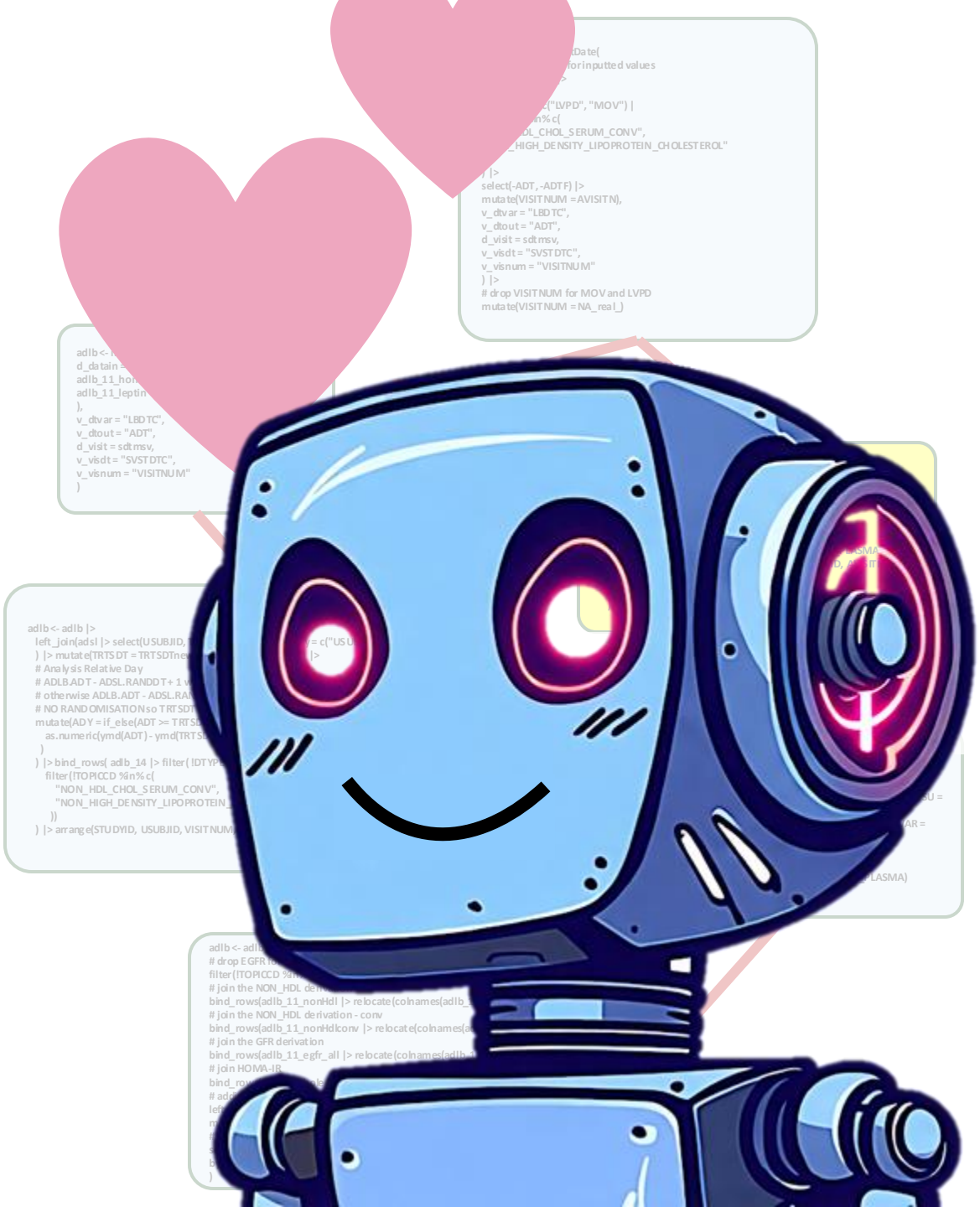
```
# Baseline flag and baseline value (last non-missing prior to first dose)
adlb <- adlb %>%
group_by(USUBJID, PARAMCD) %>%
arrange(USUBJID, PARAMCD, ADTM, .by_group = TRUE) %>%
mutate( ABLFL = if_else(!is.na(ADTM) & ADTM <= TRTSDTM & !is.na(AVAL) &
ADTM == max(ADTM[ADTM <= TRTSDTM & !is.na(AVAL)], na.rm = TRUE),
"Y", "N"),
# Handle case where no pre-dose records exist (max(..., na.rm=TRUE) gives -Inf)
ABLFL = if_else(is.infinite(max(ADTM[ADTM <= TRTSDTM & !is.na(AVAL)], na.rm = TRUE)), "N", ABLFL),
BASE = if_else(ABLFL == "Y", AVAL, NA_real_)) %>% fill(BASE, .direction = "downup") %>% ungroup()
# On-treatment and post-baseline flags
adlb <- adlb %>% mutate( POSTBLFL = if_else(!is.na(ADT) & !is.na(TRTSDT) & ADT >= TRTSDT, "Y", "N"), ONTRTFL = if_else(
!is.na(ADTM) & !is.na(TRTSDTM) & !is.na(TRTEDTM) & ADTM >= TRTSDTM & ADTM <= (TRTEDTM + days(1L)), "Y", "N" ))
# Change and percent change from baseline
adlb <- adlb %>% mutate( CHG = if_else(!is.na(AVAL) & !is.na(BASE), AVAL - BASE, NA_real_), PCHG = if_else(!is.na(AVAL) &
!is.na(BASE) & BASE != 0, 100 * (AVAL - BASE) / BASE, NA_real_))
# Normal range-based indicators and ratios (coalesce to SDTM STNR ranges if present)
adlb <- adlb %>% mutate( ANRLO = coalesce(ANRLO, LBSTNRLO), ANRHI = coalesce(ANRHI, LBSTNRHI), ANRIND = case_when(
is.na(AVAL) | (is.na(ANRLO) & is.na(ANRHI)) ~ NA_character_, !is.na(ANRLO) & AVAL < ANRLO ~ "LOW",
!is.na(ANRHI) & AVAL > ANRHI ~ "HIGH",
TRUE ~ "NORMAL" ), R2ANRHI = if_else(!is.na(ANRHI) & ANRHI > 0, AVAL / ANRHI, NA_real_), R2ANRLO = if_else(!is.na(ANRLO)
& ANRLO > 0, AVAL / ANRLO, NA_real_)
) %>% group_by(USUBJID, PARAMCD) %>% mutate( BNRIND = first(ANRIND[ABLFL == "Y"]), BASECAT = BNRIND
) %>% ungroup()
# Shift from baseline category (baseline to current)
adlb <- adlb %>% mutate( SHIFT1 = if_else( POSTBLFL == "Y",
paste0(coalesce(BASECAT, "MISSING"), " to ", coalesce(ANRIND, "MISSING")), NA_character_ ))
# CTCAE-like toxicity grades for selected labs using multiples of ULN (ANRHI)
# Note: These are simplified examples; consult CTCAE for exact, parameter-specific rules.
adlb <- adlb %>% mutate( ULN = ANRHI,
ATOXGRN = case_when(
PARAMCD %in% c("ALT", "AST") & !is.na(ULN) & !is.na(AVAL) ~ case_when(AVAL <= ULN ~ 0,
AVAL <= 3 * ULN ~ 1, AVAL <= 5 * ULN ~ 2, AVAL <= 20 * ULN ~ 3, AVAL > 20 * ULN ~ 4),
PARAMCD %in% c("ALP") & !is.na(ULN) & !is.na(AVAL) ~ case_when( AVAL <= ULN ~ 0, AVAL <= 2.5 * ULN ~ 1,
AVAL <= 5 * ULN ~ 2, AVAL <= 20 * ULN ~ 3, AVAL > 20 * ULN ~ 4),
PARAMCD %in% c("TBIL", "BILI") & !is.na(ULN) & !is.na(AVAL) ~ case_when(
AVAL <= ULN ~ 0, AVAL <= 1.5 * ULN ~ 1, AVAL <= 3 * ULN ~ 2,
AVAL <= 10 * ULN ~ 3, AVAL > 10 * ULN ~ 4 ), TRUE ~ NA_real_),
ATOXGR = if_else(!is.na(ATOXGRN), paste0("Grade ", as.integer(ATOXGRN)), NA_character_) ) %>% select(-ULN)
# Worst on-treatment record flag per parameter (by highest toxicity grade, then highest value)
worst_gr <- adlb %>%
filter(ONTRTFL == "Y") %>%
group_by(USUBJID, PARAMCD) %>%
summarise(
WORSTGRN = suppressWarnings(max(ATOXGRN, na.rm = TRUE)),
.groups = "drop"
) %>% mutate(WORSTGRN = ifelse(is.finite(WORSTGRN), WORSTGRN, NA_real_))
adlb <- adlb %>% left_join(worst_gr, by = c("USUBJID", "PARAMCD")) %>%
group_by(USUBJID, PARAMCD) %>% mutate(
# In case of ties on grade, prefer the highest AVAL; if still tied, earliest ADTM
WORSTFL = case_when( ONTRTFL == "Y" & !is.na(WORSTGRN) & !is.na(ATOXGRN) & ATOXGRN == WORSTGRN & AVAL ==
max(AVAL[ONTRTFL == "Y" & ATOXGRN == WORSTGRN], na.rm = TRUE) &
ADTM == min(ADTM[ONTRTFL == "Y" & ATOXGRN == WORSTGRN &
AVAL == max(AVAL[ONTRTFL == "Y" & ATOXGRN == WORSTGRN], na.rm = TRUE)], na.rm = TRUE) ~ "Y", TRUE ~ "N") ) %>%
ungroup() %>% select(-WORSTGRN)
# Fsairst post-baseline HIGH flag per parameter
first_hi <- adlb %>% filter(POSTBLFL == "Y", ANRIND == "HIGH") %>% group_by(USUBJID, PARAMCD) %>% slice_min(ADTM,
with_ties = FALSE) %>% ungroup() %>% transmute(USUBJID, PARAMCD, FIRSTHIDTM = ADTM)
```

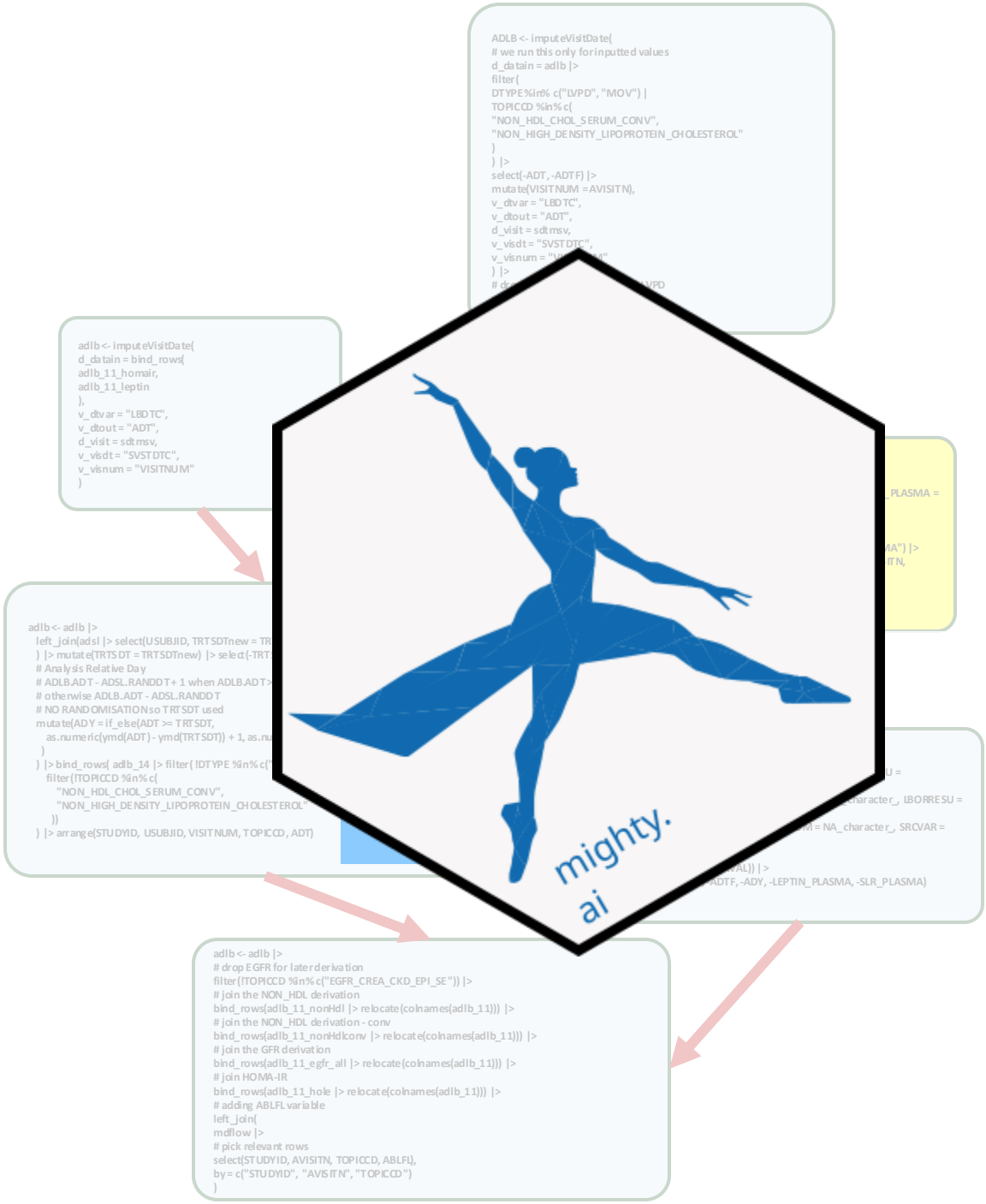


```
# Baseline flag and ba      e value (last non-missing prior to first dose)
adlb <- adlb %>%
group_by(USUBJID, PARAMCD) %>%
      e(USUBJID, PARAMCD, ADTM, .by_group = TRUE) %>%
      ABLFL = if_else(!is.na(ADTM) & ADTM <= TRTSDTM & !is.na(AVAL) &
ADTM == max(ADTM[ADTM <= TRTSDTM & !is.na(AVAL)], na.rm = TRUE),
"Y", "N"),
# Handle case where no pre-dose records exist (max(..., na.rm=TRUE) gives -Inf)
ABLFL = if_else(is.infinite(max(ADTM[ADTM <= TRTSDTM & !is.na(AVAL)], na.rm = TRUE)), "N", ABLFL),
BASE = if_else(ABLFL == "Y", AVAL, NA_real_)) %>% fill(BASE, .direction = "downup") %>% ungroup()
# On-treatment and post-baseline flags
adlb <- adlb %>% mutate( POSTBLFL = if_else(!is.na(ADT) & !is.na(TRTSDT) & ADT >= TRTSDT, "Y", "N"), ONTRTFL = if_else(
!is.na(ADTM) & !is.na(TRTSDTM) & !is.na(TRTEDTM) & ADTM >= TRTSDTM & ADTM <= (TRTEDTM + days(1L)), "Y", "N" ))
# Change and p      change from baseline
adlb <- adlb %>%      CHG = if_else(!is.na(AVAL) & !is.na(BASE), AVAL - BASE, NA_real_), PCHG = if_else(!is.na(AVAL) &
!is.na(BASE) &      100 * (AVAL - BASE) / BASE, NA_real_))
# Normal ran      ndicators and ratios (coalesce to SDTM STNR ranges if present)
adlb <- adlb %>%      ate( ANRLO = coalesce(ANRLO, LBSTNRLO), ANRHI = coalesce(ANRHI, LBSTNRHI), ANRIND = case_when(
      na(ANRLO) & is.na(ANRHI)) ~ NA_character_, !is.na(ANRLO) & AVAL < ANRLO ~ "LOW",
      ANRHI ~ "HIGH",
      RHI = if_else(!is.na(ANRHI) & ANRHI > 0, AVAL / ANRHI, NA_real_), R2ANRLO = if_else(!is.na(ANRLO)
      A_real_)
      ) %>% mutate( BNRIND = first(ANRIND[ABLFL == "Y"]), BASECAT = BNRIND
      t)
      FL == "Y",
      (ANRIND), "MISSING")), NA_character_ ))
      ected l      co      exact, parameter-specific rules.
      l,
      ) & !      ) ~ case_when(AVAL <= ULN ~ 0,
      5 *      ~ 3, AVAL > 20 * ULN ~ 4),
      when( AVAL <= ULN ~ 0, AVAL <= 2.5 * ULN ~ 1,
      4),
      ~ case_when(
      ),
      r(ATOXGRN)), NA_character_) ) %>% select(-ULN)
      oxicity grade, then highest value)
      TRUE)),
      ), WORSTGRN, NA_real_))
      , "PARAMCD")) %>%
      L; if still tied, earliest ADTM
      na(WORSTGRN) & !is.na(ATOXGRN) & ATOXGRN == WORSTGRN & AVAL ==
      ORSTGRN], na.rm = TRUE) &
      XGRN == WORSTGRN &
      OXGRN == WORSTGRN], na.rm = TRUE)], na.rm = TRUE) ~ "Y", TRUE ~ "N") ) %>%
      er parameter
      STBLFL == "Y", ANRIND == "HIGH") %>% group_by(USUBJID, PARAMCD) %>% slice_min(ADTM,
      ungroup() %>% transmute(USUBJID, PARAMCD, FIRSTHIDTM = ADTM)
```


ADLB <- imputeVisitDate(
# we run this only for inputted values
d_datain = adlb |>
filter(
DTYPE %in% c("LVPD", "MOV") |
TOPICCD %in% c(
"NON_HDL_CHOL_SERUM_CONV",
"NON_HIGH_DENSITY_LIPOPROTEIN_CHOLESTEROL"
)
) |>
select(-ADT, -ADTF) |>
mutate(VISITNUM = AVISITN),
v_dtvar = "LBDTC",
v_dtout = "ADT",
d_visit = sdtmsv,
v_visdt = "SVSTDTC",
v_visnum = "VISITNUM"
) |>
# drop VISITNUM for MOV and LVPD
mutate(VISITNUM = NA_real_)
adlb <- imputeVisitDate(
d_datain = bind_rows(
adlb_11_homair,
adlb_11_leptin
),
v_dtvar = "LBDTC",
v_dtout = "ADT",
d_visit = sdtmsv,
v_visdt = "SVSTDTC",
v_visnum = "VISITNUM"
)
adlb <- adlb |> filter(ANELFL == "Y") |>
# pick one as the template - Leptin in Plasma
filter(TOPICCD == "LEPTIN_PLASMA") |> mutate(LEPTIN_PLASMA = AVAL) | >
add Soluble Leptin as col
left_join(adlb_11 |>
filter(ANELFL == "Y") |> filter(TOPICCD == "SLR_PLASMA") |>
mutate(SLR_PLASMA = AVAL) |> select(USUBJID, AVISITN, SLR_PLASMA),
by = c("USUBJID", "AVISITN")
)
adlb <- adlb |>
left_join(adsl |> select(USUBJID, TRTSDTnew = TRTSDT), by = c("USUBJID")
) |> mutate(TRTSDT = TRTSDTnew) |> select(-TRTSDTnew) |>
Analysis Relative Day
ADLB.ADT - ADSL.RANDDT + 1 when ADLB.ADT >= ADSL.RANDDT;
otherwise ADLB.ADT - ADSL.RANDDT
NO RANDOMISATION so TRTSDT used
mutate(ADY = if_else(ADT >= TRTSDT,
as.numeric(ymd(ADT) - ymd(TRTSDT)) + 1, as.numeric(ymd(ADT) - ymd(TRTSDT))
)
) |> bind_rows(adlb_14 |> filter(!DTYPE %in% c("LVPD", "MOV")) |>
filter(!TOPICCD %in% c(
"NON_HDL_CHOL_SERUM_CONV",
"NON_HIGH_DENSITY_LIPOPROTEIN_CHOLESTEROL"
))
) |> arrange(STUDYID, USUBJID, VISITNUM, TOPICCD, ADT)
adlb <- aldb |> mutate(ATM = NA, ADTM = NA, AVALU = NA_character_, LBDTC = NA_character_,
LBFAST = NA_character_, LBORRES = NA_character_, LBORRESU = NA_character_,
DTYPE = NA_character_, SRCDOM = NA_character_, SRCVAR = NA_character_,
SRCSEQ = NA_real_
) |> filter(!is.na(AVAL)) | >
select(-ADT, -ADTF, -ADY, -LEPTIN_PLASMA, -SLR_PLASMA)
adlb <- adlb |>
# drop EGFR for later derivation
filter(!TOPICCD %in% c("EGFR_CREA_CKD_EPI_SE")) |>
join the NON_HDL derivation
bind_rows(adlb_11_nonHdl |> relocate(colnames(adlb_11))) |>
join the NON_HDL derivation - conv
bind_rows(adlb_11_nonHdlconv |> relocate(colnames(adlb_11))) |>
join the GFR derivation
bind_rows(adlb_11_egfr_all |> relocate(colnames(adlb_11))) |>
join HOMA-IR
bind_rows(adlb_11_hole |> relocate(colnames(adlb_11))) |>
adding ABLFL variable
left_join(
mdflow |>
pick relevant rows
select(STUDYID, AVISITN, TOPICCD, ABLFL),
by = c("STUDYID", "AVISITN", "TOPICCD")
)


ADLB <- imputeVisitDate(
# we run this only for inputted values
d_datain = adlb |>
filter(
DTYPE %in% c("LVPD", "MOV") |
TOPICCD %in% c(
"NON_HDL_CHOL_SERUM_CONV",
"NON_HIGH_DENSITY_LIPOPROTEIN_CHOLESTEROL"
)
) |>
select(-ADT, -ADTF) |>
mutate(VISITNUM = AVISITN),
v_dtvar = "LBDTC",
v_dtout = "ADT",
d_visit = sdtmsv,
v_visdt = "SVSTDTC",
v_visnum = "VISITNUM"
) |>
# drop VISITNUM for MOV and LVPD
mutate(VISITNUM = NA_real_)
adlb <- imputeVisitDate(
d_datain = bind_rows(
adlb_11_homair,
adlb_11_leptin
),
v_dtvar = "LBDTC",
v_dtout = "ADT",
d_visit = sdtmsv,
v_visdt = "SVSTDTC",
v_visnum = "VISITNUM"
)
adlb <- adlb |> filter(ANELFL == "Y") |>
# pick one as the template - Leptin in Plasma
filter(TOPICCD == "LEPTIN_PLASMA") |> mutate(LEPTIN_PLASMA = AVAL) | >
add Soluble Leptin as col
left_join(adlb_11 |>
filter(ANELFL == "Y") |> filter(TOPICCD == "SLR_PLASMA") |>
mutate(SLR_PLASMA = AVAL) |> select (USUBJID, AVISITN, SLR_PLASMA),
by = c("USUBJID", "AVISITN")
)
adlb <- adlb |>
left_join(adsl |> select(USUBJID, TRTSDTnew = TRTSDT), by = c("USUBJID")
) |> mutate(TRTSDT = TRTSDTnew) |> select(-TRTSDTnew) |>
Analysis Relative Day
ADLB.ADT - ADSL.RANDDT + 1 when ADLB.ADT >= ADSL.RANDDT;
otherwise ADLB.ADT - ADSL.RANDDT
NO RANDOMISATION so TRTSDT used
mutate(ADY = if_else(ADT >= TRTSDT,
as.numeric(ymd(ADT) - ymd(TRTSDT)) + 1, as.numeric(ymd(ADT) - ymd(TRTSDT))
)
) |> bind_rows(adlb_14 |> filter(!DTYPE %in% c("LVPD", "MOV")) |>
filter(!TOPICCD %in% c(
"NON_HDL_CHOL_SERUM_CONV",
"NON_HIGH_DENSITY_LIPOPROTEIN_CHOLESTEROL"
))
) |> arrange(STUDYID, USUBJID, VISITNUM, TOPICCD, ADT)
adlb <- aldb |> mutate(ATM = NA, ADTM = NA,AVALU = NA_character_, LBDTC = NA_character_,
LBFAST = NA_character_, LBORRES = NA_character_, LBORRESU = NA_character_,
DTYPE = NA_character_, SRCDOM = NA_character_, SRCVAR = NA_character_,
SRCSEQ = NA_real_
) |> filter(!is.na(AVAL)) | >
select(-ADT, -ADTF, -ADY, -LEPTIN_PLASMA, -SLR_PLASMA)
adlb <- adlb |>
# drop EGFR for later derivation
filter(!TOPICCD %in% c("EGFR_CREA_CKD_EPI_SE")) |>
join the NON_HDL derivation
bind_rows(adlb_11_nonHdl |> relocate(colnames(adlb_11))) |>
join the NON_HDL derivation - conv
bind_rows(adlb_11_nonHdlconv |> relocate(colnames(adlb_11))) |>
join the GFR derivation
bind_rows(adlb_11_egfr_all |> relocate(colnames(adlb_11))) |>
join HOMA-IR
bind_rows(adlb_11_hole |> relocate(colnames(adlb_11))) |>
adding ABLFL variable
left_join(
mdflow |>
pick relevant rows
select(STUDYID, AVISITN, TOPICCD, ABLFL),
by = c("STUDYID", "AVISITN", "TOPICCD")
)




ADLB <- imputeVisitDate(
# we run this only for inputted values
d_datain = adlb |>
filter(
DTYPE %in% c("LVPD", "MOV") |
TOPICCD %in% c(
"NON_HDL_CHOL_SERUM_CONV",
"NON_HIGH_DENSITY_LIPOPROTEIN_CHOLESTEROL"
)
) |>
select(-ADT, -ADTF) |>
mutate(VISITNUM = AVISITN),
v_dtvar = "LBDTC",
v_dtout = "ADT",
d_visit = sdtmsv,
v_visdt = "SVSTDTC",
adlb <- imputeVisitDate(
d_datain = bind_rows(
adlb_11_homair,
adlb_11_leptin
),
v_dtvar = "LBDTC",
v_dtout = "ADT",
d_visit = sdtmsv,
v_visdt = "SVSTDTC",
v_visnum = "VISITNUM"
)
adlb <- adlb |>
left_join(adsl |> select(USUBJID, TRTSDTnew = TR
) |> mutate(TRTSDT = TRTSDTnew) |> select(-TRT
Analysis Relative Day
ADLB.ADT - ADSL.RANDDT + 1 when ADLB.ADT >
otherwise ADLB.ADT - ADSL.RANDDT
NO RANDOMISATION so TRTSDT used
mutate(ADY = if_else(ADT >= TRTSDT,
as.numeric(ymd(ADT) - ymd(TRTSDT)) + 1, as.nu
)
) |> bind_rows(adlb_14 |> filter(!DTYPE %in% c("
filter(!TOPICCD %in% c(
"NON_HDL_CHOL_SERUM_CONV",
"NON_HIGH_DENSITY_LIPOPROTEIN_CHOLESTEROL"
))
) |> arrange(STUDYID, USUBJID, VISITNUM, TOPICCD, ADT)
_PLASMA =
MA") |>
ITN,
U =
character_, LBORRESU =
M = NA_character_, SRCVAR =
VAL)) | >
ADTF, -ADY, -LEPTIN_PLASMA, -SLR_PLASMA)
adlb <- adlb |>
# drop EGFR for later derivation
filter(!TOPICCD %in% c("EGFR_CREA_CKD_EPI_SE")) |>
join the NON_HDL derivation
bind_rows(adlb_11_nonHdl |> relocate(colnames(adlb_11))) |>
join the NON_HDL derivation - conv
bind_rows(adlb_11_nonHdlconv |> relocate(colnames(adlb_11))) |>
join the GFR derivation
bind_rows(adlb_11_egfr_all |> relocate(colnames(adlb_11))) |>
join HOMA-IR
bind_rows(adlb_11_hole |> relocate(colnames(adlb_11))) |>
adding ABLFL variable
left_join(
mdflow |>
pick relevant rows
select(STUDYID, AVISITN, TOPICCD, ABLFL),
by = c("STUDYID", "AVISITN", "TOPICCD")
)
mighty.
ai

5 May, 2012,
https://www.reddit.com/r/funny/comments/t8gjh/instructions_for_my_new_undies_just_in_case/

@davie_dave 24 April 2020, https://www.instagram.com/p/B_YHfJiJ_9j/

Matthew Phelps

mewp@novonordisk.com

Nocialai Skov Jensen