Work with Tables AP

Table of contents

1. Motions

```r
library(openxlsx)
motions <- readxl::read_xlsx("c:/Users/Eduard/Desktop/DL_CH_Smolensk/Assay_Param/Tests/
Assay_Parameters_AS.xlsx", sheet = "motions")
```

```r
library(dplyr)
library(tidyr)
library(stringr)


reads<- readxl::read_xlsx("c:/Users/Eduard/Desktop/DL_CH_Smolensk/Assay_Param/Tests/
Assay_Parameters_AS.xlsx", sheet = "motions")

pivot_motions_by_test <- function(motions, test_order = NULL, step_col = NULL,
show_step = TRUE) {
  stopifnot(all(c("shortname","motiontype","targettag") %in% names(motions)))

  metric_order <- c("chase_volume","air_volume","reagent_volume",
                    "sample_volume","overdraw_volume","volume")

  df <- motions %>%
    mutate(across(everything(), as.character)) %>%
    # нормализуем единицы
    mutate(across(ends_with("unit"),
                  ~ str_replace_all(str_trim(tolower(.x)), "µl|ul", "ul"))) %>%
    # --- сохраняем реальный порядок шагов ---
    group_by(assay_id, shortname) %>%
    mutate(
      .step = if (!is.null(step_col) && step_col %in% names(.))
        as.integer(.data[[step_col]])
      else dplyr::row_number()
    ) %>%
    arrange(.step, .by_group = TRUE) %>%
    # --- «несём вниз» источник после ASPIRATE ---
    mutate(src_after_last_asp = if_else(motiontype == "ASPIRATE", targettag,
NA_character_)) %>%
    tidyr::fill(src_after_last_asp, .direction = "down") %>%
    ungroup() %>%
    # для DISPENSE подменяем targettag на источник последнего ASPIRATE
    mutate(targettag_eff = if_else(motiontype == "DISPENSE", src_after_last_asp,
targettag)) %>%
    select(-src_after_last_asp)

  # чтобы не конфликтовало с именами
  df <- df %>% rename(total_volume = volume, total_unit = volume_unit)

  long <- df %>%
    pivot_longer(
      cols = matches("^(chase|air|reagent|sample|overdraw|total)_(volume|unit)$"),
      names_to = c("metric","kind"),
      names_pattern = "(chase|air|reagent|sample|overdraw|total)_(volume|unit)",
      values_to = "val"
    ) %>%
    pivot_wider(
      names_from  = kind,
      values_from = val,
      values_fill = "",
      values_fn   = ~ paste(na.omit(unique(.x)), collapse = " / ")
    ) %>%
    mutate(value_fmt = dplyr::case_when(
      volume == "" ~ NA_character_,
      unit   == "" ~ volume,
      TRUE         ~ paste0(volume, " ", unit)
    ))

  # свод по событию шага (.step) — чтобы ничего не «склеилось» между шагами
  event <- long %>%
    transmute(
      assay_id, shortname, .step,
      motiontype, targettag = targettag_eff,
      metric, value_fmt
    ) %>%
    group_by(assay_id, shortname, .step, motiontype, targettag, metric) %>%
    summarise(value = paste(na.omit(unique(value_fmt)), collapse = " / "), .groups =
"drop") %>%
    mutate(
      metric = recode(metric,
        chase="chase_volume", air="air_volume", reagent="reagent_volume",
        sample="sample_volume", overdraw="overdraw_volume", total="volume"
      ),
      metric = factor(metric, levels = metric_order, ordered = TRUE)
    )

  out <- event %>%
    arrange(assay_id, shortname, .step, metric) %>%
    group_by(assay_id, shortname) %>%
    mutate(step = dplyr::dense_rank(.step)) %>%
    ungroup() %>%
    select(step, motiontype, targettag, metric, shortname, value) %>%
    pivot_wider(names_from = shortname, values_from = value, values_fill = "") %>%
    # порядок колонок-тестов
    {
      fixed <- c("step","motiontype","targettag","metric")
      tests <- setdiff(names(.), fixed)
      tests <- if (is.null(test_order)) sort(tests) else intersect(test_order, tests)
      .[, c(fixed, tests), drop = FALSE]
    } %>%
    {
      if (!show_step) select(., -step) else .
    }
  out
}


motions_by_test <- pivot_motions_by_test(motions)
```

```r
openxlsx::write.xlsx(motions_by_test, "c:/Users/Eduard/Desktop/DL_CH_Smolensk/
Assay_Param/Tests/motions_by_test_R.xlsx")
```

2. Reads

```r
library(dplyr)
library(tidyr)
library(stringr)

# reads: data.frame/tibble с колонками assay_id, shortname, readtag
pivot_reads_by_test <- function(reads, test_order = NULL, mark = "✓") {
  stopifnot(all(c("shortname","readtag") %in% names(reads)))

  # нормализуем и оставляем уникальные пары тест–тег
  df <- reads %>%
    transmute(
      shortname = as.character(shortname),
      readtag   = as.character(readtag)
    ) %>%
    distinct()

  # разметим порядок строк: сначала обычные P-теги по номеру, потом C…P и прочее
  df <- df %>%
    mutate(
      grp = case_when(
        str_detect(readtag, "^P\\d+")      ~ "P",
        str_detect(readtag, "^C\\d+P\\d+") ~ "C",
        TRUE                               ~ "Z"
      ),
      num = suppressWarnings(as.numeric(str_extract(readtag, "\\d+")))
    )

  # что ставить в ячейку (галочка/сам тег/индекс шага)
  df$val <- mark

  # сводная таблица: строки — readtag, столбцы — shortname
  wide <- df %>%
    pivot_wider(
      id_cols     = c(grp, num, readtag),
      names_from = shortname,
      values_from= val,
      values_fill = ""
    ) %>%
    arrange(grp, num, readtag) %>%
    select(-grp, -num)

  # порядок столбцов-тестов
  fixed <- "readtag"
  tests <- setdiff(names(wide), fixed)
  if (is.null(test_order)) {
    tests <- sort(tests)
  } else {
    tests <- intersect(test_order, tests)
  }
  wide[, c(fixed, tests), drop = FALSE]
}

reads_wide <- pivot_reads_by_test(reads)              # «галочки» по наличию тега
# или так, если хотите свой порядок тестов:
# reads_wide <- pivot_reads_by_test(reads, test_order = c("6AM","A1c_E","A1c_H", ...))
```

```
openxlsx::write.xlsx(reads_wide, "c:/Users/Eduard/Desktop/DL_CH_Smolensk/Assay_Param/
Tests/reads_by_test_R.xlsx")
```

3. delta_rules

```
delta_rules<-readxl::read_xlsx("c:/Users/Eduard/Desktop/DL_CH_Smolensk/Assay_Param/
Tests/Assay_Parameters_AS.xlsx", sheet = "delta_rules")

library(dplyr)
library(tidyr)
library(stringr)

# delta_rules: assay_id, shortname, calilevel, deltalow, deltahigh
pivot_delta_rules_by_test <- function(delta_rules, test_order = NULL,
                                      as_range = TRUE, digits = 3) {
  stopifnot(all(c("shortname","calilevel","deltalow","deltahigh") %in%
names(delta_rules)))

  df <- delta_rules %>%
    transmute(
      shortname = as.character(shortname),
      calilevel = suppressWarnings(as.numeric(calilevel)),
      deltalow  = suppressWarnings(as.numeric(deltalow)),
      deltahigh = suppressWarnings(as.numeric(deltahigh))
    )

  if (as_range) {
    # одна строка с диапазоном "low–high", плюс строка с calilevel
    rng <- df %>%
      mutate(value = ifelse(is.na(deltalow) & is.na(deltahigh), NA_character_,
                      paste0(formatC(deltalow, digits = digits, format = "fg",
flag = "#"),
                             " – ",
                             formatC(deltahigh, digits = digits, format = "fg",
flag = "#")))) %>%
      select(shortname, metric = "delta_range", value)

    cli <- df %>%
      mutate(value = as.character(calilevel)) %>%
      select(shortname, metric = "calilevel", value)

    long <- bind_rows(cli, rng) %>% distinct()
  } else {
    # три отдельные строки: calilevel, deltalow, deltahigh
    long <- df %>%
```

```r
      pivot_longer(calilevel:deltahigh, names_to = "metric", values_to = "value") %>%
      mutate(value = ifelse(is.na(value), "", as.character(value)))
  }

  metric_order <- c("calilevel", "deltalow", "deltahigh", "delta_range")

  wide <- long %>%
    mutate(metric = factor(metric, levels = metric_order, ordered = TRUE)) %>%
    arrange(metric) %>%
    pivot_wider(id_cols = metric,
                names_from = shortname,
                values_from = value,
                values_fill = "") %>%
    {
      fixed <- "metric"
      tests <- setdiff(names(.), fixed)
      if (!is.null(test_order)) tests <- intersect(test_order, tests) else tests <-
sort(tests)
      .[, c(fixed, tests), drop = FALSE]
    }

  wide
}


# 1) Диапазон в одной строке + calilevel
delta_wide <- pivot_delta_rules_by_test(delta_rules, as_range = TRUE)

# 2) Отдельные строки deltalow/deltahigh
delta_wide2 <- pivot_delta_rules_by_test(delta_rules, as_range = FALSE)

# 3) Свой порядок тестов
# delta_wide <- pivot_delta_rules_by_test(delta_rules, test_order =
c("6AM","A1c_E","A1c_H", ...))
```

```r
openxlsx::write.xlsx(delta_wide2, "c:/Users/Eduard/Desktop/DL_CH_Smolensk/Assay_Param/
Tests/delta_rules_by_tests_R.xlsx")
```

4. calculation

```r
library(dplyr)
library(stringr)

# 1) компактное представление индексов: 1,2,3,5,6,7 -> "1–3,5–7"
.compress_indices <- function(idx) {
  idx <- sort(unique(as.integer(idx[!is.na(idx)])))
  if (length(idx) == 0) return("")
  runs <- cumsum(c(1, diff(idx) != 1))
  grp  <- split(idx, runs)
  paste(vapply(grp, function(v)
    if (length(v) == 1) as.character(v) else paste0(v[1], "–", v[length(v)]),
    character(1)),
    collapse = ","
  )
}
```

```r
# 2) схлопываем любые теги, заканчивающиеся цифрой, в STEM[range]
collapse_numeric_tags <- function(rules, tag_col = "tag",
                                  by = c("assay_id","shortname")) {
  stopifnot(tag_col %in% names(rules))
  x <- rules %>% mutate(..tag = .data[[tag_col]])

  m <- str_match(x$..tag, "^(.*?)(\\d+)$")  # stem + numeric tail
  x$..stem <- ifelse(is.na(m[,3]), x$..tag, m[,2])
  x$..idx  <- suppressWarnings(as.integer(m[,3]))

  # метка по группе (assay_id, shortname, stem)
  labels <- x %>%
    group_by(across(all_of(by)), ..stem) %>%
    summarise(
      ..label = if (all(is.na(..idx))) ..stem
                else paste0(..stem, "[", .compress_indices(..idx), "]"),
      .groups = "drop"
    )

  x %>%
    left_join(labels, by = c(by, "..stem")) %>%
    mutate(!!tag_col := ..label) %>%
    select(-..tag, -..stem, -..idx, -..label)
}

# === ПРИМЕНЕНИЕ ===
# уберём NA/пустые теги, схлопнем серии и развернём «как раньше»
rules_compact <- rules %>%
  filter(!is.na(tag), nzchar(tag)) %>%
  collapse_numeric_tags()

rules_wide <- pivot_rules_by_test(
  rules_compact,
  # при желании склеим несколько атрибутов в одну ячейку:
  combine_attrs   = c("formula","condition"),
  label_attrs     = TRUE,         # подписи "formula: ... | condition: ..."
  keep_attr       = FALSE,        # не показывать колонку с перечнем атрибутов
  drop_empty_rows = TRUE,
  show_assay      = TRUE
)
```

```r
calculation<-readxl::read_xlsx("c:/Users/Eduard/Desktop/DL_CH_Smolensk/Assay_Param/
Tests/Assay_Parameters_AS.xlsx", sheet = "calculation")

unique(calculation$tag)

rules_wide <- pivot_rules_by_test(calculation,
                                  test_order =
c("6AM","A1c_E","A1c_H","AAG","AAT","ACE","Acet"),
                                  drop_empty = TRUE,
                                  show_assay = TRUE)
```