Where most indicator bugs are born

group_by() and summarise() — the grouping that survives the call, the missing key that becomes its own group, and the three defaults that are the exact opposite of pandas'.

Pierre Robentz Cassion

Lesson 6 of 8

R and the Tidyverse for Programme Data — The verbs

What this lesson covers

- An indicator is two numbers
- `n()` and the two ways to count
- The grouping survives, and it is the commonest bug
- A missing key becomes its own group
- Unused factor levels: `count()` drops, `table()` keeps
- Grouped `mutate()`: a group statistic on every row
- `count()` for when that is all you want
- Two keys, and the table a report wants
- Writing the result out
- The three reversals, in one place
- What comes next

An indicator is two numbers — In R

```r
library(dplyr)

by_commune <- muac |>
  mutate(
    assessed = !is.na(muac_mm) | !is.na(oedema),
    gam = assessed & (muac_mm < 125 | oedema == "true")
  ) |>
  group_by(commune) |>
  summarise(
    screened  = n(),
    assessed  = sum(assessed, na.rm = TRUE),
    gam_cases = sum(gam, na.rm = TRUE),
    .groups = "drop"
  ) |>
  mutate(gam_rate = round(gam_cases / assessed, 3))
```

`n()` and the two ways to count — In R

```
summarise(muac, rows = n(), measured = sum(!is.na(muac_mm)))
```

`n()` and the two ways to count — In R

```r
sum(c(TRUE, NA))
#> [1] NA

sum(c(TRUE, NA), na.rm = TRUE)
#> [1] 1
```

The grouping survives, and it is the commonest bug — In R

```
out <- muac |>
  mutate(month = format(screening_date, "%Y-%m")) |>
  group_by(commune, month) |>
  summarise(n = n())
```

The grouping survives, and it is the commonest bug — Example

```
`summarise()` has regrouped the output.
i Summaries were computed grouped by commune and month.
i Output is grouped by commune.
i Use `summarise(.groups = "drop_last")` to silence this message.
```

The grouping survives, and it is the commonest bug — In R

```
class(out)[1]
#> [1] "grouped_df"

dplyr::group_vars(out)
#> [1] "commune"
```

The grouping survives, and it is the commonest bug — In R

```r
out |> mutate(share = n / sum(n))      # share within the commune, not the district
out |> arrange(desc(n))                # sorts within each commune
out |> slice_max(n, n = 5)             # top 5 per commune, not overall
```

The grouping survives, and it is the commonest bug

- Say what you want, every time

The grouping survives, and it is the commonest bug — In R

```
summarise(..., .groups = "drop")        # ungrouped result -- the usual answer
summarise(..., .groups = "drop_last")   # keep all but the last (the default)
summarise(..., .groups = "keep")        # keep all groupings
```

The grouping survives, and it is the commonest bug — In R

```r
muac |>
  summarise(n = n(), .by = c(commune, month))
```

A missing key becomes its own group — In R

```r
d <- tibble::tibble(g = c("a", NA, "a"), v = c(1, 2, 3))

d |> group_by(g) |> summarise(s = sum(v), .groups = "drop")
```

A missing key becomes its own group — Example

```
# A tibble: 2 × 2
  g         s
  <chr> <dbl>
1 a         4
2 <NA>      2
```

A missing key becomes its own group — In R

```
d |> filter(!is.na(g)) |> group_by(g) |> summarise(s = sum(v), .groups = "drop")
```

A missing key becomes its own group — In R

```
stopifnot(!any(is.na(muac$commune)))
```

Unused factor levels: `count()` drops, `table()` keeps — In R

```r
d <- tibble::tibble(k = factor(c("a", "b", "a"), levels = c("a", "b", "c")))

nrow(count(d, k))                 #> 2   -- the empty level is gone
nrow(count(d, k, .drop = FALSE)) #> 3   -- kept
length(table(d$k))                #> 3   -- kept
```

Unused factor levels: `count()` drops, `table()` keeps

- Pass `.drop` explicitly — in anything that produces a reported table

Grouped `mutate()`: a group statistic on every row — In R

```r
muac |>
  group_by(commune) |>
  mutate(
    commune_mean = mean(muac_mm, na.rm = TRUE),
    vs_commune   = muac_mm - commune_mean
  ) |>
  ungroup()
```

Grouped `mutate()`: a group statistic on every row

- Note the `ungroup()` — A grouped frame that escapes into the rest of a script is the same bug as the one above,...

`count()` for when that is all you want — In R

```r
muac |> count(commune, outcome, sort = TRUE)
muac |> count(commune, wt = muac_mm)          # sum a column instead of counting rows
```

Two keys, and the table a report wants — In R

```r
by_commune_month <- muac |>
  mutate(month = format(screening_date, "%Y-%m")) |>
  summarise(cases = sum(gam, na.rm = TRUE), .by = c(commune, month))

wide <- by_commune_month |>
  tidyr::pivot_wider(names_from = month, values_from = cases, values_fill = 0)
```

Writing the result out — In R

```
readr::write_csv(by_commune, here::here("outputs", "tables", "gam_by_commune.csv"))
```

Writing the result out — In R

```r
definitions <- tibble::tibble(
  indicator   = "GAM",
  numerator   = "MUAC < 125 mm or bilateral pitting oedema",
  denominator = "children with a MUAC measurement or a recorded oedema assessment"
)

readr::write_csv(definitions, here::here("outputs", "tables", "definitions.csv"))
```

The three reversals, in one place

	pandas	**R / dplyr**
Missing value in `mean()`	skipped silently	returns `NA` until you say `na.rm`
Missing group key	dropped	kept as its own group
Unused category	dropped by `observed=True`	kept by `table()`, dropped by `count()`

What comes next

- The table aggregates correctly and both halves of every rate come from one call.

Read the full lesson, with runnable code

https://data-analysis.cassion.dev/courses/r-for-programme-data/r-06-grouping