This has incredible benefits for workflow: instead of having to track group identifiers across each output you produce, data and outputs and are linked to group identifiers by row, and you can do further tidying to summarise these complex outputs. You essentially have a summary of your analysis right there in the data frame.

In particular, [purrr::safely() and purrr::quietly()](https://purrr.tidyverse.org/reference/safely.html) wrap a number of R’s output and error handling systems, allowing you to push through errors as you iterate and capture them along with warnings, messages and other output. Instead of getting zero output until every group is behaving, you can get to the end of your analysis and then review to see what didn’t work out.

These two functions already lend themselves well to mapping, but since they return nested lists with each component of the output, it’s not always easy to tell until you burrow in to the returned column which groups returned output, which stopped with errors and which flagged other warnings or messages.

In response, the below two functions of collateral package. Collateral provides two new map variants:

* map\_safely() wraps safely() and,
* map\_quietly() wraps quietly().

As well as automatically wrapping these functions (so you can use them exactly as you would the vanilla map()), these functions style the resulting list (or list-column) using pillar—the same system that provides enhanced tibble output. You can scan down the column and see which mapped elements returned a result, which stopped with errors, and which returned warnings, messages or other output. If you’d like to inspect these kinds of output in more detail (for example, retrieve an error or read returned warnings), you can do so just as you would a manually mapped-and-wrapped function call, using components like $result, $error or $warning.

library(tidyverse)

library(collateral)

test =

# tidy up and trim down for the example

mtcars %>%

rownames\_to\_column(var = "car") %>%

as\_data\_frame() %>%

select(car, cyl, disp, wt) %>%

# spike some rows in cyl == 4 to make them fail

mutate(wt = dplyr::case\_when(

wt < 2 ~ -wt,

TRUE ~ wt)) %>%

# nest and do some operations quietly()

nest(-cyl) %>%

mutate(qlog = map\_quietly(data, ~ log(.$wt)))

test

#> # A tibble: 3 x 4

#> cyl data qlog

#>

#> 1 6 R O \_ \_

#> 2 4 R O \_ W

#> 3 8 R O \_ \_

The captured side effects appear in the tibble columns: (R)results, (O)utput, (M)essages and (W)arnings. If you’re using map\_safely() instead, you’ll see (R)esults and (E)rrors. if we investigate row 2 further, we can see the problem:

test$qlog[[2]]$warnings

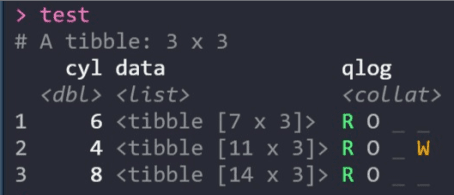
# [1] "NaNs produced"

test$qlog[[2]]$result

# [1] 0.8415672 1.1600209 1.1474025 0.7884574 NaN NaN 0.9021918 NaN 0.7608058

# [10] NaN 1.0224509

Even better: because collateral plugs into pillar, which powers enhanced tibble output, you get color in the terminal!



(Next on the to-do list: getting this going with knitted output!)