Package ‘collateral’

November 19, 2018

Title  Quickly Evaluate Captured Side Effects
Version  0.4.2
Description  The purrr package allows you to capture the side effects (errors, warning, messages and other output) of functions using safely() and quietly(). Using collateral, you can quickly see which elements of a list (or list-column) returned results, which threw errors and which returned warnings or other output.

URL  https://rensa.co/collateral/index.html,
     https://github.com/rensa/collateral

Depends  R (>= 3.1.0)
Imports  purrr, crayon, methods, pillar
License  GPL-3 | file LICENSE
Encoding  UTF-8
LazyData  true
RoxygenNote  6.1.1

BugReports  https://github.com/rensa/collateral/issues
Suggests  dplyr, knitr, magrittr, rmarkdown, tidyverse
VignetteBuilder  knitr
NeedsCompilation  no

Author  James Goldie [aut, cre] (<https://orcid.org/0000-0002-5024-6207>)
Maintainer  James Goldie <me@rensa.co>
Repository  CRAN
Date/Publication  2018-11-19 18:00:23 UTC

R topics documented:

collateral_mappers ..................................................... 2
has ................................................................. 4
summary .......................................................... 5
tally ............................................................. 6

Index  8
Map safely or quietly over a list.

Description

map_safely and map_quietly are variants of map that:

1. wrap the supplied function .f with either safely or quietly, and
2. add a class to the returned output list in order to format it nicely when it (or a tibble it appears in) is printed.

Usage

map_safely(.x, .f, ...)
map_quietly(.x, .f, ...)
map2_safely(.x, .y, .f, ...)
map2_quietly(.x, .y, .f, ...)
pmap_safely(.l, .f, ...)
pmap_quietly(.l, .f, ...)

Arguments

.x A list or atomic vector.
.f A function, formula or atomic vector.
... Other arguments supplied to map.
.y A list or atomic vector, of the same length as .x.
.l A list of lists. The length of .l determines the number of arguments that .f will be called with. List names will be used if present.

Details

map_safely will summarise the returned list with a fixed-width string of two (spaced) columns:

1. If a result component is present, R appears, and
2. If an error component is present, E appears.

If either component is missing, an underscore (_) appears in its place.

Similarly, map_quietly will summarise the returned list with a fixed-width string of four (spaced) columns:

1. If a result component is present, R appears,
2. If an output component is present, 0 appears,
3. If a messages component is present, M appears, and
4. If a warnings component is present, W appears.

If any is missing, an underscore (_) appears in its place.

Variants for iterating over two or more inputs simultaneously are also provided and function identically to their purrr counterparts:

1. map2_safely
2. map2_quietly
3. pmap_safely
4. pmap_quietly

Value

A list of the same length as .x. The list elements contain results and captured side effects as described in safely and quietly.

Examples

library(magrittr)

# like map(), these can be used to iterate over vectors or lists
list("a", 10, 100) %>% map_safely(log)
list(5, -12, 103) %>% map_quietly(log)

suppressMessages(library(tidyverse))

# if you're using tibbles, you can also iterate over list-columns, # such as nested data frames
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)))
has 

Filter elements that contain a type of side effect.

Description

Returns a logical vector indicating which elements contain a type of side effect. If you have a large data frame or list, you can use this to isolate the element that contain warnings, for example, or messages.

Usage

has_results(x)
has_errors(x)
has_warnings(x)
has_messages(x)
has_output(x)

Arguments

x
A safely_mapped or quietly_mapped list to tally.

Details

The has_*() functions power the tally_*() functions and, in turn, the summary methods.

Value

A logical vector, of the same length as x, which is TRUE for elements that contain a type of side effect and FALSE otherwise.

Examples

library(magrittr)

list("a", 10, 100) %>% map_safely(log) %>% has_errors()
list(5, -12, 103) %>% map_quietly(log) %>% has_warnings()
suppressMessages(library(tidyverse))

# if you're working with list-columns, the tally functions are useful
# in conjunction with dplyr::summarise()
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))) %>%
filter(has_warnings(qlog))

---

**summary**

Summarise mapped side effects.

**Description**

The `summary` method for a `safely_mapped` or `quietly_mapped` list (or list-column) prints out the total number of elements (rows), as well as the number that each returned results and errors (for `safely_mapped`) or returned results, output, messages and warnings (for `quietly_mapped`). It also invisibly returns a named vector with these counts.

**Usage**

```r
## S3 method for class 'safely_mapped'
summary(object, ...)
```

```r
## S3 method for class 'quietly_mapped'
summary(object, ...)
```

**Arguments**

- `object` A safely_mapped or quietly_mapped list to summarise.
- `...` Other arguments passed to `summary`.

**Details**

Although the output can be used in tidy workflows (for automated testing, for example), tally functions like `tally_results` tend to be more convenient for this purpose.

Importantly, the `summary` functions tell you how many elements # returned a type of side effect, *not the number of those side effects*. Some list elements might return more than one warning, for example, and these are not counted separately.

**Value**

A named vector containing counts of the components named in `map_safely`.
Examples

```r
library(magrittr)

list("a", 10, 100) %>% map_safely(log) %>% summary()
list(5, -12, 103) %>% map_quietly(log) %>% summary()
```

**tally**  
*Get counts of types of mapped side effects.*

Description

Unlike `summary`, the `tally` functions return counts of individual types of side effects. This makes them easy to use with `summarise`.

Usage

```r
  tally_results(x)
  tally_errors(x)
  tally_warnings(x)
  tally_messages(x)
  tally_output(x)
```

Arguments

- `x`  
  A `safely_mapped` or `quietly_mapped` list to tally.

Details

Importantly, the `tally` functions tell you how many elements returned a type of side effect, *not the number of those side effects*. Some list elements might return more than one warning, for example, and these are not counted separately.

Value

An integer vector of length 1.
Examples

library(magrittr)
list("a", 10, 100) %>% map_safely(log) %>% tally_errors()
list(5, -12, 103) %>% map_quietly(log) %>% tally_warnings()

suppressMessages(library(tidyverse))

# if you're working with list-columns, the tally functions are useful
# in conjunction with dplyr::summarise()
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))) %>%
summarise(
    num_results = tally_results(qlog),
    num_warnings = tally_warnings(qlog))
Index

collateral_mappers, 2

has, 4
has_errors (has), 4
has_messages (has), 4
has_output (has), 4
has_results (has), 4
has_warnings (has), 4

map, 2
map2_quietly (collateral_mappers), 2
map2_safely (collateral_mappers), 2
map_quietly (collateral_mappers), 2
map_safely, 5
map_safely (collateral_mappers), 2

pmap_quietly (collateral_mappers), 2
pmap_safely (collateral_mappers), 2

quietly, 2, 3

safely, 2, 3
summarise, 6
summary, 5, 6

tally, 6
tally_errors (tally), 6
tally_messages (tally), 6
tally_output (tally), 6
tally_results, 5
tally_results (tally), 6
tally_warnings (tally), 6