Package: collateral (via r-universe)

August 20, 2024

Title Quickly Evaluate Captured Side Effects

Version 0.5.2

Description Map functions while capturing results, errors, warnings, messages and other output tidily, then filter and summarise data frames or lists on the basis of those side effects.

URL https://collateral.jamesgoldie.dev, https://github.com/jimjam-slam/collateral

Language en-AU

Depends R (>= 3.1.0)

Imports purrr, crayon, methods, pillar

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.1.1

BugReports https://github.com/jimjam-slam/collateral/issues

Suggests dplyr, tibble, tidyr, furrr, ggplot2, knitr, rmarkdown, testthat

VignetteBuilder knitr

Roxygen list(markdown = TRUE)

Repository https://jimjam-slam.r-universe.dev

RemoteUrl https://github.com/jimjam-slam/collateral

RemoteRef main

RemoteSha 42ca3cb346fb5a444927c255ed7e61d332096126

Contents

collateral_mappers *Map over a list while capturing side effects.*

Description

map_safely(), map_quietly() and map_peacefully() are variants of purrr::map() that wrap the supplied function .f using purrr::safely() and/or purrr::quietly() in order to capture various side effects. Lists mapped in this way have an associated class added to them, allowing them to succinctly summarise captured side effects when displayed in a tibble.

Usage

```
map_safely(.x, .f, otherwise = NULL, quiet = TRUE, ...)

map_quietly(.x, .f, ...)

map_peacefully(.x, .f, ...)

map2_safely(.x, .y, .f, otherwise = NULL, quiet = TRUE, ...)

map2_quietly(.x, .y, .f, ...)

map2_peacefully(.x, .y, .f, ...)

pmap_safely(.l, .f, otherwise = NULL, quiet = TRUE, ...)

pmap_quietly(.l, .f, ...)

pmap_peacefully(.l, .f, ...)

future_map_safely(.x, .f, otherwise = NULL, quiet = TRUE, ...)

future_map_quietly(.x, .f, ...)

future_map_peacefully(.x, .f, ...)

future_map2_safely(.x, .y, .f, otherwise = NULL, quiet = TRUE, ...)

future_map2_quietly(.x, .y, .f, ...)

future_map2_peacefully(.x, .y, .f, ...)

future_pmap_safely(.l, .f, otherwise = NULL, quiet = TRUE, ...)

future_pmap_quietly(.l, .f, ...)

future_pmap_peacefully(.l, .f, ...)
```

Arguments

.x	A list or atomic vector.
.f	A function, formula or atomic vector, as specified by purrr::as_mapper().
otherwise	Default value to use when an error occurs.
quiet	Hide errors (TRUE, the default), or display them as they occur?
...	Other arguments supplied to purrr::map() or its variants, or to furrr::future_map() or its variants..
.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, O 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()

Further variants, prefixed by future_, allow safe or quiet mapping to happen in parallel if you have the furrr package installed:

1. future_map_safely()
2. future_map_quietly()
3. future_map2_safely()
4. future_map2_quietly()
5. future_pmap_safely()
6. future_pmap_quietly()

Value

A list of the same length as `.x`. Each element of the returned list is itself a named list, structured according to the captured side effects. The Details section elaborates on these side effects.

Examples

```
library(tibble)
library(dplyr)
library(tidyr)
library(collateral)

# 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)

# if you're using tibbles, you can also iterate over list-columns,
# such as nested data frames
mtcars %>%
  rownames_to_column(var = "car") %>%
  as_tibble() %>%
  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(data = -cyl) %>%
  mutate(qlog = map_quietly(data, ~ log(.$wt)))
```

---

has *Determine which elements 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.s

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() method.

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(tibble)
library(dplyr)
library(tidyr)
library(collateral)

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

# if you're working with list-columns, the tally functions are useful
# in conjunction with dplyr::summarise()
mtcars %>%
  rownames_to_column(var = "car") %>%
  as_tibble() %>%
  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(data = -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

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

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

## S3 method for class 'peacefully_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()` method tells you how many elements were 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

```
library(tibble)
library(dplyr)
library(tidyr)
library(collateral)

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

`tally` *Determine how many elements contain a type of mapped side effect.*

Description

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

Usage

```
tally_results(x)

tally_errors(x)

tally_warnings(x)

tally_messages(x)

tally_output(x)
```

Arguments

x	A “safely_mappedorquietly_mapped‘ list to tally.

Details

Importantly, the tally functions tell you how many *elements* returned a type of side effect, not how many *side effects* were returned. 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(tibble)
library(dplyr)
library(tidyr)
library(collateral)

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

# if you're working with list-columns, the tally functions are useful
# in conjunction with dplyr::summarise()
mtcars %>%
  rownames_to_column(var = "car") %>%
  as_tibble() %>%
  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(data = -cyl) %>%
  mutate(qlog = map_quietly(data, ~ log(.$wt))) %>%
  summarise(
    num_results = tally_results(qlog),
    num_warnings = tally_warnings(qlog))
```

Index