Package: loopurrr (via r-universe)

September 4, 2024

Title Tranlate purrr iterator functions to regular for loops

Version 0.1.1

Description Makes learning, teaching and debugging iterator functions from the purrr package easy by translating them into regular for loops.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

URL https://github.com/TimTeaFan/loopurrrr

BugReports https://github.com/TimTeaFan/loopurrr/issues

Depends R (>= 3.2), purrr (>= 0.3.4)

Suggests clipr, rstudioapi, testthat (>= 3.0.0), covr

Config/testthat/edition 3

Imports rlang (>= 1.0.0), dplyr, methods

Repository https://timteafan.r-universe.dev

RemoteUrl https://github.com/timteafan/loopurrr

RemoteRef HEAD

RemoteSha 6eb7a789674477ba9a4d922142350c97e95dd417

Contents

as_loop *Translate purrr's iterator functions to regular for loops*

Description

`as_loop()` takes a function call to one of {purrr}'s iterator functions, such as `purrr::map()`, and translates it into a regular `for` loop. Depending on the output context, the translation is either (i) printed to the console, (ii) copied to the clipboard or (iii) directly inserted into RStudio. Note that the latter two options require the {clipr} respectively the {rstudioapi} package to be installed.

The usage is pretty straight-forward: Just wrap a call to a {purrr} iterator function into `as_loop()` or us one of the pipe operators (`|>` or `%>%`) to pipe the function call into `as_loop()`. For details see the examples below.

Usage

```
as_loop(
  .expr,
  simplify = TRUE,
  output_nm = "out",
  idx = "i",
  output_context = default_context(),
  return = c("string", "eval")
)
```

Arguments

`.expr`	A function call to a {purrr} iterator function. See the "Supported functions" section below for an overview of which {purrr} iterator functions are currently supported.
`simplify`	When TRUE, the default, `as_loop()` will run the function call in `.expr` to check two things: (1) Whether the call is valid. If not, an error will be thrown, pointing out that the underlying function call is invalid. (2) Whether the resulting return value contains `NULL`. In this case the `for` loop needs to be more verbose. When `simplify` is set `FALSE` the function call in `.expr` is not checked for errors and the resulting for loop will be more verbose even if `NULL` is not among the return values. It is recommended to set `simplify` to `FALSE` for calculation-heavy function calls.
`output_nm`	sets the name of the resulting output object. The default name is `out`.
`idx`	sets the name of the index of the for loop. The default index is `i`.
`output_context`	An optional output context that defines the output target. Possible values are one or several of: • `"rstudio"`: This will insert the translation to the location where `as_loop()` was run. If it was run from within an R script, the for loop will be inserted there, otherwise in the console. Note that the {rstudioapi} package is required for this option.

- "clipboard": This will copy the for loop translation to the clipboard. Note that the {clipr} package is required for this option.
- "console": This will print the call to the console using cat().

The default setting is to call default_context(). This function first looks at the "loopurrr.output" option. If the option is not specified, then it will default to c("rstudio", "clipboard", "console"). In this case as_loop() will run the output options from left to right (starting with "rstudio") until successful. If neither the rstudioapi package nor the clipr package are installed, the output context will fall back to "console".

return — When set to "string", the default, as_loop() will return the translated code as character strings to the location specified in output_context. When set to "eval", the translated code will be evaluated in a dedicated environment and the output object will be returned. This option is especially for testing whether as_loop() works as expected. It should be irrelevant for most users.

Value

Depending on the return argument the return value is:

1. When output = "string": NULL. As a side-effect, the translated for loop will be returned to the specified output context.
2. When output = "eval": Usually the return value of the output object that is constructed with the for loop. In case of a call to walk, walk2 etc. the (first) input object will be returned.

Supported functions

The following iterator functions from the {purrr} package are currently supported:

```
options(width = 60)
get_supported_fns("as_loop")
#> $map
#>  [1] "map"     "map_at"  "map_chr" "map_dbl" "map_df"
#>  [6] "map_dfc" "map_dfr" "map_if"  "map_int" "map_lgl"
#> [11] "map_raw"
#>
#> $imap
#> [1] "imap"     "imap_chr" "imap_dbl" "imap_dfc" "imap_dfr"
#> [6] "imap_int" "imap_lgl" "imap_raw"
#>
#> $map
#> [1] "map2"     "map2_chr" "map2_dbl" "map2_df"  "map2_dfc"
#> [6] "map2_dfr" "map2_int" "map2_lgl" "map2_raw"
#>
#> $pmap
#> [1] "pmap"     "pmap_chr" "pmap_dbl" "pmap_df"  "pmap_dfc"
#> [6] "pmap_dfr" "pmap_int" "pmap_lgl" "pmap_raw"
#>
#> $lmap
```

```
#> [1] "lmap"    "lmap_at"
#>
#> $modify
#> [1] "modify"    "modify_at" "modify_if" "modify2"
#> [5] "imodify"
#>
#> $walk
#> [1] "iwalk" "pwalk" "walk"  "walk2"
#>
#> $accumulate
#> [1] "accumulate"  "accumulate2"
#>
#> $reduce
#> [1] "reduce"  "reduce2"
```

Examples

If we wrap or pipe a call to `purrr::map()` into `as_loop()` it will be translated into a regular for loop. Depending on the output context, the resulting for loop will either be (i) inserted directly into a script in RStudio, (ii) copied to the clipboard or (iii) printed to the console.

```
x <- list(1, c(1:2), c(1:3))
as_loop(map(x, sum))        # wrap a call in `as_loop()`
map(x, sum) %>% as_loop() # pipe a call into `as_loop()`

# --- convert: `map(x, sum)` as loop --- #
out <- vector("list", length = length(x))

for (i in seq_along(x)) {
  out[[i]] <- sum(x[[i]])
}
# --- end loop --- #
```

The `output_nm` argument lets us specify the name of the resulting output object. In the example below `".res"`. The `idx` argument lets us specify the index to be used. In the example below `"j"`.

```
x <- list(1, c(1:2), c(1:3))
map_dbl(x, sum) %>%
  as_loop(., output_nm = ".res", idx = "j")

# --- convert: `map_dbl(x, sum)` as loop --- #
.res <- vector("double", length = length(x))

for (j in seq_along(x)) {
  .res[[j]] <- sum(x[[j]])
}
# --- end loop --- #
```

When `simplify` is set `FALSE` `as_loop` will neither check the validity of the underlying call nor the expected output. In this case the resulting `for` loop is more verbose. This is because we need to

take the case of NULL in the return values into account. In the example below we further see what happens, when we use an unnamed object, such as 1:3, in the call to purrr::map(). as_loop() assign unnamed objects an internal name. In the example below .inp1.

```
map(1:3, sum) %>% as_loop(., simplify = FALSE)

# --- convert: `map(1:3, sum)` as loop --- #
.inp1 <- 1:3
out <- vector("list", length = length(.inp1))

for (i in seq_along(.inp1)) {
  .tmp <- sum(.inp1[[i]])
  if (!is.null(.tmp))
    out[[i]] <- .tmp
}
# --- end loop --- #
```

---

get_output_opt *Get and set output loopurrr's options*

Description

get_output_opt() and set_output_opt() get and set loopurrr's output options. They are light wrappers around getOption("loopurrr.output") and options("loopurrr.output").

default_context() inspects if the "loopurrr.output" option is set. If the option is not specified, it will default to c("rstudio", "clipboard", "console"). If "console" is not among the output options it will be automatically included as last option.

Usage

```
get_output_opt(default = NULL)

set_output_opt(x = list("rstudio", "clipboard", "console", NULL))

default_context()
```

Arguments

default	If the specified option is not set in the options list, this value is returned. This argument is for internal use only.
x	Either NULL or one or several of "rstudio", "clipboard", "console". If set to more than one option, the output options will be run in order from left to right until successful.

Value

For `get_output_opt()`, the current value set for option `"loopurrr.output"`, or default (which defaults to NULL) if the option is unset.

For `default_context()`, either the current value set for option `"loopurrr.output"`. In this case, if `"console"` is not among the options, it will be automatically included as last option. Or, if option `"loopurrr.output"` is not specified `c("rstudio", "clipboard", "console")`.

Examples

```
set_output_opt(c("clipboard", "rstudio"))

get_ouptut_opt()
#> [1] "clipboard" "rstudio"

default_context()
#> [1] "clipboard" "rstudio" "console"
```

---

get_supported_fns *Show a list of supported function names*

Description

`get_supported_fns()` shows which functions are supported for a specific {loopurrr} function. Currently, only works on `as_loop()`.

Usage

```
get_supported_fns(fn)
```

Arguments

`fn`	The name of a {loopurrr} function as string.

Value

A list of supported function names as named character vectors.

Examples

```
options(width = 60)
get_supported_fns("as_loop")
#> $map
#>  [1] "map"     "map_at"  "map_chr" "map_dbl" "map_df"
#>  [6] "map_dfc" "map_dfr" "map_if"  "map_int" "map_lgl"
#> [11] "map_raw"
#>
#> $imap
```

```
#> [1] "imap"     "imap_chr" "imap_dbl" "imap_dfc" "imap_dfr"
#> [6] "imap_int" "imap_lgl" "imap_raw"
#>
#> $map
#> [1] "map2"     "map2_chr" "map2_dbl" "map2_df"  "map2_dfc"
#> [6] "map2_dfr" "map2_int" "map2_lgl" "map2_raw"
#>
#> $pmap
#> [1] "pmap"     "pmap_chr" "pmap_dbl" "pmap_df"  "pmap_dfc"
#> [6] "pmap_dfr" "pmap_int" "pmap_lgl" "pmap_raw"
#>
#> $lmap
#> [1] "lmap"    "lmap_at"
#>
#> $modify
#> [1] "modify"    "modify_at" "modify_if" "modify2"
#> [5] "imodify"
#>
#> $walk
#> [1] "iwalk" "pwalk" "walk"  "walk2"
#>
#> $accumulate
#> [1] "accumulate"  "accumulate2"
#>
#> $reduce
#> [1] "reduce"  "reduce2"
```

Index