Package ‘evaluate’

June 10, 2024

Type Package
Title Parsing and Evaluation Tools that Provide More Details than the Default
Version 0.24.0
Description Parsing and evaluation tools that make it easy to recreate
the command line behaviour of R.
License MIT + file LICENSE
URL https://github.com/r-lib/evaluate
BugReports https://github.com/r-lib/evaluate/issues
Depends R (>= 4.0.0)
Imports methods
Suggests covr, ggplot2, lattice, rlang, testthat (>= 3.0.0), withr
Config/Needs/website tidyverse/tidytemplate
Config/testthat/edition 3
Encoding UTF-8
RoxygenNote 7.2.3
NeedsCompilation no
Author Hadley Wickham [aut, cre],
Yihui Xie [aut] (<https://orcid.org/0000-0003-0645-5666>),
Michael Lawrence [ctb],
Thomas Kluyver [ctb],
Jeroen Ooms [ctb],
Barret Schloerke [ctb],
Adam Ryczkowski [ctb],
Hiroaki Yutani [ctb],
Michel Lang [ctb],
Karolis Koncevičius [ctb],
Posit Software, PBC [cph, fnd]
Maintainer Hadley Wickham <hadley@posit.co>
Repository CRAN
Date/Publication 2024-06-10 14:10:02 UTC
evaluate

Contents

evaluate ................................................................. 2
flush_console .......................................................... 3
new_output_handler .................................................... 4
parse_all ............................................................... 5
replay ................................................................. 6

Index 7

evaluate Evaluate input and return all details of evaluation.

Description

Compare to eval(), evaluate captures all of the information necessary to recreate the output as if you had copied and pasted the code into a R terminal. It captures messages, warnings, errors and output, all correctly interleaved in the order in which they occurred. It stores the final result, whether or not it should be visible, and the contents of the current graphics device.

Usage

evaluate(
  input,
  envir = parent.frame(),
  enclos = NULL,
  debug = FALSE,
  stop_on_error = 0L,
  keep_warning = TRUE,
  keep_message = TRUE,
  log_echo = FALSE,
  log_warning = FALSE,
  new_device = TRUE,
  output_handler = default_output_handler,
  filename = NULL,
  include_timing = FALSE
)

Arguments

input input object to be parsed and evaluated. May be a string, file connection or function. Passed on to parse_all().

envir environment in which to evaluate expressions.

enclos when envir is a list or data frame, this is treated as the parent environment to envir.

default debug if TRUE, displays information useful for debugging, including all output that evaluate captures.
flush_console

stop_on_error
if 2, evaluation will halt on first error and you will get no results back. If 1, evaluation will stop on first error without signaling the error, and you will get back all results up to that point. If 0 will continue running all code, just as if you’d pasted the code into the command line.

keep_warning, keep_message
whether to record warnings and messages; if FALSE, messages will be suppressed; if NA, they will not be captured (normally they will be sent to the console). Note that if the environment variable R_EVALUATE_BYPASS_MESSAGES is set to true, these arguments will always be set to NA, meaning that messages will not be captured by this function.

log_echo, log_warning
If TRUE, will immediately log code and warnings (respectively) to stderr.

new_device
if TRUE, will open a new graphics device and automatically close it after completion. This prevents evaluation from interfering with your existing graphics environment.

output_handler
an instance of output_handler() that processes the output from the evaluation. The default simply prints the visible return values.

filename
string overriding the base::srcfile() filename.

include_timing
if TRUE, evaluate will wrap each input expression in system.time(), which will be accessed by following replay() call to produce timing information for each evaluated command.

flush_console
An emulation of flush.console() in evaluate()

Description
When evaluate() is evaluating code, the text output is diverted into an internal connection, and there is no way to flush that connection. This function provides a way to "flush" the connection so that any text output can be immediately written out, and more importantly, the text handler (specified in the output_handler argument of evaluate()) will be called, which makes it possible for users to know it when the code produces text output using the handler.

Usage
flush_console()

Note
This function is supposed to be called inside evaluate() (e.g. either a direct evaluate() call or in knitr code chunks).
new_output_handler

Custom output handlers.

Description

An output_handler handles the results of \texttt{evaluate()}, including the values, graphics, conditions. Each type of output is handled by a particular function in the handler object.

Usage

\begin{verbatim}
new_output_handler(
  source = identity,
  text = identity,
  graphics = identity,
  message = identity,
  warning = identity,
  error = identity,
  value = render,
  calling_handlers = list()
)
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{source} Function to handle the echoed source code under evaluation. This function
      should take two arguments (\texttt{src} and \texttt{call}), and return an object that will be
      inserted into the evaluate outputs. \texttt{src} is the unparsed text of the source code, and
      \texttt{call} is the parsed language object. If \texttt{src} is unparsable, \texttt{call} will be
      \texttt{expression()}. Return \texttt{src} for the default evaluate behaviour.Return \texttt{NULL}
      to drop the source from the output.
  \item \texttt{text} Function to handle any textual console output.
  \item \texttt{graphics} Function to handle graphics, as returned by \texttt{recordPlot()}. 
  \item \texttt{message} Function to handle \texttt{message()} output.
  \item \texttt{warning} Function to handle \texttt{warning()} output.
  \item \texttt{error} Function to handle \texttt{stop()} output.
  \item \texttt{value} Function to handle the values returned from evaluation. If it only has one
      argument, only visible values are handled; if it has more arguments, the second
      argument indicates whether the value is visible.
  \item \texttt{calling_handlers} List of \texttt{calling handlers}. These handlers have precedence over the exiting handler
      installed by \texttt{evaluate()} when \texttt{stop_on_error} is set to 0.
\end{itemize}
### Details

The handler functions should accept an output object as their first argument. The return value of the handlers is ignored, except in the case of the `value` handler, where a visible return value is saved in the output list.

Calling the constructor with no arguments results in the default handler, which mimics the behavior of the console by printing visible values.

Note that recursion is common: for example, if `value` does any printing, then the `text` or `graphics` handlers may be called.

### Value

A new output_handler object

---

**parse_all**  
*Parse, retaining comments.*

---

### Description

Works very similarly to `parse`, but also keeps original formatting and comments.

### Usage

```r
parse_all(x, filename = NULL, allow_error = FALSE)
```

### Arguments

- **x**: object to parse. Can be a string, a file connection, or a function. If a connection, will be opened and closed only if it was closed initially.
- **filename**: string overriding the file name
- **allow_error**: whether to allow syntax errors in `x`

### Value

A data.frame with columns `src`, the source code, and `expr`. If there are syntax errors in `x` and `allow_error = TRUE`, the data frame has an attribute `PARSE_ERROR` that stores the error object.
**replay**

Replay a list of evaluated results.

---

**Description**

Replay a list of evaluated results, as if you’d run them in an R terminal.

**Usage**

`replay(x)`

**Arguments**

- **x**
  - result from `evaluate()`

**Examples**

```r
samples <- system.file("tests", "testthat", package = "evaluate")
if (file_test("-d", samples)) {
  replay(evaluate(file(file.path(samples, "order.R"))))
  replay(evaluate(file(file.path(samples, "plot.R"))))
  replay(evaluate(file(file.path(samples, "data.R"))))
}
```
Index

base::srcfile(), 3

calling handlers, 4

eval(), 2
evaluate, 2
evaluate(), 3, 4, 6

flush_console, 3

message(), 4

new_output_handler, 4

output_handler (new_output_handler), 4
output_handler(), 3

parse_all, 5
parse_all(), 2

recordPlot(), 4
replay, 6

stop(), 4

warning(), 4