# Package 'reprex'

June 23, 2018

**Title** Prepare Reproducible Example Code via the Clipboard

**Version** 0.2.0

**Description** Convenience wrapper that uses the 'rmarkdown' package to render small snippets of code to target formats that include both code and output. The goal is to encourage the sharing of small, reproducible, and runnable examples on code-oriented websites, such as <https://stackoverflow.com> and <https://github.com>, or in email. The user's clipboard is the default source of input code and the default target for rendered output. 'reprex' also extracts clean, runnable R code from various common formats, such as copy/paste from an R session.

**License** MIT + file LICENSE

**URL** https://reprex.tidyverse.org, https://github.com/tidyverse/reprex#readme

**BugReports** https://github.com/tidyverse/reprex/issues

**Depends** R (>= 3.1)

**Imports** callr (>= 2.0.0), clipr (>= 0.4.0), rlang, rmarkdown, tools, utils, whisker, withr

**Suggests** covr, devtools, fortunes, knitr, miniUI, rprojroot, rstudioapi, shiny, styler, testthat (>= 2.0.0)

**VignetteBuilder** knitr

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.0.1.9000

**SystemRequirements** pandoc (>= 1.12.3) - http://pandoc.org

**NeedsCompilation** no

**Author** Jennifer Bryan [aut, cre] (<https://orcid.org/0000-0002-6983-2759>), Jim Hester [aut] (<https://orcid.org/0000-0002-2739-7082>), David Robinson [aut], Hadley Wickham [aut] (<https://orcid.org/0000-0003-4757-117X>), RStudio [cph, fnd]

**Maintainer** Jennifer Bryan <jenny@rstudio.com>
**Repository** CRAN
**Date/Publication** 2018-06-22 23:40:09 UTC

# R topics documented:

opt . . . . . 2
reprex . . . . . 3
reprex_addin . . . . . 7
un-reprex . . . . . 8

**Index** **11**

---

`opt` *Consult an option, then default*

---

## Description

Arguments that appear like so in the usage:

```
f(..., arg = opt(DEFAULT), ...)
```

get their value according to this logic:

```
user-specified value or, if not given,
  getOption("reprex.arg") or if does not exist,
    DEFAULT
```

It's shorthand for:

```
f(..., arg = getOption("reprex.arg", DEFAULT), ...)
```

This is not an exported function and should not be called directly.

## Details

Many of the arguments of `reprex()` use `opt()`. If you don't like the official defaults, override them in your `.Rprofile`. Here's an example for someone who dislikes the "Created by ..." string, always wants session info, prefers to restyle their code, uses a winky face comment string, and likes the tidyverse startup message.

```
options(
  reprex.advertise = FALSE,
  reprex.si = TRUE,
  reprex.style = TRUE,
  reprex.comment = "#;-)",
  reprex.tidyverse_quiet = FALSE
)
```

`reprex` *Render a reprex*

## Description

Run a bit of R code using `rmarkdown::render()` and write the rendered result to user's clipboard. The goal is to make it easy to share a small reproducible example ("reprex"), e.g., in a GitHub issue. Reprex source can be

- read from clipboard
- read from current selection or active document in RStudio (with `reprex_addin()`)
- provided directly as expression, character vector, or string
- read from file

## Usage

```
reprex(x = NULL, input = NULL, outfile = NULL, venue = c("gh", "so",
  "ds", "r"), advertise = opt(TRUE), si = opt(FALSE), style = opt(FALSE),
  show = opt(TRUE), comment = opt("#>"), opts_chunk = NULL,
  opts_knit = NULL, tidyverse_quiet = opt(TRUE), std_out_err = opt(FALSE),
  render = TRUE)
```

## Arguments

| | |
|---|---|
| `x` | An expression. If not given, `reprex()` looks for code in `input` or on the clipboard, in that order. |
| `input` | Character. If has length one and lacks a terminating newline, interpreted as the path to a file containing reprex code. Otherwise, assumed to hold reprex code as character vector. |
| `outfile` | Optional basename for output files. When `NULL` (default), reprex writes to temp files below the session temp directory. If `outfile = "foo"`, expect output files in current working directory, like `foo_reprex.R`, `foo_reprex.md`, and, if `venue = "r"`, `foo_rendered.R`. If `outfile = NA`, expect output files in a location and with basename derived from `input`, if sensible, or in current working directory with basename derived from `tempfile()` otherwise. |
| `venue` | Character. Must be one of the following:<br>• "gh" for GitHub, the default<br>• "so" for Stack Overflow<br>• "ds" for Discourse, e.g., community.rstudio.com. Note: this is currently just an alias for "gh"!<br>• "r" or "R" for a runnable R script, with commented output interleaved |
| `advertise` | Logical. Whether to include a footer that describes when and how the reprex was created. Read more about `opt()`. |

si — Logical. Whether to include devtools::session_info(), if available, or sessionInfo() at the end of the reprex. When venue is "gh" or "ds", the session info is wrapped in a collapsible details tag. Read more about opt().

style — Logical. Whether to style code with styler::style_text(). Read more about opt().

show — Logical. Whether to show rendered output in a viewer (RStudio or browser). Read more about opt().

comment — Character. Prefix with which to comment out output, defaults to "#>". Read more about opt().

opts_chunk, opts_knit — Named list. Optional knitr chunk and package options that are forwarded to knitr::opts_chunk$set() and knitr::opts_knit$set(), respectively.

tidyverse_quiet — Logical. Sets the option tidyverse.quiet, which suppresses (TRUE, the default) or includes (FALSE) the startup message for the tidyverse package. Read more about opt().

std_out_err — Logical. Whether to append a section for output sent to stdout and stderr by the reprex rendering process. This can be necessary to reveal output if the reprex spawns child processes or system() calls. Note this cannot be properly interleaved with output from the main R process, nor is there any guarantee that the lines from standard output and standard error are in correct chronological order. See callr::r_safe() for more. Read more about opt().

render — Logical. Whether to render the reprex or just create the templated .R file. Defaults to TRUE. Mostly for internal testing purposes.

## Details

The usual "code + commented output" is returned invisibly, put on the clipboard, and written to file. An HTML preview displays in RStudio's Viewer pane, if available, or in the default browser, otherwise. Leading "> " prompts, are stripped from the input code. Read more at http://reprex.tidyverse.org/.

reprex sets specific knitr options, which you can supplement or override via the opts_chunk and opts_knit arguments or via explicit calls to knitr in your reprex code (see examples). If all you want to override is the comment option, use the dedicated argument, e.g.commment = "#;-)".

- Chunk options default to collapse = TRUE, comment = "#>", error = TRUE. Note that error = TRUE, because a common use case is bug reporting.
- reprex also sets knitr's upload.fun. It defaults to knitr::imgur_upload() so figures produced by the reprex appear properly on GitHub, Stack Overflow, or Discourse. Note that this function requires the packages httr & xml2 or RCurl & XML, depending on your knitr version. When venue = "r", upload.fun is set to identity, so that figures remain local. In that case, you may also want to set outfile.

## Value

Character vector of rendered reprex, invisibly.

## Examples

```
## Not run:
# put some code like this on the clipboard
# (y <- 1:4)
# mean(y)
reprex()

# provide code as an expression
reprex(rbinom(3, size = 10, prob = 0.5))
reprex({y <- 1:4; mean(y)})
reprex({y <- 1:4; mean(y)}, style = TRUE)

# note that you can include newlines in those brackets
# in fact, that is often a good idea
reprex({
  x <- 1:4
  y <- 2:5
  x + y
})

## provide code via character vector
reprex(input = c("x <- 1:4", "y <- 2:5", "x + y"))

## if just one line, terminate with '\n'
reprex(input = "rnorm(3)\n")

## customize the output comment prefix
reprex(rbinom(3, size = 10, prob = 0.5), comment = "#;-)")

# override a default chunk option, in general
reprex({(y <- 1:4); median(y)}, opts_chunk = list(collapse = FALSE))
# the above is simply shorthand for this and produces same result
reprex({
  #+ setup, include = FALSE
  knitr::opts_chunk$set(collapse = FALSE)

  #+ actual-reprex-code
  (y <- 1:4)
  median(y)
})

# add prose, use general markdown formatting
reprex({
  #' # A Big Heading
  #'
  #' Look at my cute example. I love the
  #' [reprex](https://github.com/tidyverse/reprex#readme) package!
  y <- 1:4
  mean(y)
}, advertise = FALSE)

# read reprex from file
```

```
tmp <- file.path(tempdir(), "foofy.R")
writeLines(c("x <- 1:4", "mean(x)"), tmp)
reprex(input = tmp)

# read from file and write to similarly-named outfiles
reprex(input = tmp, outfile = NA)
list.files(dirname(tmp), pattern = "foofy")

# clean up
file.remove(list.files(dirname(tmp), pattern = "foofy", full.names = TRUE))

# write rendered reprex to file
tmp <- file.path(tempdir(), "foofy")
reprex({
  x <- 1:4
  y <- 2:5
  x + y
}, outfile = tmp)
list.files(dirname(tmp), pattern = "foofy")

# clean up
file.remove(list.files(dirname(tmp), pattern = "foofy", full.names = TRUE))

# write reprex to file AND keep figure local too, i.e. don't post to imgur
tmp <- file.path(tempdir(), "foofy")
reprex({
  #' Some prose
  ## regular comment
  (x <- 1:4)
  median(x)
  plot(x)
  }, outfile = tmp, opts_knit = list(upload.fun = identity))
list.files(dirname(tmp), pattern = "foofy")

# clean up
unlink(
  list.files(dirname(tmp), pattern = "foofy", full.names = TRUE),
  recursive = TRUE
)

## target venue = Stack Overflow
## https://stackoverflow.com/editing-help
ret <- reprex({
  x <- 1:4
  y <- 2:5
  x + y
}, venue = "so")
ret

## target venue = R, also good for email or Slack snippets
ret <- reprex({
  x <- 1:4
  y <- 2:5
```

```
  x + y
}, venue = "R")
ret

## include prompt and don't comment the output
## use this when you want to make your code hard to execute :)
reprex({
  x <- 1:4
  y <- 2:5
  x + y
}, opts_chunk = list(comment = NA, prompt = TRUE))

## leading prompts are stripped from source
reprex(input = c("> x <- 1:3", "> median(x)"))

## End(Not run)
```

---

`reprex_addin` *Render a reprex*

---

## Description

`reprex_addin()` opens an RStudio gadget and addin that allows you to say where the reprex source is (clipboard? current selection? active file? other file?) and to control a few other arguments. Appears as "Render reprex" in the RStudio Addins menu.

`reprex_selection()` is an addin that reprexes the current selection, optionally customised by options. Appears as "Reprex selection" in the RStudio Addins menu. Heavy users might want to create a keyboard shortcut.

## Usage

```
reprex_addin()

reprex_selection(venue = getOption("reprex.venue", "gh"))
```

## Arguments

| | |
|---|---|
| venue | Character. Must be one of the following:<br>• "gh" for GitHub, the default<br>• "so" for Stack Overflow<br>• "ds" for Discourse, e.g., community.rstudio.com. Note: this is currently just an alias for "gh"!<br>• "r" or "R" for a runnable R script, with commented output interleaved |

`un-reprex` *Un-render a reprex*

## Description

Recover clean, runnable code from a reprex captured in the wild and write it to user's clipboard. The code is also returned invisibly and optionally written to file. Three different functions address various forms of wild-caught reprex.

## Usage

```
reprex_invert(input = NULL, outfile = NULL, venue = c("gh", "so", "ds",
  "r"), comment = opt("#>"))

reprex_clean(input = NULL, outfile = NULL, comment = opt("#>"))

reprex_rescue(input = NULL, outfile = NULL, prompt = getOption("prompt"),
  continue = getOption("continue"))
```

## Arguments

| | |
|---|---|
| `input` | Character. If has length one and lacks a terminating newline, interpreted as the path to a file containing reprex code. Otherwise, assumed to hold reprex code as character vector. If not provided, the clipboard is consulted for input. |
| `outfile` | Optional basename for output file. When `NULL`, no file is left behind. If `outfile = "foo"`, expect an output file in current working directory named `foo_clean.R`. If `outfile = NA`, expect on output file in a location and with basename derived from `input`, if a path, or in current working directory with basename derived from `tempfile()` otherwise. |
| `venue` | Character. Must be one of the following:<br>• "gh" for GitHub, the default<br>• "so" for Stack Overflow<br>• "ds" for Discourse, e.g., community.rstudio.com. Note: this is currently just an alias for "gh"!<br>• "r" or "R" for a runnable R script, with commented output interleaved |
| `comment` | regular expression that matches commented output lines |
| `prompt` | character, the prompt at the start of R commands |
| `continue` | character, the prompt for continuation lines |

## Value

Character vector holding just the clean R code, invisibly

## Functions

- `reprex_invert`: Attempts to reverse the effect of `reprex()`. When `venue = "r"`, this just becomes a wrapper around `reprex_clean()`.
- `reprex_clean`: Assumes R code is top-level, possibly interleaved with commented output, e.g., a displayed reprex copied from GitHub or the direct output of `reprex(..., venue = "R")`. This function removes commented output.
- `reprex_rescue`: Assumes R code lines start with a prompt and that printed output is top-level, e.g., what you'd get from copy/paste from the R Console. Removes lines of output and strips prompts from lines holding R commands.

## Examples

```
## Not run:
## a rendered reprex can be inverted, at least approximately
tmp_in <- file.path(tempdir(), "roundtrip-input")
x <- reprex({
  #' Some text
  #+ chunk-label-and-options-cannot-be-recovered, message = TRUE
  (x <- 1:4)
  #' More text
  y <- 2:5
  x + y
}, show = FALSE, advertise = FALSE, outfile = tmp_in)
tmp_out <- file.path(tempdir(), "roundtrip-output")
x <- reprex_invert(x, outfile = tmp_out)
x

# clean up
file.remove(list.files(dirname(tmp),pattern = "roundtrip", full.names = TRUE))

## End(Not run)
## Not run:
## a displayed reprex can be cleaned of commented output
tmp <- file.path(tempdir(), "commented-code")
x <- c(
  "## a regular comment, which is retained",
  "(x <- 1:4)",
  "#> [1] 1 2 3 4",
  "median(x)",
  "#> [1] 2.5"
  )
out <- reprex_clean(x, outfile = tmp)
out

# clean up
file.remove(
  list.files(dirname(tmp), pattern = "commented-code", full.names = TRUE)
)

## round trip with reprex(..., venue = "R")
code_in <- c("x <- rnorm(2)", "min(x)")
```

```
res <- reprex(input = code_in, venue = "R", advertise = FALSE)
res
(code_out <- reprex_clean(res))
identical(code_in, code_out)

## End(Not run)
## Not run:
## rescue a reprex that was copied from a live R session
tmp <- file.path(tempdir(), "live-transcript")
x <- c(
  "> ## a regular comment, which is retained",
  "> (x <- 1:4)",
  "[1] 1 2 3 4",
  "> median(x)",
  "[1] 2.5"
)
out <- reprex_rescue(x, outfile = tmp)
out

# clean up
file.remove(
  list.files(dirname(tmp),pattern = "live-transcript", full.names = TRUE)
)

## End(Not run)
```

# Index


callr::r_safe(), *4*

devtools::session_info(), *4*

knitr::imgur_upload(), *4*

opt, 2
opt(), *3*, *4*

reprex, 3
reprex(), *2*, *9*
reprex_addin, 7
reprex_addin(), *3*
reprex_clean *(*un-reprex*)*, 8
reprex_invert *(*un-reprex*)*, 8
reprex_rescue *(*un-reprex*)*, 8
reprex_selection *(*reprex_addin*)*, 7
rmarkdown::render(), *3*

sessionInfo(), *4*
styler::style_text(), *4*

tempfile(), *3*, *8*

un-reprex, 8