Package: cliExtras (via r-universe)

October 29, 2024

Type Package

Title A Collection of Helper Functions for the cli package

Version 0.1.0.9001

Maintainer Eli Pousson <eli.pousson@gmail.com>

Description Extends the cli package with user response prompts and stopifnot equivalents.

License MIT + file LICENSE

URL https://github.com/elipousson/cliExtras, https://elipousson.github.io/cliExtras/

BugReports https://github.com/elipousson/cliExtras/issues

Imports cli (>= 3.4.0), lifecycle, rlang, utils

Suggests covr, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

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

RemoteUrl https://github.com/elipousson/cliExtras

RemoteRef HEAD

RemoteSha b4637a7f80381698d24d04d8e07bdb8888902930

Contents

`cli_abort_ifnot` *Signal an error, warning, or message with a cli formatted message if any expressions in ... are not all TRUE or are all TRUE*

Description

Signal an error, warning, or message with a cli formatted message if any expressions in ... are not all TRUE or are all TRUE

Usage

```
cli_abort_ifnot(
  ...,
  message = NULL,
  call = .envir,
  .envir = parent.frame(),
  .frame = .envir,
  condition = NULL
)

cli_abort_if(
  ...,
  message = NULL,
  call = .envir,
  .envir = parent.frame(),
  .frame = .envir,
  condition = NULL
)

cli_warn_ifnot(..., message = NULL, .envir = parent.frame(), condition = NULL)

cli_warn_if(..., message = NULL, .envir = parent.frame(), condition = NULL)

cli_inform_ifnot(
  ...,
  message = NULL,
  .envir = parent.frame(),
  condition = NULL
)

cli_inform_if(..., message = NULL, .envir = parent.frame(), condition = NULL)
```

Arguments

`...`	Any number of R expressions which should each evaluate to a logical vector. If `...` is named, the names will be passed to as the message parameter. If message is provided, any names for the logical expressions are ignored. If only some items from `...` are named, the missing names are created with `rlang::exprs_auto_name()`. If a single list is provided, the list is assumed to be a named list of logical values or expressions that evaluate to logical values.
`message`	It is formatted via a call to `cli_bullets()`.
`call`	The execution environment of a currently running function, e.g. `call = caller_env()`. The corresponding function call is retrieved and mentioned in error messages as the source of the error. You only need to supply `call` when throwing a condition from a helper function which wouldn't be relevant to mention in the message. Can also be `NULL` or a defused function call to respectively not display any call or hard-code a code to display. For more information about error calls, see Including function calls in error messages.
`.envir`	Environment to evaluate the glue expressions in.
`.frame`	The throwing context. Used as default for `.trace_bottom`, and to determine the internal package to mention in internal errors when `.internal` is `TRUE`.
`condition`	Logical. For `ifnot` style functions, if `FALSE`, signal an error, warning, or message. For `if` style functions, signal an error, warning, or message if `TRUE`. Defaults to `NULL`. Ignored if multiple parameters are provided to `...`

`cli_alert_ifnot` *CLI conditional alerts*

Description

Alerts are typically short status messages. Note: These functions use `wrap = TRUE` by default. Alert messages can be muffled by `set_cli_quiet()` while `cli::cli_inform()` messages are not.

Usage

```
cli_alert_ifnot(
  text = NULL,
  condition = NULL,
  .fn = cli::cli_alert,
  id = NULL,
  class = NULL,
  wrap = TRUE,
  .envir = parent.frame(),
  call = parent.frame()
)

cli_danger_ifnot(text = NULL, condition = NULL, ..., .envir = parent.frame())

cli_info_ifnot(text = NULL, condition = NULL, ..., .envir = parent.frame())

cli_success_ifnot(text = NULL, condition = NULL, ..., .envir = parent.frame())

cli_warning_ifnot(text = NULL, condition = NULL, ..., .envir = parent.frame())

cli_alert_if(
  text = NULL,
  condition = NULL,
  .fn = cli::cli_alert,
  id = NULL,
  class = NULL,
  wrap = TRUE,
  .envir = parent.frame(),
  call = parent.frame()
)

cli_danger_if(text = NULL, condition = NULL, ..., .envir = parent.frame())

cli_info_if(text = NULL, condition = NULL, ..., .envir = parent.frame())

cli_success_if(text = NULL, condition = NULL, ..., .envir = parent.frame())

cli_warning_if(text = NULL, condition = NULL, ..., .envir = parent.frame())
```

Arguments

`text`	Text of the alert.
`condition`	If `TRUE`, display alert for "if" functions. If `FALSE`, display alert for "ifnot" functions.
`.fn`	cli function to use for alert. Defaults to `cli::cli_alert`. Supported options also include "danger", "info", "success", and "warning".
`id`	Id of the alert element. Can be used in themes.

`class`	Class of the alert element. Can be used in themes.
`wrap`	Whether to auto-wrap the text of the alert.
`.envir`	Environment to evaluate the glue expressions in.
`call`	Caller environment. Used to improve error messages for argument checks.
`...`	Additional parameters passed to cli_alert_if or cli_alert_ifnot by functions like `cli_info_ifnot()`.

See Also

`cli::cli_alert()`

Examples

```
## Not run:
if (interactive()) {
  cli_alert_if(text = "Example text", condition = TRUE)

  cli_alert_ifnot(text = "Example text", condition = FALSE)

  cli_alert_success_if(text = "Success!", condition = TRUE)

  cli_alert_danger_ifnot(text = "Danger!", condition = FALSE)
}

## End(Not run)
```

`cli_ask` *Display a Message then Read a Line from the Terminal*

Description

Display a Message then Read a Line from the Terminal

Usage

```
cli_ask(prompt = "?", ..., .envir = rlang::caller_env(), call = .envir)
```

Arguments

`prompt`	Characters to show as user prompt in console following displayed message. A non-breaking space is always placed after the prompt before passing to `readline()`. Defaults to "?".
`...`	Arguments passed on to `cli::cli_bullets` `text` Character vector of items. See details below on how names are interpreted. `id` Optional id of the `div.bullets` element, can be used in themes.

	`class` Optional additional class(es) for the `div.bullets` element.
`.envir`	Environment to evaluate the glue expressions in.
`call`	The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

See Also

`cli_yesno()`

`cli_bulletize` *List of items using bulletize helper*

Description

List of items using bulletize helper

Usage

```
cli_bulletize(
  items,
  bullet = "*",
  n_show = Inf,
  n_fudge = 2,
  style = NULL,
  sep = NULL,
  before = NULL,
  after = NULL,
  id = NULL,
  class = NULL,
  .envir = parent.frame()
)
```

Arguments

`items`	A named vector or list to use in creating a bulletted list with `cli::cli_bullets()`.
`bullet`	Character to use for bullet. Defaults to "".
`n_show`	The maximum number of items to include in the bullet list. Defaults to 5.
`n_fudge`	The minimum number of items to include in summary of additional bullet items. If the summary would only include a number of items equal or less than n_fudge, they are included in the bullet list and the summary is not displayed. Defaults to 2.
`style`	A cli style name, e.g. code, val, file, url
`sep, before, after`	Additional characters or character vectors applied using `paste0(before, sep, out, after)`. Defaults to `NULL`.

`id`	Optional id of the `div.bullets` element, can be used in themes.
`class`	Optional additional class(es) for the `div.bullets` element.
`.envir`	Environment to evaluate the glue expressions in.

`cli_helpers` *Assorted helper functions to format text and vectors for cli messages*

Description

These functions are convenient alternatives to the internal cli functions but may be removed as my understanding of the cli package improves.

- `bracketize()` pastes open and close brackets around a vector.
- `stylize()` appends a style prefix, e.g. "`{.val value}`" to a vector.
- `bulletize()` turns a vector into a neatly abbreviated bullet list.

`bulletize()` is adapted from the gargle package (see `utils-ui.R`). This function is available under a MIT license and is the work of the gargle authors.

Usage

```
bracketize(..., .open = "{", .close = "}", collapse = NULL)

stylize(
  x,
  style = NULL,
  bracket = FALSE,
  .open = "{",
  .close = "}",
  collapse = NULL
)

bulletize(
  x,
  bullet = "*",
  sep = NULL,
  before = NULL,
  after = NULL,
  style = NULL,
  n_show = 5,
  n_fudge = 2,
  bracket = FALSE
)
```

Arguments

`...`	one or more R objects, to be converted to character vectors.
`.open, .close`	Open and close bracket characters.
`collapse`	an optional character string to separate the results. Not `NA_character_`. When `collapse` is a string, the result is always a string (`character` of length 1).
`x`	A vector.
`style`	A cli style name, e.g. code, val, file, url
`bracket`	If `TRUE`, pass x to bracketize.
`bullet`	Character to use for bullet. Defaults to "".
`sep, before, after`	Additional characters or character vectors applied using `paste0(before, sep, out, after)`.
`n_show`	The maximum number of items to include in the bullet list. Defaults to 5.
`n_fudge`	The minimum number of items to include in summary of additional bullet items. If the summary would only include a number of items equal or less than n_fudge, they are included in the bullet list and the summary is not displayed. Defaults to 2.

Examples

```
bracketize("value")

stylize("styled value", "val")

stylize("styled variable", "val", bracket = TRUE)

bulletize(c("val 1", "val 2", "val 3"))

bulletize(rep("val", 20), n_show = 3)
```

`cli_if` *Execute a cli function if a predicate function returns TRUE*

Description

Execute a function if a predicate function returns TRUE. Intended for use with cli functions.

Usage

```
cli_if(
  x = NULL,
  ...,
  .predicate = rlang::is_true,
  .fn = NULL,
  .default = cli::cli_alert,
  .envir = rlang::caller_env(),
  call = rlang::caller_env()
)

cli_ifnot(
  x = NULL,
  ...,
  .predicate = rlang::is_false,
  .fn = NULL,
  .default = cli::cli_alert,
  .envir = rlang::caller_env(),
  call = rlang::caller_env()
)
```

Arguments

`x`	Parameter to passed to .predicate function, Default: `NULL`
`...`	Additional parameters passed to .fn.
`.predicate`	Single parameter predicate function, Defaults to `rlang::is_true` for `cli_if()` or `rlang::is_false` for `cli_ifnot()`. If .predicate returns `TRUE`, execute .fn. Aborts if .predicate does not return a boolean value.
`.fn`	Function to call with ... parameters if x, Default: `NULL`
`.default`	Default function to execute when .predicate function returns `TRUE`, Default: `cli::cli_alert`
`call`	The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.

Value

The output from the .fn function or .default if .fn is `NULL`

Examples

```
cli_if(FALSE, "No alert.")

cli_if(TRUE, "Alert on TRUE!")

cli_ifnot(FALSE, "Alert on FALSE!")
```

cli_list_files *Display a list of files as a list of items*

Description

Display a list of files as a list of items

Usage

```
cli_list_files(
  path = NULL,
  files = NULL,
  pattern = NULL,
  text = NULL,
  bullet = "*",
  n_show = 10,
  show_full = FALSE,
  include_dirs = TRUE,
  return_list = FALSE,
  .envir = current_env(),
  ...
)
```

Arguments

`path`	a character vector of full path names; the default corresponds to the working directory, `getwd()`. Tilde expansion (see `path.expand`) is performed. Missing values will be ignored. Elements with a marked encoding will be converted to the native encoding (and if that fails, considered non-existent).
`files`	List to file names to display. Ignored if path is provided. If path is a vector of existing files, path is used as files. If files share a single directory, path is set to that directory, otherwise path is set to `NULL`.
`pattern`	an optional regular expression. Only file names which match the regular expression will be returned.
`text`	Passed to `cli::cli_alert_info()`. If `NULL` (default), text appears reporting the number of files/folders found at the path (if path provided).
`bullet`	Character defining style to use list of file names.
`n_show`	Number of file names to show in list. The remaining number of files n_show are noted at the end of the list but the file names are not displayed. Defaults to 10.
`show_full`	If `TRUE`, always show the full file path available. If `FALSE`, show just the base name. Set to `FALSE` automatically if path is a vector of file names is a single directory.
`include_dirs`	If `TRUE`, include directories in listed files. Defaults to `FALSE`. Passed to the include.dirs parameter of `base::list.files()` if path is a directory.
`return_list`	If `TRUE`, return the list of files after displaying the cli message. Defaults to `FALSE`.

`.envir` Passed to `cli::cli_inform()`. Defaults to `rlang::current_env()` rather than `parent.frame()` to support evaluation of default message that includes the number of files found at the path.

`...` Arguments passed on to `base::list.files`

- `all.files` a logical value. If `FALSE`, only the names of visible files are returned (following Unix-style visibility, that is files whose name does not start with a dot). If `TRUE`, all file names will be returned.
- `full.names` a logical value. If `TRUE`, the directory path is prepended to the file names to give a relative file path. If `FALSE`, the file names (rather than paths) are returned.
- `recursive` logical. Should the listing recurse into directories?
- `ignore.case` logical. Should pattern-matching be case-insensitive?
- `include.dirs` logical. Should subdirectory names be included in recursive listings? (They always are in non-recursive ones).
- `no..` logical. Should both `"."` and `".."` be excluded also from non-recursive listings?

See Also

`cli::cli_bullets()`

Examples

```
cli_list_files(system.file("R", package = "cliExtras"), n_show = 5)
```

`cli_menu` *Display a Message then Read a Line from the Terminal*

Description

Display a Message then Read a Line from the Terminal

Usage

```
cli_menu(
  choices,
  title = NULL,
  message = "Enter your selection or press {.kbd 0} to exit.",
  prompt = "Selection:",
  exit = 0,
  ind = FALSE,
  id = NULL,
  call = .envir,
  .envir = parent.frame()
)
```

Arguments

`choices`	Required vector or list of choices.
`title`	Title for menu. Character vector passed to `cli::cli_h2()` if title is length 1 or to `cli::cli_bullets()` if length is greater than 1.
`message`	Additional message to display after choices and before prompt.
`prompt`	Characters to show as user prompt in console following displayed menu options. Defaults to "Selection:" (matching `utils::menu()`)
`exit`	Character to use to exit menu.
`ind`	If `TRUE`, return index position, if `FALSE` (default), return item from choices.
`id`	Optional id of the `div.bullets` element, can be used in themes.
`call`	The execution environment of a currently running function, e.g. `caller_env()`. The function will be mentioned in error messages as the source of the error. See the `call` argument of `abort()` for more information.
`.envir`	Environment to evaluate the glue expressions in.

Examples

```
## Not run:
if (interactive()) {
  cli_menu(list("A", "B", "C"), title = "Pick a letter?")

  cli_menu(c("A", "B", "C"), labels = "Alpha")

  cli_menu(list(list(1, 2, 3), "A", "B", "C"), title = "Pick from a list")
}

## End(Not run)
```

`cli_paths` *Display a list of file paths*

Description

[Superseded]

Usage

```
cli_paths(path, msg)
```

Arguments

`path`	One or more file paths.
`msg`	Passed to message for `cli_inform()`

Details

Use `cli_list_files()` instead.

See Also

`cli::cli_bullets()`

`cli_progress_pipe` *Simplified cli progress message for combining with a pipe*

Description

A convenience function for creating progress messages that use piped data as an input.

Usage

```
cli_progress_pipe(
  data,
  ...,
  current = TRUE,
  clear = FALSE,
  .auto_close = TRUE,
  .envir = current_env(),
  time = NULL
)
```

Arguments

`data`	Input data. Reference w/ "data" if .envir is currrent_env().
`...`	Arguments passed on to `cli::cli_progress_bar`

`name` This is typically used as a label, and should be short, at most 20 characters.

`status` New status string of the progress bar, if not `NULL`.

`type` Type of the progress bar. It is used to select a default display if `format` is not specified. Currently supported types:

- `iterator`: e.g. a for loop or a mapping function,
- `tasks`: a (typically small) number of tasks,
- `download`: download of one file,
- `custom`: custom type, `format` must not be `NULL` for this type.

`total` Total number of progress units, or `NA` if it is unknown. `cli_progress_update()` can update the total number of units. This is handy if you don't know the size of a download at the beginning, and also in some other cases. If `format` is set to `NULL`, `format` (plus `format_done` and `format_failed`) will be updated when you change `total` from `NA` to a number. I.e. default format strings will be updated, custom ones won't be.

- `format` Format string. It has to be specified for custom progress bars, otherwise it is optional, and a default display is selected based on the progress bat type and whether the number of total units is known. Format strings may contain glue substitution, the support pluralization and cli styling. See progress-variables for special variables that you can use in the custom format.
- `format_done` Format string for successful termination. By default the same as `format`.
- `format_failed` Format string for unsuccessful termination. By default the same as `format`.
- `auto_terminate` Whether to terminate the progress bar if the number of current units reaches the number of total units.
- `extra` Extra data to add to the progress bar. This can be used in custom format strings for example. It should be a named list. `cli_progress_update()` can update the extra data. Often you can get away with referring to local variables in the format string, and then you don't need to use this argument. Explicitly including these constants or variables in `extra` can result in cleaner code. In the rare cases when you need to refer to the same progress bar from multiple functions, and you can them to `extra`.

`current`	Passed to `cli_progress_bar()`.
`clear`	Whether to remove the progress bar from the screen after it has terminated. Defaults to `FALSE` (`cli::cli_progress_bar()` uses `TRUE`).
`.auto_close`	Passed to `cli_progress_bar()`.
`.envir`	The environment to use for auto-termination and for glue substitution. It is also used to find and set the current progress bar. Defaults to `current_env()` (`cli::cli_progress_bar()` uses `parent.frame()`)
`time`	Passed to `Sys.sleep()` for optional pause after displaying progress message. Defaults to `NULL`.

Examples

```
df <- data.frame("letters" = LETTERS, "numbers" = c(1:26))

df |>
  cli_progress_pipe("Data has {nrow(data)} rows and {ncol(data)} columns.") |>
  head(2)
```

`cli_quiet` *Use rlang to set cli.default_handler to suppressMessages as a local or permanent option*

Description

`cli_quiet()` is a helper to enable a quiet option in other functions.

Usage

```
cli_quiet(quiet = FALSE, push = FALSE, .frame = rlang::caller_env())
```

Arguments

`quiet`	If `FALSE`, leave cli.default_handler option unchanged. If `TRUE`, set cli.default_handler to `suppressMessages` temporaily with `rlang::local_options()` or permanently with `rlang::push_options()`.
`push`	If `TRUE`, set cli.default_handler option with `rlang::push_options()`.
`.frame`	The environment of a stack frame which defines the scope of the temporary options. When the frame returns, the options are set back to their original values.

Examples

```
test_fn <- function(quiet = FALSE) {
  cli_quiet(quiet = quiet)
  cli::cli_alert_info(
    "{.arg quiet} is {.val {quiet}}"
  )
}

options("cli.default_handler" = NULL)

test_fn()

test_fn(quiet = TRUE)
```

`cli_ul_items` *Format a list of items as an unordered list with cli_ul()*

Description

Format a list of items as an unordered list with cli_ul()

Usage

```
cli_ul_items(items, style = c("code", "val"), sep = ": ", bracket = FALSE)
```

Arguments

`items`	Character vector of items, or `NULL`.
`style`	Length 2 character vector indicating cli style to use for names of items and style to use for items.
`sep`	Length 1 character vector with separator between item names and names.
`bracket`	If `TRUE`, pass x to bracketize.

`cli_yesno`	*Yes No with Variable Responses using cli*

Description

Adapted from `yesno::yesno()` and `usethis::ui_yeah()` to work with `cli_inform()`. This function does not respect the cliExtras.quiet option and aborts if the session is not interactive.

Usage

```
cli_yesno(
  message,
 yes = c("Yes", "Definitely", "For sure", "Yup", "Yeah", "I agree", "Absolutely"),
  no = c("No way", "Not now", "Negative", "No", "Nope", "Absolutely not"),
  n_yes = 2,
  n_no = 1,
  call = .envir,
  .envir = rlang::caller_env()
)

check_yes(
  prompt = NULL,
  yes = c("", "Y", "Yes", "Yup", "Yep", "Yeah"),
  message = "Aborted. A yes is required.",
  .envir = rlang::caller_env(),
  call = .envir
)
```

Arguments

`message`	Passed to `cli_inform()`
`yes, no`	Character strings with yes and no options.
`n_yes, n_no`	Number of yes and no options to provide in user prompt.
`call`	The execution environment of a currently running function, e.g. `call = caller_env()`. The corresponding function call is retrieved and mentioned in error messages as the source of the error. You only need to supply `call` when throwing a condition from a helper function which wouldn't be relevant to mention in the message. Can also be `NULL` or a defused function call to respectively not display any call or hard-code a code to display. For more information about error calls, see Including function calls in error messages.
`.envir`	Environment to evaluate the glue expressions in.
`prompt`	For `check_yes()`, the prompt is always preceded by "? " and followed by "(Y/n)" and padded with non-breaking spaces on both sides.

Details

The yesno and usethis packages are both available under an MIT license (yesno LICENSE and usethis LICENSE) and are the work of the yesno and usethis authors.

Value

`TRUE` if yes response and `FALSE` if no response.

`quiet_cli_inform` *Quiet version of* `cli_inform()`

Description

The implementation of the cliExtras.quiet option is based on the implementation of a similar setting in googlesheets4, googledrive, and gargle. If the cliExtras.quiet option is set to `TRUE`, `quiet_cli_inform()` does not trigger a message. It is used by `cli_inform_ifnot()` and `cli_inform_if()` but not by most other functions in cliExtras.

Usage

```
quiet_cli_inform(..., .envir = parent.frame())
```

Arguments

`...`	Additional parameters passed to `cli_inform()`
`.envir`	Environment to evaluate the glue expressions in.

`register_cli_format` *Register a custom cli format method with registerS3method*

Description

Register a custom cli format method with registerS3method

Usage

```
register_cli_format(class = NULL, method = NULL, envir = parent.frame())
```

Arguments

`class, method`	Object class and format function passed to `registerS3method()`.
`envir`	Environment where the S3 method is registered. Defaults to parent.frame().

`set_cli_quiet` *Set the cli.default_handler and cliExtras.quiet options*

Description

[Deprecated]

Usage

```
set_cli_quiet(quiet = FALSE, msg = !quiet)
```

Arguments

`quiet`	If `TRUE` set the cli.default_handler option to `suppressMessages` and cliExtras.quiet option to `TRUE`. Defaults to `FALSE`.
`msg`	If `TRUE`, `set_cli_quiet()` displays a message confirming the option changes. If `FALSE`, the function does not display a message. Defaults to `!quiet`

Index