Package 'arrg'

May 7, 2026

Version 0.1.0

Date 2024-09-23

Title Flexible Argument Parsing for R Scripts

Imports ore

Suggests tinytest, covr

Description Argument parsing for R scripts, with support for long and short Unix-style options including option clustering, positional arguments including those of variable length, and multiple usage patterns which may take different subsets of options.

Encoding UTF-8

License GPL-2

URL https://github.com/jonclayden/arrg

BugReports https://github.com/jonclayden/arrg/issues

RoxygenNote 7.3.2

NeedsCompilation no

Author Jon Clayden [cre, aut] (ORCID: <https://orcid.org/0000-0002-6608-0619>)

Maintainer Jon Clayden <code@clayden.org>

Repository CRAN

Date/Publication 2024-09-25 09:50:02 UTC

Contents

`arrg` *Create an argument parser*

Description

This function creates an argument parser that handles the specified options and usage patterns. To parse arguments or display usage information, the methods `parse` or `show` contained in the return value should be called.

Usage

```
arrg(name, ..., patterns = list(), header = NULL, footer = NULL)
```

Arguments

`name`	The name of the command.
`...`	Option specifications. See `opt()` for details.
`patterns`	A list of usage patterns that are valid for the command, each specifying acceptable options and positional arguments. See `pat()` for details.
`header`, `footer`	Optional paragraphs of text to be prepended and/or appended to the usage text produced by the `show` method of the return value. Typically used to introduce the command or give brief guidance on usage.

Value

A list with function elements

- `parse(args)`: Parse the character vector of arguments passed in, or by default, the value of `commandArgs(trailingOnly=TRUE)`.
- `show(con, width)`: Print a usage summary, detailing the valid options and patterns. Text will be printed to the specified connection, default `stdout()`, and wrapped to the width given, which defaults to the value of the standard `width` option.

Author(s)

Jon Clayden

See Also

`opt()`, `pat()`

Examples

```
# A simple parser for a command called "test" with only one option, -h
p <- arrg("test", opt("h", "Print help"), patterns=list(pat(options="h!")))

# Print out usage information
p$show()

# Parse the option
p$parse("-h")
```

---

opt *Specify an option in long or short form*

Description

This function specifies an option that is accepted by an argument parser. The results of one or more calls to this function are typically passed to `arrg()`.

Usage

```
opt(label, description, arg = FALSE, default = NA_character_)
```

Arguments

`label`	A short-form (single character) and/or long-form label for the option, specified comma-separated in a single string. At most one of each form must be given. Leading hyphens are optional.
`description`	A textual description of the option, for use in the usage summary.
`arg`	The name of the option's argument, if it takes one. Otherwise `FALSE`, indicating no argument.
`default`	A default value for the argument, if one is accepted. This does not have to be a string, and arguments will be coerced to match the mode of the default when parsed. If the option takes no argument the default value will be `FALSE`.

Value

A data frame giving details of the option. This will not usually be used directly, but passed to `arrg()`.

Author(s)

Jon Clayden

See Also

`arrg()`

Examples

```
# A simple flag-style option with no argument
opt("h,help", "Display this usage information and exit")

# An option that takes an integer argument called "count"
opt("n,times", "Run this many times", arg="count", default=1L)
```

---

pat *Specify a usage pattern*

Description

This function is used to specify a valid usage pattern for the command, which may be one of a number of mutually exclusive patterns available. Its return value is generally passed to `arrg()`.

Usage

```
pat(..., options = NULL)
```

Arguments

`...`	Character strings naming positional arguments, if any are valid. Positional arguments are required by default; if not required they should be followed by a question mark. The final positional argument (only) may take multiple values, in which case it should contain an ellipsis (...), before the question mark if the argument is also optional.
`options`	A string naming the long or short labels of options that can be specified with this pattern, comma-separated. Short form options may be given in one letter cluster for convenience. Options are only required if followed by an exclamation mark.

Details

When parsing arguments, patterns are tried in the order specified, and the first valid pattern will be chosen. A pattern will be considered a valid match if all required options and positional arguments have been specified, and no unexpected options are included.

Value

A list capturing the positional arguments, with options in an attribute. This will not usually be used directly, but passed to `arrg()`.

Author(s)

Jon Clayden

See Also

arrg()

Examples

```
# A pattern with no positional arguments, but requiring the -h flag
pat(options="h!")

# A pattern that takes a command and variable number of arguments, and
# accepts the -n and -t options (note the latter are specified in cluster
# form, but "n,t" is also valid)
pat("command", "arg...?", options="nt")
```

Index