# Package ‘giftwrap’

October 13, 2022

**Title** Take Shell Commands and Turn Them into R Functions

**Version** 0.0.4

**Description**
Wrapping command line functions into R functions, which in turn run the command line functions.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.1

**Depends** R (>= 3.5.0)

**Imports** namespace, processx, utils, readr, tibble

**Suggests** testthat, covr

**NeedsCompilation** no

**Author** Chris Cardillo [aut, cre],
David Robinson [ctb]

**Maintainer** Chris Cardillo <cfcardillo23@gmail.com>

**Repository** CRAN

**Date/Publication** 2020-12-11 10:20:05 UTC

## R topics documented:

## `check_lexicon` *Ensure specificity for certain functions*

### Description

Ensure specificity for certain functions

### Usage

```
check_lexicon(lexicon)
```

### Arguments

| | |
|---|---|
| `lexicon` | a lexicon to be evaluated |

### Value

error if lexicon is not properly formatted

## `check_lexicon_name` *Checks lexicon name passed to* `lexcion`

### Description

Checks lexicon name passed to `lexcion`

### Usage

```
check_lexicon_name(lexicon_name)
```

### Arguments

| | |
|---|---|
| `lexicon_name` | the lexicon name passed to `lexcion` |

### Value

error and message if lexicon not in available lexicons

## `check_named` *Check to see if needed parameters are missing*

### Description

Check to see if needed parameters are missing

### Usage

```
check_named(list)
```

### Arguments

| | |
|---|---|
| `list` | a list to check |

### Value

error if list is not named

## `check_null` *Check to see if needed parameters are missing*

### Description

Check to see if needed parameters are missing

### Usage

```
check_null(...)
```

### Arguments

| | |
|---|---|
| `...` | named items to check |

### Value

error if null

## `format_arg` *Create an argument with shell formatting*

### Description

Create an argument with shell formatting

### Usage

```
format_arg(x)
```

### Arguments

| | |
|---|---|
| `x` | the argument to be passed to the shell command. Both named arguments and unnamed arguments work. A list is expected. |

### Value

a shell-formatted argument

## `format_args` *format multiple arguments*

### Description

format multiple arguments

### Usage

```
format_args(...)
```

### Arguments

| | |
|---|---|
| `...` | named and unnamed arguments to be formatted |

### Value

a vector of properly formatted arguments

## giftwrap *The caller function that will live inside of factory function*

### Description

The caller function that will live inside of factory function

### Usage

```
giftwrap(command, ..., process_echo = TRUE)
```

### Arguments

| | |
|---|---|
| `command` | a shell command |
| `...` | named and unnamed arguments to be giftwrapped |
| `process_echo` | set to FALSE to suppress terminal messages when command is run |

### Value

messages from the running command, errors if failure

## lexicon *For fetching a giftwrap lexicon*

### Description

For fetching a giftwrap lexicon

### Usage

```
lexicon(lexicon)
```

### Arguments

| | |
|---|---|
| `lexicon` | The name of a giftwrap lexicon |

### Value

a dataframe of lexicon

## temporary *Write an object to a temporary file*

### Description

Write an object to a temporary CSV, RDS, or raw text file, and return the path to that file. This can be used, for example, as an argument to a giftwrapped function.

### Usage

```
temp_csv(x, ...)

temp_rds(x, ...)

temp_lines(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | An object |
| `...` | Extra arguments passed to the relevant writing function (such as `write_csv` for `temp_csv`) |

### Value

A string with the temporary filename containing the output. This will be in the temporary per-session directory.

### See Also

tempfile

### Examples

```
wrap_commands("cat")

cat(temp_csv(mtcars))
```

## wrap_commands *To convert shell commands into giftwrapped functions*

### Description

To convert shell commands into giftwrapped functions

### Usage

```
wrap_commands(
  ...,
  env = parent.frame(),
  base_remove = NULL,
  use_namespace = NULL
)
```

### Arguments

| | |
|---|---|
| `...` | shell commands to be turned into giftwrapped functions |
| `env` | the environment in which the function should be created |
| `base_remove` | remove the base from the function name by adding the base name here |
| `use_namespace` | a character string of a namespace to use for the resulting functions |

### Value

A function or functions exported to the specified environment

## wrap_lexicon *Load a functions from a lexicon into an environment*

### Description

Load a functions from a lexicon into an environment

### Usage

```
wrap_lexicon(
  lexicon,
  commands = NULL,
  subcommands = NULL,
  drop_base = FALSE,
  env = parent.frame(),
  use_namespace = NULL
)
```

### Arguments

`lexicon` a dataframe containing columns for base, command, subcommand, and giftwrap_command (the full command)

`commands` regex filtering for any commands in the lexicon

`subcommands` regex filtering for any subcommands in the lexicon

`drop_base` drop the base-level command from the caller function

`env` the environment into which the giftwrap functions should be exported

`use_namespace` a character string of a namespace for giftwrap to create and export the lexicon functions into

### Value

Functions exported to the specified environment

# Index