Package: namespacify (via r-universe)

November 14, 2024

Title Add Namespace Prefixes

Version 0.0.0.9000

Description Add namespace prefixes to R functions in scripts to streamline package development.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports cli, clipr, dplyr, fs, glue, here, purrr, rlang, stringr

URL https://github.com/cbedwards-dfw/namespacify

BugReports https://github.com/cbedwards-dfw/namespacify/issues

Config/pak/sysreqs make libicu-dev libx11-dev

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

RemoteUrl https://github.com/cbedwards-dfw/namespacify

RemoteRef HEAD

RemoteSha 4c28d760f03d9d16b75297c9ba7819bea68ac58e

Contents

detect_packages *Detect packages likely to be relevant for namespacify*

Description

This function can be run before `namespacify()` to identify packages that are being loaded in the specified folder or files. This function (invisibly) returns a vector of package names, and loads a corresponding `usethis::use_package()` call into the clipboard. Before adding the packages to the namespace with use_package() or running namespacify, consider reviewing the package list to make sure that it makes sense: it might include calls to meta packages like `tidyverse`, or it might include outdated library calls for packages that are no longer in use.

Usage

```
detect_packages(folder = NULL)
```

Arguments

`folder`	Identify the folder or files to apply namespace prefixes to. If Defaults to the "./R/" folder, which is where the R files of an R package live.

Value

Vector of packages in use, which can be used as the `packages` argument for `namespacify()`.

identify_files *Identify files to namespacify*

Description

For internal use.

Usage

```
identify_files(folder, verbose = TRUE)
```

Arguments

`folder`	Identify the folder or files to apply namespace prefixes to. If Defaults to the "./R/" folder, which is where the R files of an R package live.
`verbose`	Print details to console? Defaults to TRUE.

Value

vector of R file paths

make_prefix_vector *Title*

Description

Title

Usage

```
make_prefix_vector(
  packages,
  error.on.collision = TRUE,
  funs.ignore = NULL,
  verbose = TRUE
)
```

Arguments

packages	character string of packages to apply namespace prefixes for. In the specified files (below), all functions used by these packages will be the package prefix (e.g. "dplyr::"). To detect likely packages to include, see `detect_packages()`.
error.on.collision	If two or more packages share a function name (e.g., {stats} and {dplyr}), should the function error (TRUE) or warn (FALSE)
funs.ignore	Character vector of functions to skip when adding namespace prefixes as a solution to function collisions. Defaults to NULL.
verbose	Print details to console? Defaults to TRUE.

Value

Vector of patterns to replace; vector names are what should be replaced. Designed to work with `stringr::string_replace_all()`.

Examples

```
## Not run:
make_prefix_vector(packages = c("dplyr", "ggplot2"))

## End(Not run)
```

`namespacify` *Apply namespace prefixes to one or more files*

Description

Workhorse function – this can be called to add the appropriate namespace prefixes to all functions of the specified package(s) (e.g., adding `dplyr::` to `mutate()`) for specific files, all R-related files in a folder, or all R-related files in the "R/" folder (see the `folder` argument for details). Users must provide the packages for which prefixes can be added, but `detect_packages()` provides a suggested list by scanning for `library()` and `require()` calls in the files to be updated. See Details for notes on how function collisions are handled.

Usage

```
namespacify(
  packages,
  error.on.collision = TRUE,
  funs.ignore = NULL,
  verbose = TRUE,
  folder = NULL
)
```

Arguments

`packages`	character string of packages to apply namespace prefixes for. In the specified files (below), all functions used by these packages will be the package prefix (e.g. "dplyr::"). To detect likely packages to include, see `detect_packages()`.
`error.on.collision`	If two or more packages share a function name (e.g., {stats} and {dplyr}), should the function error (TRUE) or warn (FALSE)
`funs.ignore`	Character vector of functions to skip when adding namespace prefixes as a solution to function collisions. Defaults to NULL.
`verbose`	Print details to console? Defaults to TRUE.
`folder`	Identify the folder or files to apply namespace prefixes to. If Defaults to the "./R/" folder, which is where the R files of an R package live.

Details

There is special handling for "function collisions" – cases when two or more specified packages have one or more functions with the same name. If `error.on.collision` is set to TRUE, collisions will cause `namspacify` to identify collisions, and then stop. Users can then specify individual functions to ignore with the `funs.ignore` argument. This is the safest approach to ambiguity.

For example, {dplyr} and {stats} both have a `filter()` function. `namespacify()` doesn't know whether to replace `filter(` with `dplyr::filter(` or `stats::filter(`. For this reason, `namespacify()` will error unless a user included "filter" in the `funs.ignore` argument. Then the user can manually add the appropriate prefix within their scripts, identifying the specific package in each case.

In cases where collisions are common and anticipated, users can set `error.on.collision` to FALSE. In this case, `namespacify()` will instead provide a warning, and then in cases of collisions will use the first relevant package in the `packages` argument. This may be useful when including packages that reexport functions. In that case, two packages may have identical functions with the same name, and the choice of package doesn't matter; setting `error.on.collision` to FALSE will allow the assignment of a relevant namespace prefix to all functions. (Note that reexporting among the tidyverse packages is already addressed; see details see details).

Dev note: The Tidyverse packages frequently re-export functions (e.g., dplyr includes functions from the tibble package). This means that including multiple tidyverse packages will automatically lead to "function collisions", even though they are identical versions of the function, so the choice of namespace doesn't matter. Right now I have a workaround that skips these internal-to-tidyverse collisions – these within-tidyverse collisions are not counted as "function collisions" by `namespace`. However, my workaround is janky – in cases of within-tidyverse collisions it uses the first relevant tidyverse package listed in the `packages` argument.A cleaner solution would be to identify which package is being rexported *from*, and use that. I could not identify an easy way to do so.

Value

Nothing

Examples

```
## Not run:
namespacify(packages = c("dplyr", "ggplot2", "purrr", "tidyr",
"DBI", "readr", "stringr", "tidyselect"))

## End(Not run)
```

`namespacify_file` *Apply namespace prefixes to files*

Description

Apply namespace prefixes to files

Usage

```
namespacify_file(file, vec.replacement, verbose = TRUE)
```

Arguments

`file`	character string of filepath for file to apply prefixes to.
`vec.replacement`	Vector of patterns to replace; vector names are what should be replaced. Typically generated by `make_prefix_vector`.
`verbose`	Print details to console? Defaults to TRUE.

Value

none

`namespacify_text` *Apply namespace prefixes to text.*

Description

Apply namespace prefixes to text.

Usage

```
namespacify_text(original.text, vec.replacement)
```

Arguments

`original.text`	character vector with code to be updated
`vec.replacement`	vector of replacements generated from make_prefix_vector

Value

Replaced text

Index