# Package 'conflicted'

October 12, 2022

**Title** An Alternative Conflict Resolution Strategy

**Version** 1.1.0

**Description** R's default conflict management system gives the most recently loaded package precedence. This can make it hard to detect conflicts, particularly when they arise because a package update creates ambiguity that did not previously exist. 'conflicted' takes a different approach, making every conflict an error and forcing you to choose which function to use.

**License** MIT + file LICENSE

**URL** https://conflicted.r-lib.org, https://github.com/r-lib/conflicted

**BugReports** https://github.com/r-lib/conflicted/issues

**Depends** R (>= 3.2)

**Imports** memoise, rlang (>= 0.3.4)

**Suggests** callr, crayon, dplyr, Matrix, methods, pkgload, testthat (>= 3.0.0)

**Encoding** UTF-8

**RoxygenNote** 7.1.2

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Hadley Wickham [aut, cre], RStudio [cph]

**Maintainer** Hadley Wickham <hadley@rstudio.com>

**Repository** CRAN

**Date/Publication** 2021-11-26 22:00:02 UTC

## R topics documented:

conflict_prefer *Persistently prefer one function over another*

## Description

conflict_prefer() allows you to declare "winners" of conflicts. You can either declare a specific pairing (i.e. dplyr::filter() beats base::filter()), or an overall winner (i.e. dplyr::filter() beats all comers).

Use conficted_prefer_all() to prefer all functions in a package, or conflicted_prefer_matching() to prefer functions that match a regular expression.

## Usage

```
conflict_prefer(name, winner, losers = NULL, quiet = FALSE)

conflict_prefer_matching(pattern, winner, losers = NULL, quiet = FALSE)

conflict_prefer_all(winner, losers = NULL, quiet = FALSE)
```

## Arguments

| | |
|---|---|
| name | Name of function. |
| winner | Name of package that should win the conflict. |
| losers | Optional vector of packages that should lose the conflict. If omitted, winner will beat all comers. |
| quiet | If TRUE, all output will be suppressed |
| pattern | Regular expression used to select objects from the winner package. |

## Best practices

I recommend placing calls to conflict_prefer() at the top of your script, immediately underneath the relevant library() call.

## Examples

```
# Prefer over all other packages
conflict_prefer("filter", "dplyr")

# Prefer over specified package or packages
conflict_prefer("filter", "dplyr", "base")
conflict_prefer("filter", "dplyr", c("base", "filtration"))

# Prefer many functions that match a pattern
## Not run:
# Prefer col_* from vroom
conflict_prefer_matching("^col_", "vroom")
```

```
## End(Not run)
# Or all functions from a package:
## Not run:
# Prefer all tidylog functions over dtplyr functions
conflict_prefer_all("tidylog", "dtplyr")

## End(Not run)
```

---

`conflict_scout` *Find conflicts amongst a set of packages*

---

## Description

`conflict_scout()` is the workhorse behind the conflicted package. You can call it directly yourself if you want to see all conflicts before hitting them in practice.

## Usage

```
conflict_scout(pkgs = NULL)
```

## Arguments

| | |
|---|---|
| `pkgs` | Set of packages for which to report conflicts. If `NULL`, the default, will report conflicts for all loaded packages |

## Value

A named list of character vectors. The names are functions and the values are the packages where they appear. If there is only a single package listed, it means that an automated disambiguation has selected that function.

A user friendly print method displays the result as bulleted list.

## Examples

```
conflict_scout()
```

# Index