Handling R Packages

Fan Wang

2024-01-01

Contents

1 R Package Installation

Go to the **RMD**, **R**, **PDF**, or **HTML** version of this file. Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

1.1 Duplicate function names across packages

dplyr::filter and stats::filter are two functions from two popular packages that have the same name. And this leads to erros, when `dplyr::filter` is confused for `stats::filter`. We use the conflicted::conflict_prefer to resolve this issue.

This is an issue related to namespaces.

Below return the environment on the search path via rlang::search_envs():

```
print(rlang::search_envs())
```

```
##  [[1]] $ <env: global>
##  [[2]] $ <env: .conflicts>
##  [[3]] $ <env: tools:rstudio>
##  [[4]] $ <env: package:stats>
##  [[5]] $ <env: package:graphics>
##  [[6]] $ <env: package:grDevices>
##  [[7]] $ <env: package:utils>
##  [[8]] $ <env: package:datasets>
##  [[9]] $ <env: package:reticulate>
## [[10]] $ <env: package:conflicted>
## [[11]] $ <env: package:formatR>
## [[12]] $ <env: package:REconTools>
## [[13]] $ <env: package:kableExtra>
## [[14]] $ <env: package:knitr>
## [[15]] $ <env: package:lubridate>
## [[16]] $ <env: package:forcats>
## [[17]] $ <env: package:stringr>
## [[18]] $ <env: package:dplyr>
## [[19]] $ <env: package:purrr>
## [[20]] $ <env: package:readr>
## ... and 7 more environments
```

We can use tidyverse_conflicts() to "lists all the conflicts between packages in the tidyverse and other packages that you have loaded". We can see that we have problems due to `filter`, `lag`, and `group_rows`.

```
tidyverse_conflicts()
```

```
## -- Conflicts ------------------------------------------------ tidyverse_conflicts() --
## x stats::filter()          masks dplyr::filter()
## x kableExtra::group_rows() masks dplyr::group_rows()
## x stats::lag()             masks dplyr::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

For example, the code below fails:

```
library(stats)
library(dplyr)
as_tibble(mtcars, rownames = "car") %>% filter(car == "Valiant")

# Error message
# > as_tibble(mtcars, rownames = "car") %>% filter(car == "Valiant")
# Error: object 'car' not found
```

The code below works, because we explicitly write `dplyr::filter`:

```
library(stats)
library(dplyr)
print(as_tibble(mtcars, rownames = "car") %>% dplyr::filter(car == "Valiant"))
```

To deal with this, we use the conflicted::conflict_prefer to resolve this issue. Now we can run the `filter` function safely, knowing that it is the `dplyr::filter` function will be used.

```
library(conflicted)
conflict_prefer("filter", "dplyr", "stats")
library(stats)
library(dplyr)
print(as_tibble(mtcars, rownames = "car") %>% filter(car == "Valiant"))

# > conflict_prefer("filter", "dplyr", "stats")
# [conflicted] Will prefer dplyr::filter over stats::filter.
# > library(stats)
# > library(dplyr)
# > print(as_tibble(mtcars, rownames = "car") %>% filter(car == "Valiant"))
# # A tibble: 1 × 12
#   car       mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb
#   <chr>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
# 1 Valiant  18.1     6   225   105  2.76  3.46  20.2     1     0     3     1
```