Handling Unbound Tidyr Variables Using Rlang Data Masking

2025-11-30

Table of contents

Today I Learned added on 2025-11-19, learned on 2025-11-18; edited on 2026-06-14.

During some routine R-package development, I noticed a `NOTE` in a GitHub Actions (`Run r-lib/actions/check-r-package@v2`) failure, produced by the workflow `R-CMD-check.yaml`:

R CMD Check Workflow

```yaml
# Workflow derived from https://github.com/r-lib/actions/tree/v2/examples
# Need help debugging build failures? Start at
 ↪  https://github.com/r-lib/actions#where-to-find-help
name: R-CMD-check.yaml

on:
  pull_request:
  push:
    branches: [main]

concurrency:
  group: ${{ github.workflow }}-${{ github.head_ref || github.ref }}
  cancel-in-progress: true

permissions: read-all

jobs:
  R-CMD-check:
    runs-on: ${{ matrix.config.os }}

    name: ${{ matrix.config.os }} (${{ matrix.config.r }})

    strategy:
      fail-fast: false
      matrix:
        config:
          - {os: macos-latest,   r: 'release'}
          - {os: windows-latest, r: 'release'}
          - {os: ubuntu-latest,  r: 'release'}

    env:
      GITHUB_PAT: ${{ secrets.GITHUB_TOKEN }}
      R_KEEP_PKG_SOURCE: yes

    steps:
      - uses: actions/checkout@v4
```

```
      - uses: r-lib/actions/setup-pandoc@v2

      - uses: r-lib/actions/setup-r@v2
        with:
          r-version: ${{ matrix.config.r }}
          http-user-agent: ${{ matrix.config.http-user-agent }}
          use-public-rspm: true

      - uses: r-lib/actions/setup-r-dependencies@v2
        with:
          extra-packages: any::rcmdcheck
          needs: check

      - uses: r-lib/actions/check-r-package@v2
        with:
          upload-snapshots: true
          build_args: 'c("--no-manual","--compact-vignettes=gs+qpdf")'
```

Here is a section of the `NOTE`:

```
* checking R code for possible problems ... NOTE
check_authorized_users: no visible binding for global variable ‘.data’
  (/Users/runner/work/hubhelpr/hubhelpr/check/hubhelpr.Rcheck/00_pkg_src/hubhelpr/R/check_au
47)
check_authorized_users: no visible global function definition for
  ‘na.omit’
  (/Users/runner/work/hubhelpr/hubhelpr/check/hubhelpr.Rcheck/00_pkg_src/hubhelpr/R/check_au
47)
```

For more context, here is the relevant code that the above segment refers to:

```
  changed_dirs_tbl <- tibble::tibble(dir = changed_model_ids)

  authorization_check <- changed_dirs_tbl |>
      dplyr::left_join(model_metadata, by = "dir", na_matches = "never") |>
      dplyr::group_by(.data$dir) |>
      dplyr::summarize(
          modifiable = all(tidyr::replace_na(.data$is_model_dir, FALSE)),
          actor_authorized = !!gh_actor %in% .data$designated_github_users,
          authorized_users = paste(
              na.omit(.data$designated_github_users),
              collapse = ", "
          ),
          has_authorized_users =
           ↪  length(na.omit(.data$designated_github_users)) > 0,
          .groups = "drop"
      )
```

The `tidyverse` documentation for `dplyr` has a section on this `NOTE` (see https://dplyr.tidyverse.org/articles/in-packages.html). In particular, they write:

If you're writing a package and you have a function that uses data masking or tidy selection:

```
my_summary_function <- function(data) {
 data %>%
   select(grp, x, y) %>%
   filter(x > 0) %>%
   group_by(grp) %>%
   summarise(y = mean(y), n = n())
}
```

You'll get an `NOTE` because `R CMD check` doesn't know that `dplyr` functions use tidy evaluation:

```
N  checking R code for possible problems
   my_summary_function: no visible binding for global variable ‘grp’, ‘x’, ‘y’
   Undefined global functions or variables:
     grp x y
```

To eliminate this note:

- For data masking, import `.data` from rlang and then use `.data$var` instead of `var`.
- For tidy selection, use `"var"` instead of `var`.

That yields:

```
#' @importFrom rlang .data
my_summary_function <- function(data) {
  data %>%
    select("grp", "x", "y") %>%
    filter(.data$x > 0) %>%
    group_by(.data$grp) %>%
    summarise(y = mean(.data$y), n = n())
}
```

If I wanted to simply get rid of the `NOTE`, I could add `@importFrom rlang .data` to my problematic file and then import `rlang` in my `DESCRIPTION` file. Before exploring this further through a small example, I wanted to see what `rlang` had to say.

Both https://rlang.r-lib.org/ and https://rlang.r-lib.org/reference/topic-data-mask.html provide copious and somewhat interesting reference information. These statements (which you can read about further) in the latter article, seemed more relevant:

> To keep these objects and functions in scope, the data frame is inserted at the bottom of the current chain of environments. It comes first and has precedence over the user environment. In other words, it masks the user environment.

and

> The tidyverse embraced the data-masking approach in packages like ggplot2 and dplyr and eventually developed its own programming framework in the rlang package. None of this would have been possible without the following landmark developments from S and R authors.

For the example, I need the following:

- An `R-CMD-check.yaml` workflow file (see dropdown above).
- An R package using `dplyr`.

Steps:

- In `R`, run: `install.packages(c("devtools", "roxygen2"))`.

REMAINDER PENDING.