styleguide

Google’s R Style Guide

R is a high-level programming language used primarily for statistical computing and graphics. The goal of the R Programming Style Guide is to make our R code easier to read, share, and verify.

The Google R Style Guide is a fork of the Tidyverse Style Guide by Hadley Wickham license. Google modifications were developed in collaboration with the internal R user community. The rest of this document explains Google’s primary differences with the Tidyverse guide, and why these differences exist.

Syntax

Naming conventions

Google prefers identifying functions with `BigCamelCase` to clearly distinguish them from other objects.

```
# Good
DoNothing <- function() {
  return(invisible(NULL))
}
```

The names of private functions should begin with a dot. This helps communicate both the origin of the function and its intended use.

```
# Good
.DoNothingPrivately <- function() {
  return(invisible(NULL))
}
```

We previously recommended naming objects with `dot.case`. We’re moving away from that, as it creates confusion with S3 methods.

Don’t use attach()

The possibilities for creating errors when using `attach()` are numerous.

Pipes

Right-hand assignment

We do not support using right-hand assignment.

```
# Bad
iris %>%
  dplyr::summarize(max_petal = max(Petal.Width)) -> results
```

This convention differs substantially from practices in other languages and makes it harder to see in code where an object is defined. E.g. searching for `foo <-` is easier than searching for `foo <-` and `-> foo` (possibly split over lines).

Use explicit returns

Do not rely on R’s implicit return feature. It is better to be clear about your intent to `return()` an object.

```
# Good
AddValues <- function(x, y) {
  return(x + y)
}

# Bad
AddValues <- function(x, y) {
  x + y
}
```

Qualifying namespaces

Users should explicitly qualify namespaces for all external functions.

```
# Good
purrr::map()
```

We discourage using the `@import` Roxygen tag to bring in all functions into a NAMESPACE. Google has a very big R codebase, and importing all functions creates too much risk for name collisions.

While there is a small performance penalty for using `::`, it makes it easier to understand dependencies in your code. There are some exceptions to this rule.

- Infix functions (`%name%`) always need to be imported.
- Certain `rlang` pronouns, notably `.data`, need to be imported.
- Functions from default R packages, including `datasets`, `utils`, `grDevices`, `graphics`, `stats` and `methods`. If needed, you can `@import` the full package.

When importing functions, place the `@importFrom` tag in the Roxygen header above the function where the external dependency is used.

Documentation

Package-level documentation

All packages should have a package documentation file, in a `packagename-package.R` file.