The Journal of Open Source Software

Static Code Analysis for R

Jim Hester[1]

1 Netflix

DOI:

Software

- Review
- Repository
- Archive

Submitted:
Published:

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC-BY).

Statement of Need

The R programming language (R Core Team, 2023) is a popular choice for statistical analysis and visualization, and is used by a wide range of researchers and data scientists. The `{lintr}` package is an open-source R package that provides static code analysis to check for a variety of common problems related to readability, efficiency, consistency, style, etc. It is designed to be easy to use and integrate into existing workflows, and can be run from the command line or used as part of an automated build or continuous integration process. `{lintr}` also integrates with a number of popular IDEs and text editors, such as RStudio and Visual Studio Code, making it convenient for users to run `{lintr}` checks on their code as they work.

Features

There are over 85 linters offered by `{lintr}`!

```r
library(lintr)

length(all_linters())
#> [1] 87
```

Naturally, we can't discuss all of them here. To see details about all available linters, we encourage readers to see https://lintr.r-lib.org/dev/reference/index.html#individual-linters.

We will showcase one linter for each kind of common problem found in R code.

- **Best practices**

`{lintr}` offers linters that can detect problematic antipatterns and suggest alternative patterns that follow best practices.

For example, usage of vectorized `&` and `|` logical operators in conditional statements is error-prone, and scalar `&&` and `||`, respectively, are to be preferred. The `vector_logic_linter()` linter detects such problematic usages.

```r
lint(
  text = "if (x & y) 1",
  linters = vector_logic_linter()
)
#> <text>:1:7: warning: [vector_logic_linter] Conditional expressions require scal
#> if (x & y) 1
#>       ^
```

- **Efficiency**

Sometimes the users might not be aware of a more efficient way offered by R for carrying out a computation. `{lintr}` offers linters to provide suggestions to improve code efficiency.

```r
lint(
  text = "any(is.na(x), na.rm = TRUE)",
  linters = any_is_na_linter()
)
#> <text>:1:1: warning: [any_is_na_linter] anyNA(x) is better than any(is.na(x)).
#> any(is.na(x), na.rm = TRUE)
#> ^~~~~~~~~~~~~~~~~~~~~~~~~~
```

- **Readability**

Coders spend significantly more time reading compared to writing code (McConnell, 2004). Thus, writing readable code makes the code more maintainable and reduces the possibility of introducing bugs stemming from a poor understanding of the code.

`{lintr}` provides a number of linters that suggest more readable alternatives. For example, `function_left_parentheses_linter()`.

```r
lint(
  text = "stats::sd (c (x, y, z))",
  linters = function_left_parentheses_linter()
)
#> <text>:1:10: style: [function_left_parentheses_linter] Remove spaces before the
#> stats::sd (c (x, y, z))
#>          ^
#> <text>:1:13: style: [function_left_parentheses_linter] Remove spaces before the
#> stats::sd (c (x, y, z))
#>             ^
```

- **Tidyverse style**

`{lintr}` also provides linters to enforce the style used throughout the `{tidyverse}` (Wickham et al., 2019) ecosystem of R packages. This style of coding has been outlined in the tidyverse style guide (https://style.tidyverse.org/index.html).

```r
lint(
  text = "1:3 %>% mean %>% as.character",
  linters = pipe_call_linter()
)
#> <text>:1:9: warning: [pipe_call_linter] Use explicit calls in magrittr pipes, i
#> 1:3 %>% mean %>% as.character
#>         ^~~~
#> <text>:1:18: warning: [pipe_call_linter] Use explicit calls in magrittr pipes,
#> 1:3 %>% mean %>% as.character
#>                  ^~~~~~~~~~~
```

Benefits of using `{lintr}`

There are several benefits to using `{lintr}` to analyze and improve R code. One of the most obvious is that it can help users identify and fix problems in their code, which can save time and effort during the development process. By catching issues early on, `{lintr}` can help prevent bugs and other issues from creeping into code, which can save time and effort when it comes to debugging and testing.

Another benefit of `{lintr}` is that it can help users write more readable and maintainable code. By enforcing a consistent style and highlighting potential issues, `{lintr}` can help users write code that is easier to understand and work with. This is especially important for larger projects or teams, where multiple contributors may be working on the same codebase and it is important to ensure that code is easy to follow and understand.

Finally, `{lintr}` can be a useful tool for teaching and learning R. By providing feedback on code style and potential issues, it can help users learn good coding practices and improve their skills over time. This can be especially useful for beginners, who may not yet be familiar with all of the best practices for writing R code.

Conclusion

In conclusion, `{lintr}` is a valuable tool for R users to help improve the quality and reliability of their code. Its static code analysis capabilities, combined with its flexibility and ease of use, make it relevant and valuable for a wide range of applications.

Licensing and Availability

`{lintr}` is licensed under the MIT License, with all source code openly developed and stored on GitHub (https://github.com/r-lib/lintr), along with a corresponding issue tracker for bug reporting and feature enhancements.

Acknowledgments

References

McConnell, S. (2004). *Code complete*. Pearson Education.

R Core Team. (2023). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., … Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686