Package: flint (via r-universe)

September 7, 2024

Type Package

Title Find and Fix Lints in R Code

Version 0.0.8

Description Lints are code patterns that are not optimal because they are inefficient, forget corner cases, or less readable. 'flint' provides a small set of functions to detect those lints and automatically fix them. It builds on 'astgrepr', which itself uses the Rust crate 'ast-grep' to parse and navigate R code.

Depends R (>= 4.2)

Imports astgrepr, cli, crayon, data.table, digest, fs, yaml

Suggests glue, knitr, patrick, rex, rmarkdown, rstudioapi, testthat (>= 3.0.0), tibble, usethis, utils, withr

Remotes etiennebacher/astgrepr

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

URL https://flint.etiennebacher.com, https://github.com/etiennebacher/flint

BugReports https://github.com/etiennebacher/flint/issues

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Repository https://etiennebacher.r-universe.dev

RemoteUrl https://github.com/etiennebacher/flint

RemoteRef v0.0.8

RemoteSha eaa3c04b3a6a5c01ffe9dcf317598348fad510e3

Contents

`any_duplicated_linter` *Require usage of* `anyDuplicated(x) > 0` *over* `any(duplicated(x))`

Description

See https://lintr.r-lib.org/reference/any_duplicated_linter.

Usage

```
any_duplicated_linter
```

`any_is_na_linter` *Require usage of* `anyNA(x)` *over* `any(is.na(x))`

Description

See https://lintr.r-lib.org/reference/any_is_na_linter.

Usage

```
any_is_na_linter
```

`class_equals_linter` *Block comparison of class with* `==`

Description

See https://lintr.r-lib.org/reference/class_equals_linter.

Usage

```
class_equals_linter
```

`double_assignment_linter` *double_assignment*

Description

double_assignment

Usage

```
double_assignment_linter
```

`duplicate_argument_linter` *Duplicate argument linter*

Description

See https://lintr.r-lib.org/reference/duplicate_argument_linter.

Usage

```
duplicate_argument_linter
```

`empty_assignment_linter` *empty_assignment*

Description

empty_assignment

Usage

```
empty_assignment_linter
```

equals_na_linter *Equality check with NA linter*

Description

See https://lintr.r-lib.org/reference/equals_na_linter.

Usage

```
equals_na_linter
```

equal_assignment_linter *equal_assignment*

Description

equal_assignment

Usage

```
equal_assignment_linter
```

expect_comparison_linter *Require usage of* `expect_gt(x, y)` *over* `expect_true(x > y)` *(and similar)*

Description

See https://lintr.r-lib.org/reference/expect_comparison_linter.

Usage

```
expect_comparison_linter
```

---

expect_length_linter *Require usage of* expect_length(x, n) *over* expect_equal(length(x), n)

Description

See https://lintr.r-lib.org/reference/expect_length_linter.

Usage

```
expect_length_linter
```

---

expect_named_linter *Require usage of* expect_named(x, n) *over* expect_equal(names(x), n)

Description

See https://lintr.r-lib.org/reference/expect_named_linter.

Usage

```
expect_named_linter
```

---

expect_not_linter *Require usage of* expect_false(x) *over* expect_true(!x)

Description

See https://lintr.r-lib.org/reference/expect_not_linter.

Usage

```
expect_not_linter
```

---

expect_null_linter *Require usage of* expect_null *for checking* NULL

Description

See https://lintr.r-lib.org/reference/expect_null_linter.

Usage

```
expect_null_linter
```

expect_true_false_linter *Require usage of* `expect_true(x)` *over* `expect_equal(x, TRUE)`

Description

See https://lintr.r-lib.org/reference/expect_true_false_linter.

Usage

```
expect_true_false_linter
```

expect_type_linter *Require usage of* `expect_type(x, type)` *over* `expect_equal(typeof(x), type)`

Description

See https://lintr.r-lib.org/reference/expect_type_linter.

Usage

```
expect_type_linter
```

fix *Automatically replace lints*

Description

`fix()`, `fix_package()`, and `fix_dir()` all replace lints in files. The only difference is in the input they take:

- `fix()` takes path to files or directories
- `fix_dir()` takes a path to one directory
- `fix_package()` takes a path to the root of a package and looks at the following list of folders: `R`, `tests`, `inst`, `vignettes`, `data-raw`, `demo`, `exec`.

`fix_text()` takes some text input. Its main interest is to be able to quickly experiment with some lints and fixes.

Usage

```
fix(
  path = ".",
  linters = NULL,
  exclude_path = NULL,
  exclude_linters = NULL,
  force = FALSE,
  verbose = TRUE
)

fix_dir(
  path = ".",
  linters = NULL,
  exclude_path = NULL,
  exclude_linters = NULL
)

fix_package(
  path = ".",
  linters = NULL,
  exclude_path = NULL,
  exclude_linters = NULL
)

fix_text(text, linters = NULL, exclude_linters = NULL)
```

Arguments

path	A valid path to a file or a directory. Relative paths are accepted.
linters	A character vector with the names of the rules to apply. See the entire list of rules with `list_linters()`.
exclude_path	One or several paths that will be ignored from the `path` selection.
exclude_linters	One or several linters that will not be checked. Values can be the names of linters (such as `"any_is_na"`) or its associated function, such as `any_is_na_linter()` (this is mostly for compatibility with `lintr`).
force	Force the application of fixes on the files. This is used only in the case where Git is not detected, several files will be modified, and the code is run in a non-interactive setting.
verbose	Show messages.
text	Text to analyze (and to fix if necessary).

Ignoring lines

`flint` supports ignoring single lines of code with `# flint-ignore`. For example, this will not warn:

```
# flint-ignore
any(duplicated(x))
```

However, this will warn for the second `any(duplicated())`:

```
# flint-ignore
any(duplicated(x))
any(duplicated(y))
```

To ignore more than one line of code, use `# flint-ignore-start` and `# flint-ignore-end`:

```
# flint-ignore-start
any(duplicated(x))
any(duplicated(y))
# flint-ignore-end
```

Examples

```
# `fix_text()` is convenient to explore with a small example
fix_text("any(duplicated(rnorm(5)))")

fix_text("any(duplicated(rnorm(5)))
any(is.na(x))
")

# Setup for the example with `fix()`
destfile <- tempfile()
cat("
x = c(1, 2, 3)
any(duplicated(x), na.rm = TRUE)

any(duplicated(x))

if (any(is.na(x))) {
  TRUE
}

any(
  duplicated(x)
)", file = destfile)

fix(destfile)
cat(paste(readLines(destfile), collapse = "\n"))
```

`for_loop_index_linter` *Block usage of for loops directly overwriting the indexing variable*

Description

See https://lintr.r-lib.org/reference/for_loop_index_linter.

Usage

```
for_loop_index_linter
```

`function_return_linter`
Lint common mistakes/style issues cropping up from return statements

Description

See https://lintr.r-lib.org/reference/function_return_linter.

Usage

```
function_return_linter
```

`implicit_assignment_linter`
implicit_assignment

Description

implicit_assignment

Usage

```
implicit_assignment_linter
```

`is_numeric_linter` *Redirect* `is.numeric(x) || is.integer(x)` *to just use* `is.numeric(x)`

Description

See https://lintr.r-lib.org/reference/is_numeric_linter.

Usage

```
is_numeric_linter
```

lengths_linter *Require usage of* `lengths()` *where possible*

Description

See https://lintr.r-lib.org/reference/lengths_linter.

Usage

```
lengths_linter
```

length_levels_linter *Require usage of nlevels over length(levels(.))*

Description

See https://lintr.r-lib.org/reference/length_levels_linter.

Usage

```
length_levels_linter
```

length_test_linter *Check for a common mistake where length is applied in the wrong place*

Description

See https://lintr.r-lib.org/reference/length_test_linter.

Usage

```
length_test_linter
```

library_call_linter *Library call linter*

Description

See https://lintr.r-lib.org/reference/library_call_linter.

Usage

```
library_call_linter
```

`lint` *List all lints in a file or a directory*

Description

`lint()`, `lint_text()`, `lint_package()`, and `lint_dir()` all produce a data.frame containing the lints, their location, and potential fixes. The only difference is in the input they take:

- `lint()` takes path to files or directories
- `lint_text()` takes some text input
- `lint_dir()` takes a path to one directory
- `lint_package()` takes a path to the root of a package and looks at the following list of folders: `R`, `tests`, `inst`, `vignettes`, `data-raw`, `demo`, `exec`.

Usage

```
lint(
  path = ".",
  linters = NULL,
  exclude_path = NULL,
  exclude_linters = NULL,
  open = TRUE,
  use_cache = TRUE,
  verbose = TRUE
)

lint_dir(
  path = ".",
  linters = NULL,
  open = TRUE,
  exclude_path = NULL,
  exclude_linters = NULL,
  use_cache = TRUE,
  verbose = TRUE
)

lint_package(
  path = ".",
  linters = NULL,
  open = TRUE,
  exclude_path = NULL,
  exclude_linters = NULL,
  use_cache = TRUE,
  verbose = TRUE
)

lint_text(text, linters = NULL, exclude_linters = NULL)
```

Arguments

path	A valid path to a file or a directory. Relative paths are accepted.
linters	A character vector with the names of the rules to apply. See the entire list of rules with list_linters().
exclude_path	One or several paths that will be ignored from the path selection.
exclude_linters	One or several linters that will not be checked. Values can be the names of linters (such as "any_is_na") or its associated function, such as any_is_na_linter() (this is mostly for compatibility with lintr).
open	If TRUE (default) and if this is used in the RStudio IDE, lints will be shown with markers.
use_cache	Do not re-parse files that haven't changed since the last time this function ran.
verbose	Show messages.
text	Text to analyze.

Value

A dataframe where each row is a lint. The columns show the text, its location (both the position in the text and the file in which it was found) and the severity.

Ignoring lines

flint supports ignoring single lines of code with # flint-ignore. For example, this will not warn:

```
# flint-ignore
any(duplicated(x))
```

However, this will warn for the second any(duplicated()):

```
# flint-ignore
any(duplicated(x))
any(duplicated(y))
```

To ignore more than one line of code, use # flint-ignore-start and # flint-ignore-end:

```
# flint-ignore-start
any(duplicated(x))
any(duplicated(y))
# flint-ignore-end
```

Examples

```
# `lint_text()` is convenient to explore with a small example
lint_text("any(duplicated(rnorm(5)))")

lint_text("any(duplicated(rnorm(5)))
any(is.na(x))
")

# Setup for the example with `lint()`
destfile <- tempfile()
cat("
x = c(1, 2, 3)
any(duplicated(x), na.rm = TRUE)

any(duplicated(x))

if (any(is.na(x))) {
  TRUE
}

any(
  duplicated(x)
)", file = destfile)

lint(destfile)
```

`list_linters` *Get the list of linters in* `flint`

Description

Get the list of linters in `flint`

Usage

```
list_linters()
```

Value

A character vector

Examples

```
list_linters()
```

literal_coercion_linter

Require usage of correctly-typed literals over literal coercions

Description

See https://lintr.r-lib.org/reference/literal_coercion_linter.

Usage

```
literal_coercion_linter
```

matrix_apply_linter

Require usage of `colSums(x)` *or* `rowSums(x)` *over* `apply(x, ., sum)`

Description

See https://lintr.r-lib.org/reference/matrix_apply_linter.

Usage

```
matrix_apply_linter
```

missing_argument_linter

Missing argument linter

Description

See https://lintr.r-lib.org/reference/missing_argument_linter.

Usage

```
missing_argument_linter
```

nested_ifelse_linter

Block usage of nested `ifelse()` *calls*

Description

See https://lintr.r-lib.org/reference/nested_ifelse_linter.

Usage

```
nested_ifelse_linter
```

numeric_leading_zero_linter *Require usage of a leading zero in all fractional numerics*

Description

See https://lintr.r-lib.org/reference/numeric_leading_zero_linter.

Usage

```
numeric_leading_zero_linter
```

outer_negation_linter *Require usage of* `!any(x)` *over* `all(!x)`, `!all(x)` *over* `any(!x)`

Description

See https://lintr.r-lib.org/reference/outer_negation_linter.

Usage

```
outer_negation_linter
```

package_hooks_linter *Package hooks linter*

Description

See https://lintr.r-lib.org/reference/package_hooks_linter.

Usage

```
package_hooks_linter
```

paste_linter *Raise lints for several common poor usages of* `paste()`

Description

See https://lintr.r-lib.org/reference/paste_linter.

Usage

```
paste_linter
```

redundant_equals_linter

Block usage of `==`, `!=` *on logical vectors*

Description

See https://lintr.r-lib.org/reference/redundant_equals_linter.

Usage

```
redundant_equals_linter
```

redundant_ifelse_linter

Prevent `ifelse()` *from being used to produce* `TRUE/FALSE` *or* `1/0`

Description

See https://lintr.r-lib.org/reference/redundant_ifelse_linter.

Usage

```
redundant_ifelse_linter
```

right_assignment_linter

right_assignment

Description

right_assignment

Usage

```
right_assignment_linter
```

semicolon_linter *Semicolon linter*

Description

See https://lintr.r-lib.org/reference/semicolon_linter.

Usage

```
semicolon_linter
```

seq_linter *Sequence linter*

Description

See https://lintr.r-lib.org/reference/seq_linter.

Usage

```
seq_linter
```

setup_flint *Setup flint*

Description

This stores the default rules and internal files in `inst/flint`. It also imports `sgconfig.yml` that is used by `ast-grep`. This file must live at the root of the project and cannot be renamed.

Usage

```
setup_flint(path = ".")
```

Arguments

`path`	Path to package or project root.

Value

Imports files necessary for `flint` to work but doesn't return any value in R.

setup_flint_gha *Create a Github Actions workflow for* flint

Description

Create a Github Actions workflow for `flint`

Usage

```
setup_flint_gha(path = ".", overwrite = FALSE)
```

Arguments

`path`	Root path to the package.
`overwrite`	Whether to overwrite `.github/workflows/flint.yaml` if it already exists.

Value

Creates `.github/workflows/flint.yaml` but doesn't return any value.

sort_linter *Check for common mistakes around sorting vectors*

Description

See https://lintr.r-lib.org/reference/sort_linter.

Usage

```
sort_linter
```

todo_comment_linter *TODO comment linter*

Description

See https://lintr.r-lib.org/reference/todo_comment_linter.

Usage

```
todo_comment_linter
```

T_and_F_symbol_linter T *and* F *symbol linter*

Description

See https://lintr.r-lib.org/reference/T_and_F_symbol_linter.

Usage

```
T_and_F_symbol_linter
```

undesirable_function_linter *Undesirable function linter*

Description

See https://lintr.r-lib.org/reference/undesirable_function_linter.

Usage

```
undesirable_function_linter
```

undesirable_operator_linter *Undesirable operator linter*

Description

See https://lintr.r-lib.org/reference/undesirable_operator_linter.

Usage

```
undesirable_operator_linter
```

unnecessary_nesting_linter

Block instances of unnecessary nesting

Description

See https://lintr.r-lib.org/reference/unnecessary_nesting_linter.

Usage

```
unnecessary_nesting_linter
```

unreachable_code_linter

Block unreachable code and comments following return statements

Description

See https://lintr.r-lib.org/reference/unreachable_code_linter.

Usage

```
unreachable_code_linter
```

update_flint

Update the `flint` *setup*

Description

When `flint` is updated, it can ship new built-in rules. `update_flint()` will automatically add those new rules in the `flint/rules` folder. New rules are only determined by their names and rules that already exist in `flint/rules` are not affected.

For instance, if you added a custom rule `use_paste.yml`, then it will never be removed by `update_flint()`, even if `flint` later adds a built-in rule also named `use_paste.yml`.

Usage

```
update_flint(path = ".")
```

Arguments

`path`	Path to package or project root.

Value

Can add new files in the `flint/rules` folder, doesn't return anything.

Examples

```
## Not run:
  update_flint()

## End(Not run)
```

Index