

# Package ‘box.linters’

May 8, 2026

**Title** Linters for 'box' Modules

**Version** 0.10.7

**Description** Static code analysis of 'box' modules.

The package enhances code quality by providing linters that check for common issues, enforce best practices, and ensure consistent coding standards.

**URL** <https://appsilon.github.io/box.linters/>,  
<https://github.com/Appsilon/box.linters>

**License** LGPL-3

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Depends** R (>= 2.10)

**Imports** cli, fs, glue, lintr (>= 3.1.0), purrr, rlang, stringr, withr,  
xfun, xml2, xmlparsedata

**Suggests** box, covr, dplyr, knitr, prettycode, rcmdcheck, rmarkdown,  
R6, rex, rhino, shiny, spelling, testthat (>= 3.0.0),  
treesitter, treesitter.r (>= 1.1.0)

**Config/testthat/edition** 3

**Config/testthat/parallel** true

**Language** en-US

**NeedsCompilation** no

**Author** Ricardo Rodrigo Basa [aut, cre],  
Jakub Nowicki [aut],  
Mateusz Kołomański [ctb],  
Guilherme Vituri [ctb],  
Appsilon Sp. z o.o. [cph]

**Maintainer** Ricardo Rodrigo Basa <opensource+rodrigo@appsilon.com>

**Repository** CRAN

**Date/Publication** 2025-12-09 10:40:02 UTC

## Contents

box_alphabetical_calls_linter . . . . .	2
box_default_linters . . . . .	4
box_func_import_count_linter . . . . .	4
box_mod_fun_exists_linter . . . . .	5
box_pkg_fun_exists_linter . . . . .	6
box_separate_calls_linter . . . . .	7
box_trailing_commas_linter . . . . .	8
box_universal_import_linter . . . . .	9
box_unused_attached_mod_linter . . . . .	10
box_unused_attached_pkg_linter . . . . .	11
box_unused_att_mod_obj_linter . . . . .	13
box_unused_att_pkg_fun_linter . . . . .	14
box_usage_linter . . . . .	15
is_treesitter_installed . . . . .	17
namespaced_function_calls . . . . .	17
r6_usage_linter . . . . .	18
rhino_default_linters . . . . .	20
style_box_use_dir . . . . .	20
style_box_use_file . . . . .	21
style_box_use_text . . . . .	22
unused_declared_object_linter . . . . .	23
use_box_lintr . . . . .	24
<b>Index</b>	<b>26</b>

---

box\_alphabetical\_calls\_linter  
*box library alphabetical module and function imports linter*

---

### Description

Checks that module and function imports are sorted alphabetically. Aliases are ignored. The sort check is on package/module names and attached function names.

### Usage

```
box_alphabetical_calls_linter()
```

### Details

Alphabetical sort order places upper-case/capital letters first: (A, B, C, a, b, c).

For use in rhino, see the [Explanation: Rhino style guide](#) to learn about the details.

### Value

A custom linter function for use with `r-lib/lintr`.

**Examples**

```
# will produce lints
lintr::lint(
  text = "box::use(packageB, packageA)",
  linters = box_alphabetical_calls_linter()
)

lintr::lint(
  text = "box::use(package[functionB, functionA])",
  linters = box_alphabetical_calls_linter()
)

lintr::lint(
  text = "box::use(bslib, config, dplyr, DT)",
  linters = box_alphabetical_calls_linter()
)

lintr::lint(
  text = "box::use(path/to/B, path/to/A)",
  linters = box_alphabetical_calls_linter()
)

lintr::lint(
  text = "box::use(path/to/A[functionB, functionA])",
  linters = box_alphabetical_calls_linter()
)

lintr::lint(
  text = "box::use(path/to/A[alias = functionB, functionA])",
  linters = box_alphabetical_calls_linter()
)

# okay
lintr::lint(
  text = "box::use(packageA, packageB)",
  linters = box_alphabetical_calls_linter()
)

lintr::lint(
  text = "box::use(package[functionA, functionB])",
  linters = box_alphabetical_calls_linter()
)

lintr::lint(
  text = "box::use(DT, bslib, config, dplyr)",
  linters = box_alphabetical_calls_linter()
)

lintr::lint(
  text = "box::use(path/to/A, path/to/B)",
  linters = box_alphabetical_calls_linter()
)
```

```

lintr::lint(
  text = "box::use(path/to/A[functionA, functionB])",
  linters = box_alphabetical_calls_linter()
)

lintr::lint(
  text = "box::use(path/to/A[functionA, alias = functionB])",
  linters = box_alphabetical_calls_linter()
)

```

---

box\_default\_linters    *Box-compatible default linters*

---

### Description

A replacement for `lintr::object_usage_linter()` that works with box modules.

### Usage

```
box_default_linters
```

### Format

An object of class `list` of length 35.

### Examples

```

linters <- lintr::linters_with_defaults(defaults = box.linters::box_default_linters)

names(linters)

```

---

box\_func\_import\_count\_linter  
*box library function import count linter*

---

### Description

Checks that function imports do not exceed the defined max.

### Usage

```
box_func_import_count_linter(max = 8L)
```

## Arguments

`max` Maximum function imports allowed between [ and ]. Defaults to 8.

## Details

For use in rhino, see the [Explanation: Rhino style guide](#) to learn about the details.

## Value

A custom linter function for use with `r-lib/lintr`.

## Examples

```
# will produce lints
lintr::lint(
  text = "box::use(package[one, two, three, four, five, six, seven, eight, nine])",
  linters = box_func_import_count_linter()
)

lintr::lint(
  text = "box::use(package[one, two, three, four])",
  linters = box_func_import_count_linter(3)
)

# okay
lintr::lint(
  text = "box::use(package[one, two, three, four, five])",
  linters = box_func_import_count_linter()
)

lintr::lint(
  text = "box::use(package[one, two, three])",
  linters = box_func_import_count_linter(3)
)
```

---

box\_mod\_fun\_exists\_linter

*box library attached function exists and exported by called module linter*

---

## Description

Checks that functions being attached exist and are exported by the local module being called.

## Usage

```
box_mod_fun_exists_linter()
```

## Details

For use in rhino, see the [Explanation: Rhino style guide](#) to learn about the details.

## Value

A custom linter function for use with `r-lib/lintr`

## Examples

```
## Not run:
# will produce lint
lintr::lint(
  text = "box::use(path/to/module_a[function_not_exists],)",
  linter = box_mod_fun_exists_linter()
)

# okay
lintr::lint(
  text = "box::use(path/to/module_a[function_exists],)",
  linter = box_mod_fun_exists_linter()
)

## End(Not run)
```

---

box\_pkg\_fun\_exists\_linter

*box library attached function exists and exported by package linter*

---

## Description

Checks that functions being attached exist and are exported by the package/library being called.

## Usage

```
box_pkg_fun_exists_linter()
```

## Details

For use in rhino, see the [Explanation: Rhino style guide](#) to learn about the details.

## Value

A custom linter function for use with `r-lib/lintr`

**Examples**

```
# will produce lint
lintr::lint(
  text = "box::use(stringr[function_not_exists],)",
  linter = box_pkg_fun_exists_linter()
)

# okay
lintr::lint(
  text = "box::use(stringr[str_pad],)",
  linter = box_pkg_fun_exists_linter()
)
```

---

 box\_separate\_calls\_linter

*box library separate packages and module imports linter*

---

**Description**

Checks that packages and modules are imported in separate `box::use()` statements.

**Usage**

```
box_separate_calls_linter()
```

**Details**

For use in rhino, see the [Explanation: Rhino style guide](#) to learn about the details.

**Value**

A custom linter function for use with `r-lib/lintr`

**Examples**

```
# will produce lints
lintr::lint(
  text = "box::use(package, path/to/file)",
  linters = box_separate_calls_linter()
)

lintr::lint(
  text = "box::use(path/to/file, package)",
  linters = box_separate_calls_linter()
)

# okay
lintr::lint(
  text = "box::use(package1, package2)"
```

```

    box::use(path/to/file1, path/to/file2)",
    linters = box_separate_calls_linter()
  )

```

---

box\_trailing\_commas\_linter

*box library trailing commas linter*

---

## Description

Checks that all `box::use` imports have a trailing comma. This applies to package or module imports between `(` and `)`, and, optionally, function imports between `[` and `]`. Take note that `lintr::commas_linter()` may come into play.

## Usage

```
box_trailing_commas_linter(check_functions = FALSE)
```

## Arguments

`check_functions`

Boolean flag to include function imports between `[` and `]`. Defaults to `FALSE`.

## Details

For use in rhino, see the [Explanation: Rhino style guide](#) to learn about the details.

## Value

A custom linter function for use with `r-lib/lintr`

## Examples

```

# will produce lints
lintr::lint(
  text = "box::use(base, rlang)",
  linters = box_trailing_commas_linter()
)

lintr::lint(
  text = "box::use(
  dplyr[select, mutate]
)",
  linters = box_trailing_commas_linter()
)

# okay
lintr::lint(

```

```
text = "box::use(base, rlang, )",
linters = box_trailing_commas_linter()
)

lintr::lint(
  text = "box::use(
    dplyr[select, mutate],
  )",
  linters = box_trailing_commas_linter()
)
```

---

box\_universal\_import\_linter

*box library universal import linter*

---

### Description

Checks that all function imports are explicit. `package[...]` is not used.

### Usage

```
box_universal_import_linter()
```

### Details

For use in rhino, see the [Explanation: Rhino style guide](#) to learn about the details.

### Value

A custom linter function for use with `r-lib/lintr`

### Examples

```
# will produce lints
lintr::lint(
  text = "box::use(base[...])",
  linters = box_universal_import_linter()
)

lintr::lint(
  text = "box::use(path/to/file[...])",
  linters = box_universal_import_linter()
)

# okay
lintr::lint(
  text = "box::use(base[print])",
  linters = box_universal_import_linter()
)
```

```
lintr::lint(
  text = "box::use(path/to/file[do_something])",
  linters = box_universal_import_linter()
)
```

---

```
box_unused_attached_mod_linter
```

```
  box library unused attached module linter
```

---

### Description

Checks that all attached modules are used within the source file. This also covers modules attached using the . . . .

### Usage

```
box_unused_attached_mod_linter()
```

### Details

For use in rhino, see the [Explanation: Rhino style guide](#) to learn about the details.

### Value

A custom linter function for use with r-lib/lintr.

### Examples

```
## Not run:
# will produce lints
code <- "
box::use(
  path/to/module
)
"
```

```
lintr::lint(code, linters = box_unused_attached_mod_linter())
```

```
code <- "
box::use(
  alias = path/to/module
)
"
```

```
lintr::lint(code, linters = box_unused_attached_mod_linter())
```

```
code <- "
box::use(
```

```

    path/to/module[...]
  )
"

lintr::lint(code, linters = box_unused_attached_mod_linter())

# okay
code <- "
box::use(
  path/to/module
)

module$some_function()
"

lintr::lint(code, linters = box_unused_attached_mod_linter())

code <- "
box::use(
  alias = path/to/module
)

alias$some_function()
"

lintr::lint(code, linters = box_unused_attached_mod_linter())

code <- "
box::use(
  path/to/module[...] # module exports some_function()
)

some_function()
"

lintr::lint(code, linters = box_unused_attached_mod_linter())

## End(Not run)

```

---

```

box_unused_attached_pkg_linter
      box library unused attached package linter

```

---

### Description

Checks that all attached packages are used within the source file. This also covers packages attached using the ....

## Usage

```
box_unused_attached_pkg_linter()
```

## Details

For use in rhino, see the [Explanation: Rhino style guide](#) to learn about the details.

## Value

A custom linter function for use with `r-lib/lintr`.

## Examples

```
# will produce lints
code <- "
box::use(
  stringr
)
"

lintr::lint(text = code, linters = box_unused_attached_pkg_linter())

code <- "
box::use(
  alias = stringr
)
"

lintr::lint(text = code, linters = box_unused_attached_pkg_linter())

code <- "
box::use(
  stringr[...]
)
"

lintr::lint(text = code, linters = box_unused_attached_pkg_linter())

# okay
code <- "
box::use(
  stringr
)

stringr$str_pad()
"

lintr::lint(text = code, linters = box_unused_attached_pkg_linter())

code <- "
box::use(
  alias = stringr
```

```

)

alias$str_pad()
"

lintr::lint(text = code, linters = box_unused_attached_pkg_linter())

code <- "
box::use(
  stringr[...]
)

str_pad()
"

lintr::lint(text = code, linters = box_unused_attached_pkg_linter())

```

---

```

box_unused_att_mod_obj_linter
      box library unused attached module object linter

```

---

**Description**

Checks that all attached module functions and data objects are used within the source file.

**Usage**

```
box_unused_att_mod_obj_linter()
```

**Details**

For use in rhino, see the [Explanation: Rhino style guide](#) to learn about the details.

**Value**

A custom linter function for use with `r-lib/lintr`.

**Examples**

```

## Not run:
# will produce lints
code <- "
box::use(
  path/to/module[some_function, some_object],
)
"

lintr::lint(text = code, linters = box_unused_att_mod_obj_linter())

```

```

code <- "
box::use(
  path/to/module[alias_func = some_function, alias_obj = some_object],
)
"

lintr::lint(text = code, linters = box_unused_att_mod_obj_linter())

# okay
code <- "
box::use(
  path/to/module[some_function, some_object],
)

x <- sum(some_object)
some_function()
"

lintr::lint(text = code, linters = box_unused_att_mod_obj_linter())

code <- "
box::use(
  path/to/module[alias_func = some_function, alias_obj = some_object],
)

x <- sum(alias_obj)
alias_func()
"

lintr::lint(text = code, linters = box_unused_att_mod_obj_linter())

## End(Not run)

```

---

```
box_unused_att_pkg_fun_linter
```

```
  box library unused attached package function linter
```

---

### Description

Checks that all attached package functions are used within the source file.

### Usage

```
box_unused_att_pkg_fun_linter()
```

### Details

For use in rhino, see the [Explanation: Rhino style guide](#) to learn about the details.

**Value**

A custom linter function for use with `r-lib/lintr`.

**Examples**

```
# will produce lints
code <- "
box::use(
  stringr[str_pad],
)
"

lintr::lint(text = code, linters = box_unused_att_pkg_fun_linter())

code <- "
box::use(
  stringr[alias_func = str_pad],
)
"

lintr::lint(text = code, linters = box_unused_att_pkg_fun_linter())

# okay
code <- "
box::use(
  stringr[str_pad],
)

str_pad()
"

lintr::lint(text = code, linters = box_unused_att_pkg_fun_linter())

code <- "
box::use(
  stringr[alias_func = str_pad],
)

alias_func()
"

lintr::lint(text = code, linters = box_unused_att_pkg_fun_linter())
```

## Description

Checks that all function and data object calls made within a source file are valid. There are three ways for functions and data object calls to be come "valid". First is via base R packages. Second is via local declaration/definition. The third is via `box::use()` attachment.

## Usage

```
box_usage_linter()
```

## Details

For use in rhino, see the [Explanation: Rhino style guide](#) to learn about the details.

## Value

A custom linter function for use with `r-lib/lintr`.

## Examples

```
## Not run:
box::use(
  dplyr[">%>%`, filter, pull],
  stringr,
)

mpg <- mtcars %>%
  filter(mpg <= 10) %>%
  pull(mpg)

mpg <- mtcars %>%
  filter(mpg <= 10) %>%
  select(mpg)          # will lint

trimmed_string <- stringr$str_trim(" some string ")
trimmed_string <- stringr$str_trim(" some string ")    # will lint

existing_function <- function(x, y, z) {
  mean(c(x, y, z))
}

existing_function(1, 2, 3)
non_existing_function(1, 2, 3)    # will lint

average(1, 2, 3)    # will lint

## End(Not run)
```

---

`is_treesitter_installed`*Check if treesitter and dependencies are installed*

---

**Description**

Treesitter required R >= 4.3.0. Treesitter is required by a few `{box.linters}` functions.

**Usage**

```
is_treesitter_installed()
```

**Value**

Logical TRUE/FALSE if the treesitter dependencies exist.

**Examples**

```
## Not run:  
  
# Bare environment  
  
is_treesitter_installed()  
#> [1] FALSE  
  
install.packages(c("treesitter", "treesitter.r"))  
is_treesitter_installed()  
#> [1] TRUE  
  
## End(Not run)
```

---

`namespaced_function_calls`*Check that namespace::function() calls except for box::\*() are not made.*

---

**Description**

Check that `namespace::function()` calls except for `box::*()` are not made.

**Usage**

```
namespaced_function_calls(allow = NULL)
```

**Arguments**

`allow` Character vector of namespace or namespace::function to allow in the source code. Take note that the ( ) are not included. The box namespace will always be allowed

**Examples**

```
# will produce lints
code <- "box::use(package)
tidyr::pivot_longer()"

lintr::lint(text = code, linters = namespaced_function_calls())

## allow `tidyr::pivot_longer()`
code <- "box::use(package)
tidyr::pivot_longer()
tidyr::pivot_wider()"

lintr::lint(text = code, linters = namespaced_function_calls(allow = c("tidyr::pivot_longer")))

# okay
code <- "box::use(package)"

lintr::lint(text = code, linters = namespaced_function_calls())

## allow all `tidyr`
code <- "box::use(package)
tidyr::pivot_longer()
tidyr::pivot_wider()"

lintr::lint(text = code, linters = namespaced_function_calls(allow = c("tidyr")))

## allow `tidyr::pivot_longer()`
code <- "box::use(package)
tidyr::pivot_longer()"

lintr::lint(text = code, linters = namespaced_function_calls(allow = c("tidyr::pivot_longer")))
```

---

r6\_usage\_linter

*R6 class usage linter*


---

**Description**

Checks method and attribute calls within an R6 class. Covers public, private, and active objects. All internal calls should exist. All private methods and attributes should be used.

**Usage**

```
r6_usage_linter()
```

**Details**

For use in rhino, see the [Explanation: Rhino style guide](#) to learn about the details.

**Value**

A custom linter function for use with r-lib/lintr.

**Examples**

```
# will produce lints
code = "
box::use(
  R6[R6Class],
)

badClass <- R6Class('badClass',
  public = list(
    initialize = function() {
      private$not_exists()
    }
  ),
  private = list(
    unused_attribute = 'private data',
    unused_method = function() {
      self$attribute_not_exists
      self$function_not_exists()
    }
  )
)
"

lintr::lint(
  text = code,
  linters = r6_usage_linter()
)

# okay
code = "
box::use(
  R6[R6Class],
)

goodClass <- R6Class('goodClass',
  public = list(
    public_attr = NULL,
    initialize = function() {
      private$private_func()
    },
    some_function = function () {
      private$private_attr
    }
  ),
```

```

private = list(
  private_attr = 'private data',
  private_func = function() {
    self$public_attr
  }
)
)
"

lintr::lint(
  text = code,
  linters = r6_usage_linter()
)

```

---

rhino\_default\_linters *Rhino default linters*

---

### Description

See the [Explanation: Rhino style guide](#) to learn about the details.

### Usage

```
rhino_default_linters
```

### Format

An object of class `list` of length 40.

### Examples

```

linters <- lintr::linters_with_defaults(defaults = box.linters::rhino_default_linters)

names(linters)

```

---

style\_box\_use\_dir *Style the box::use() calls for a directory*

---

### Description

Style the `box::use()` calls for a directory

**Usage**

```

style_box_use_dir(
  path = ".",
  recursive = TRUE,
  exclude_files = c(),
  exclude_dirs = c("packrat", "renv"),
  indent_spaces = 2,
  trailing_commas_func = FALSE
)

```

**Arguments**

path	Path to a directory with files to style.
recursive	A logical value indicating whether or not files in sub-directories
exclude_files	A character vector of regular expressions to exclude files (not paths) from styling.
exclude_dirs	A character vector of directories to exclude.
indent_spaces	An integer scalar indicating tab width in units of spaces
trailing_commas_func	A boolean to activate adding a trailing comma to the end of the lists of functions to attach.

**Details**

Refer to [style\\_box\\_use\\_text\(\)](#) for styling details.

**Examples**

```

## Not run:
style_box_use_dir("path/to/dir")

# to exclude `__init__.R` files from styling
style_box_use_dir("path/to/dir", exclude_files = c("__init__\\.R"))

## End(Not run)

```

---

style\_box\_use\_file     *Style the box::use() calls of a source code*

---

**Description**

Style the box::use() calls of a source code

**Usage**

```

style_box_use_file(filename, indent_spaces = 2, trailing_commas_func = FALSE)

```

**Arguments**

filename        A file path to style.  
 indent\_spaces   An integer scalar indicating tab width in units of spaces  
 trailing\_commas\_func  
                  A boolean to activate adding a trailing comma to the end of the lists of functions to attach.

**Details**

Refer to [style\\_box\\_use\\_text\(\)](#) for styling details.

**Examples**

```
code <- "box::use(stringr[str_trim, str_pad], dplyr)"
file <- tempfile("style", fileext = ".R")
writeLines(code, file)

style_box_use_file(file)
```

---

style\_box\_use\_text        *Style the box::use() calls of source code text*

---

**Description**

Styles `box::use()` calls.

- All packages are called under one `box::use()`.
- All modules are called under one `box::use()`.
- Package and module levels are re-formatted to multiple lines. One package per line.
- Packages and modules are sorted alphabetically, ignoring the aliases.
- Functions attached in a single line retain the single line format.
- Functions attached in multiple lines retain the multiple line format.
- Functions are sorted alphabetically, ignoring the aliases.
- A trailing comma is added to packages, modules, and functions.

**Usage**

```
style_box_use_text(
  text,
  indent_spaces = 2,
  trailing_commas_func = FALSE,
  colored = getOption("styler.colored_print.vertical", default = FALSE),
  style = prettysyntax::default_style()
)
```

**Arguments**

text	Source code in text format
indent_spaces	Number of spaces per indent level
trailing_commas_func	A boolean to activate adding a trailing comma to the end of the lists of functions to attach.
colored	Boolean. For syntax highlighting using {prettycode}
style	A style from {prettycode}

**Examples**

```
code <- "box::use(stringr[str_trim, str_pad], dplyr)"

style_box_use_text(code)

code <- "box::use(stringr[
  str_trim,
  str_pad
],
shiny[...], # nolint
dplyr[alias = select, mutate], alias = tidy
path/to/module)
"

style_box_use_text(code)

style_box_use_text(code, trailing_commas_func = TRUE)
```

---

```
unused_declared_object_linter
```

*Unused declared function and data objects linter*

---

**Description**

Checks that all defined/declared functions and data objects are used within the source file. Functions and data objects that are marked with @export are ignored.

**Usage**

```
unused_declared_object_linter()
```

**Details**

For use in rhino, see the [Explanation: Rhino style guide](#) to learn about the details.

**Value**

A custom linter function for use with r-lib/lintr.

**Examples**

```
# will produce lint
code <- "
#' @export
public_function <- function() {

}

private_function <- function() {

}

local_data <- \"A\"
"

lintr::lint(text = code, linters = unused_declared_object_linter())

# okay
code <- "
#' @export
public_function <- function(local_data) {
  private_function(local_data)
}

private_function <- function() {

}

local_data <- \"A\"
"

lintr::lint(text = code, linters = unused_declared_object_linter())
```

---

use\_box\_lintr

*Use lintr with box.linters in your project*

---

**Description**

Create a minimal lintr config file with box modules support as a starting point for customization

**Usage**

```
use_box_lintr(path = ".", type = c("basic_box", "rhino"))
```

**Arguments**

path	Path to project root where a .lintr file should be created. If the .lintr file already exists, an error will be thrown.
type	The kind of configuration to create <ul style="list-style-type: none"><li>• <code>basic_box</code> creates a minimal lintr config based on the tidyverse configuration of lintr. This starts with <code>lintr::linters_with_defaults()</code> and is customized for box module compatibility</li><li>• <code>rhino</code> creates a lintr config based on the <a href="#">Rhino style guide</a></li></ul>

**Value**

Path to the generated configuration, invisibly.

**Examples**

```
## Not run:  
# use default box-compatible set of linters  
box.linters::use_box_lintr()  
  
# use `rhino` set of linters  
box.linters::use_box_lintr(type = "rhino")  
  
## End(Not run)
```

# Index

## \* datasets

- [box\\_default\\_linters](#), 4
- [rhino\\_default\\_linters](#), 20

- [box\\_alphabetical\\_calls\\_linter](#), 2
- [box\\_default\\_linters](#), 4
- [box\\_func\\_import\\_count\\_linter](#), 4
- [box\\_mod\\_fun\\_exists\\_linter](#), 5
- [box\\_pkg\\_fun\\_exists\\_linter](#), 6
- [box\\_separate\\_calls\\_linter](#), 7
- [box\\_trailing\\_commas\\_linter](#), 8
- [box\\_universal\\_import\\_linter](#), 9
- [box\\_unused\\_att\\_mod\\_obj\\_linter](#), 13
- [box\\_unused\\_att\\_pkg\\_fun\\_linter](#), 14
- [box\\_unused\\_attached\\_mod\\_linter](#), 10
- [box\\_unused\\_attached\\_pkg\\_linter](#), 11
- [box\\_usage\\_linter](#), 15

- [is\\_treesitter\\_installed](#), 17

- [namespaced\\_function\\_calls](#), 17

- [r6\\_usage\\_linter](#), 18
- [rhino\\_default\\_linters](#), 20

- [style\\_box\\_use\\_dir](#), 20
- [style\\_box\\_use\\_file](#), 21
- [style\\_box\\_use\\_text](#), 22
- [style\\_box\\_use\\_text\(\)](#), 21, 22

- [unused\\_declared\\_object\\_linter](#), 23
- [use\\_box\\_lintr](#), 24