Package 'styler'

June 26, 2018

Title Non-Invasive Pretty Printing of R Code

Version 1.0.2

Description Pretty-prints R code without changing the user's formatting intent.

Imports backports, cli, enc (>= 0.2), magrittr, purrr (>= 0.2.3), rematch2, rlang, rprojroot, tibble (>= 1.4.2), withr

Suggests data.tree, dplyr, here, knitr, rmarkdown, rstudioapi, testthat

License GPL-3

Encoding UTF-8

LazyData true

Date 2018-06-21

BugReports https://github.com/r-lib/styler/issues

URL https://github.com/r-lib/styler

RoxygenNote 6.0.1

VignetteBuilder knitr

Collate 'addins.R' 'communicate.R' 'compat-dplyr.R' 'compat-tidyr.R' 'expr-is.R' 'indent.R' 'initialize.R' 'nest.R' 'nested-to-tree.R' 'parse.R' 'reindent.R' 'token-define.R' 'relevel.R' 'rules-line-break.R' 'rules-other.R' 'rules-replacement.R' 'rules-spacing.R' 'serialize.R' 'set-assert-args.R' 'style-guides.R' 'styler.R' 'testing.R' 'token-create.R' 'transform-code.R' 'transform-files.R' 'ui.R' 'unindent.R' 'utils.R' 'vertical.R' 'visit.R' 'zzz.R'

NeedsCompilation no

Author Kirill Müller [aut], Lorenz Walthert [cre, aut]

Maintainer Lorenz Walthert <lorenz.walthert@icloud.com>

Repository CRAN

Date/Publication 2018-06-26 19:22:55 UTC

R topics documented:

styler-package . . . 2
create_style_guide . . . 3
has_crlf_as_first_line_sep . . . 4
math_token_spacing . . . 4
reindention . . . 5
style_dir . . . 6
style_file . . . 7
style_pkg . . . 9
style_text . . . 10
tidyverse_style . . . 11
two_cols_match . . . 12

Index **13**

`styler-package` *Non-invasive pretty printing of R code*

Description

styler allows you to format .R files, packages or entire R source trees according to a style guide. The following functions can be used for styling:

- `style_text()` to style a character vector.
- `style_file()` to style a single .R file.
- `style_dir()` to style all .R files in a directory.
- `style_pkg()` to style the source files of an R package.
- styler_addins (RStudio Addins) to style either selected code or the active file.

Author(s)

Maintainer: Lorenz Walthert <lorenz.walthert@icloud.com>

Authors:

- Kirill Müller <krlmlr+r@mailbox.org>

See Also

Useful links:

- https://github.com/r-lib/styler
- Report bugs at https://github.com/r-lib/styler/issues

Examples

```
style_text("call( 1)")
style_text("1    + 1", strict = FALSE)
style_text("a%>%b", scope = "spaces")
style_text("a%>%b; a", scope = "line_breaks")
style_text("a%>%b; a", scope = "tokens")
```

`create_style_guide` *Create a style guide*

Description

This is a helper function to create a style guide, which is technically speaking a named list of groups of transformer functions where each transformer function corresponds to one styling rule. The output of this function can be used as an argument for `style` in top level functions like `style_text()` and friends.

Usage

```
create_style_guide(initialize = default_style_guide_attributes,
  line_break = NULL, space = NULL, token = NULL, indention = NULL,
  use_raw_indention = FALSE, reindention = tidyverse_reindention())
```

Arguments

`initialize`	The bare name of a function that initializes various variables on each level of nesting.
`line_break`	A list of transformer functions that manipulate line_break information.
`space`	A list of transformer functions that manipulate spacing information.
`token`	A list of transformer functions that manipulate token text.
`indention`	A list of transformer functions that manipulate indention.
`use_raw_indention`	Boolean indicating whether or not the raw indention should be used.
`reindention`	A list of parameters for regex re-indention, most conveniently constructed using `specify_reindention()`.

Examples

```
set_line_break_before_curly_opening <- function(pd_flat) {
  op <- pd_flat$token %in% "'{'"
  pd_flat$lag_newlines[op] <- 1L
  pd_flat
}
set_line_break_before_curly_opening_style <- function() {
  create_style_guide(line_break = tibble::lst(set_line_break_before_curly_opening))
}
style_text("a <- function(x) { x }", style = set_line_break_before_curly_opening_style)
```

has_crlf_as_first_line_sep *Check if a string uses CRLF EOLs*

Description

Check if a string uses CRLF EOLs

Usage

```
has_crlf_as_first_line_sep(message, initial_text)
```

Arguments

message	A message returned with tryCatch().
initial_text	The inital text to style.

math_token_spacing *Specify spacing around math tokens*

Description

Helper function to create the input for the argument math_token_spacing in tidyverse_style().

Usage

```
specify_math_token_spacing(zero = "'^'", one = c("'+'", "'-'", "'*'",
  "'/'"))

tidyverse_math_token_spacing()
```

Arguments

zero	Character vector of tokens that should be surrounded with zero spaces.
one	Character vector with tokens that should be surrounded by at least one space (depending on strict = TRUE in the styling functions style_text() and friends). See 'Examples'.

Functions

- specify_math_token_spacing: Allows to fully specify the math token spacing.
- tidyverse_math_token_spacing: Simple forwarder to specify_math_token_spacing with spacing around math tokens according to the tidyverse style guide.

Examples

```
style_text(
  "1+1   -3",
  math_token_spacing = specify_math_token_spacing(zero = "'+'"),
  strict = FALSE
)
style_text(
  "1+1   -3",
  math_token_spacing = specify_math_token_spacing(zero = "'+'"),
  strict = TRUE
)
style_text(
  "1+1   -3",
  math_token_spacing = tidyverse_math_token_spacing(),
  strict = FALSE
)
style_text(
  "1+1   -3",
  math_token_spacing = tidyverse_math_token_spacing(),
  strict = TRUE
)
```

---

reindention *Specify what is re-indented how*

Description

This function returns a list that can be used as an input for the argument `reindention` of the function `tidyverse_style()`. It features sensible defaults, so the user can specify deviations from them conveniently without the need of setting all arguments explicitly.

Usage

```
specify_reindention(regex_pattern = NULL, indention = 0,
  comments_only = TRUE)

tidyverse_reindention()
```

Arguments

`regex_pattern`	Character vector with regular expression patterns that are to be re-indented with spaces, `NULL` if no reindention needed.
`indention`	The indention tokens should have if they match `regex_pattern`.
`comments_only`	Whether the `regex_reindention_pattern` should only be matched against comments or against all tokens. Mainly added for performance.

Functions

- `specify_reindention`: Allows to specify which tokens are reindented and how.
- `tidyverse_reindention`: Simple forwarder to `specify_reindention` with reindention according to the tidyverse style guide.

Examples

```
style_text("a <- xyz", reindention = specify_reindention(
  regex_pattern = "xyz", indention = 4, comments_only = FALSE)
)
style_text("a <- xyz", reindention = tidyverse_reindention())
```

---

style_dir *Prettify arbitrary R code*

Description

Performs various substitutions in all `.R` files in a directory. Carefully examine the results after running this function!

Usage

```
style_dir(path = ".", ..., style = tidyverse_style,
  transformers = style(...), filetype = "R", recursive = TRUE,
  exclude_files = NULL)
```

Arguments

`path`	Path to a directory with files to transform.
`...`	Arguments passed on to the `style` function.
`style`	A function that creates a style guide to use, by default `tidyverse_style()` (without the parentheses). Not used further except to construct the argument `transformers`. See `style_guides()` for details.
`transformers`	A set of transformer functions. This argument is most conveniently constructed via the `style` argument and `....` See 'Examples'.
`filetype`	Vector of file extensions indicating which file types should be styled. Case is ignored, and the `.` is optional, e.g. `c(".R", ".Rmd")` or `c("r", "rmd")`.
`recursive`	A logical value indicating whether or not files in subdirectories of `path` should be styled as well.
`exclude_files`	Character vector with paths to files that should be excluded from styling.

Value

Invisibly returns a data frame that indicates for each file considered for styling whether or not it was actually changed.

Warning

This function overwrites files (if styling results in a change of the code to be formatted). It is strongly suggested to only style files that are under version control or to create a backup copy.

We suggest to first style with `scope < "tokens"` and inspect and commit changes, because these changes are guaranteed to leave the abstract syntax tree (AST) unchanged. See section 'Roundtrip Validation' for details.

Then, we suggest to style with `scope = "tokens"` (if desired) and carefully inspect the changes to make sure the AST is not changed in an unexpected way that invalidates code.

Roundtrip Validation

The following section describes when and how styling is guaranteed to yield correct code.

If the style guide has `scope < "tokens"`, no tokens are changed and the abstract syntax tree (AST) should not change. Hence, it is possible to validate the styling by comparing whether the parsed expression before and after styling have the same AST. This comparison omits comments. styler compares error if the AST has changed through styling.

Note that with `scope = "tokens"` such a comparison is not conducted because the AST might well change and such a change is intended. There is no way styler can validate styling, that is why we inform the user to carefully inspect the changes.

See section 'Warning' for a good strategy to apply styling safely.

See Also

Other stylers: `style_file`, `style_pkg`, `style_text`, `styler_addins`

Examples

```
## Not run:
style_dir(file_type = "r")

## End(Not run)
```

`style_file` *Style* `.R` *and/or* `.Rmd` *files*

Description

Performs various substitutions in the files specified. Carefully examine the results after running this function!

Usage

```
style_file(path, ..., style = tidyverse_style, transformers = style(...))
```

Arguments

path	A character vector with paths to files to style.
...	Arguments passed on to the style function.
style	A function that creates a style guide to use, by default tidyverse_style() (without the parentheses). Not used further except to construct the argument transformers. See style_guides() for details.
transformers	A set of transformer functions. This argument is most conveniently constructed via the style argument and .... See 'Examples'.

Value

Invisibly returns a data frame that indicates for each file considered for styling whether or not it was actually changed.

Warning

This function overwrites files (if styling results in a change of the code to be formatted). It is strongly suggested to only style files that are under version control or to create a backup copy.

We suggest to first style with scope < "tokens" and inspect and commit changes, because these changes are guaranteed to leave the abstract syntax tree (AST) unchanged. See section 'Roundtrip Validation' for details.

Then, we suggest to style with scope = "tokens" (if desired) and carefully inspect the changes to make sure the AST is not changed in an unexpected way that invalidates code.

Roundtrip Validation

The following section describes when and how styling is guaranteed to yield correct code.

If the style guide has scope < "tokens", no tokens are changed and the abstract syntax tree (AST) should not change. Hence, it is possible to validate the styling by comparing whether the parsed expression before and after styling have the same AST. This comparison omits comments. styler compares error if the AST has changed through styling.

Note that with scope = "tokens" such a comparison is not conducted because the AST might well change and such a change is intended. There is no way styler can validate styling, that is why we inform the user to carefully inspect the changes.

See section 'Warning' for a good strategy to apply styling safely.

See Also

Other stylers: style_dir, style_pkg, style_text, styler_addins

Examples

```
# the following is identical but the former is more convenient:
file <- tempfile("styler", fileext = ".R")
enc::write_lines_enc("1++1", file)
style_file(file, style = tidyverse_style, strict = TRUE)
style_file(file, transformers = tidyverse_style(strict = TRUE))
```

```
enc::read_lines_enc(file)
unlink(file)
```

`style_pkg` *Prettify R source code*

Description

Performs various substitutions in all `.R` files in a package (code and tests). Carefully examine the results after running this function!

Usage

```
style_pkg(pkg = ".", ..., style = tidyverse_style,
  transformers = style(...), filetype = "R",
  exclude_files = "R/RcppExports.R")
```

Arguments

`pkg`	Path to a (subdirectory of an) R package.
`...`	Arguments passed on to the `style` function.
`style`	A function that creates a style guide to use, by default `tidyverse_style()` (without the parentheses). Not used further except to construct the argument `transformers`. See `style_guides()` for details.
`transformers`	A set of transformer functions. This argument is most conveniently constructed via the `style` argument and `...`. See 'Examples'.
`filetype`	Vector of file extensions indicating which file types should be styled. Case is ignored, and the `.` is optional, e.g. `c(".R", ".Rmd")` or `c("r", "rmd")`.
`exclude_files`	Character vector with paths to files that should be excluded from styling.

Warning

This function overwrites files (if styling results in a change of the code to be formatted). It is strongly suggested to only style files that are under version control or to create a backup copy.

We suggest to first style with `scope < "tokens"` and inspect and commit changes, because these changes are guaranteed to leave the abstract syntax tree (AST) unchanged. See section 'Roundtrip Validation' for details.

Then, we suggest to style with `scope = "tokens"` (if desired) and carefully inspect the changes to make sure the AST is not changed in an unexpected way that invalidates code.

Roundtrip Validation

The following section describes when and how styling is guaranteed to yield correct code.

If the style guide has `scope < "tokens"`, no tokens are changed and the abstract syntax tree (AST) should not change. Hence, it is possible to validate the styling by comparing whether the parsed expression before and after styling have the same AST. This comparison omits comments. styler compares error if the AST has changed through styling.

Note that with `scope = "tokens"` such a comparison is not conducted because the AST might well change and such a change is intended. There is no way styler can validate styling, that is why we inform the user to carefully inspect the changes.

See section 'Warning' for a good strategy to apply styling safely.

Value

Invisibly returns a data frame that indicates for each file considered for styling whether or not it was actually changed.

See Also

Other stylers: `style_dir`, `style_file`, `style_text`, `styler_addins`

Examples

```
## Not run:

style_pkg(style = tidyverse_style, strict = TRUE)
style_pkg(
  scope = "line_breaks",
  math_token_spacing = specify_math_token_spacing(zero = "'+'")
)

## End(Not run)
```

`style_text` *Style a string*

Description

Styles a character vector. Each element of the character vector corresponds to one line of code.

Usage

```
style_text(text, ..., style = tidyverse_style, transformers = style(...))
```

Arguments

text	A character vector with text to style.
...	Arguments passed on to the style function.
style	A function that creates a style guide to use, by default tidyverse_style() (without the parentheses). Not used further except to construct the argument transformers. See style_guides() for details.
transformers	A set of transformer functions. This argument is most conveniently constructed via the style argument and .... See 'Examples'.

See Also

Other stylers: style_dir, style_file, style_pkg, styler_addins

Examples

```
style_text("call( 1)")
style_text("1    + 1", strict = FALSE)
style_text("a%>%b", scope = "spaces")
style_text("a%>%b; a", scope = "line_breaks")
style_text("a%>%b; a", scope = "tokens")
# the following is identical but the former is more convenient:
style_text("a<-3++1", style = tidyverse_style, strict = TRUE)
style_text("a<-3++1", transformers = tidyverse_style(strict = TRUE))
```

---

tidyverse_style *The tidyverse style*

Description

Style code according to the tidyverse style guide.

Usage

```
tidyverse_style(scope = "tokens", strict = TRUE, indent_by = 2,
  start_comments_with_one_space = FALSE,
  reindention = tidyverse_reindention(),
  math_token_spacing = tidyverse_math_token_spacing())
```

Arguments

scope	The extent of manipulation. Can range from "none" (least invasive) to "token" (most invasive). See 'Details'. This argument is a vector of length one.
strict	A logical value indicating whether a set of strict or not so strict transformer functions should be returned. Compare the functions returned with or without strict = TRUE. For example, strict = TRUE means force *one* space e.g. after "," and *one* line break e.g. after a closing curly brace. strict = FALSE means to set spaces and line breaks to one if there is none and leave the code untouched otherwise. See 'Examples'.

indent_by — How many spaces of indention should be inserted after operators such as '('.

start_comments_with_one_space — Whether or not comments should start with only one space (see start_comments_with_space()).

reindention — A list of parameters for regex re-indention, most conveniently constructed using specify_reindention().

math_token_spacing — A list of parameters that define spacing around math token, conveniently constructed using specify_math_token_spacing().

Details

The following options for scope are available.

- "none": Performs no transformation at all.
- "spaces": Manipulates spacing between token on the same line.
- "indention": In addition to "spaces", this option also manipulates the indention level.
- "line_breaks": In addition to "indention", this option also manipulates line breaks.
- "tokens": In addition to "line_breaks", this option also manipulates tokens.

As it becomes clear from this description, more invasive operations can only be performed if all less invasive operations are performed too.

Examples

```
style_text("call( 1)", style = tidyverse_style, scope = "spaces")
style_text("call( 1)", transformers = tidyverse_style(strict = TRUE))
style_text(c("ab <- 3", "a  <-3"), strict = FALSE) # keeps alignment of "<-"
style_text(c("ab <- 3", "a  <-3"), strict = TRUE) # drops alignment of "<-"
```

---

two_cols_match *Check whether two columns match*

Description

Check whether two columns match

Usage

```
two_cols_match(col1, col2, data)
```

Arguments

col1, col2 — Column names as string.

data — The data frames that contains col1 and col2.

Index

create_style_guide, 3

has_crlf_as_first_line_sep, 4

math_token_spacing, 4

reindention, 5

specify_math_token_spacing
        *(*math_token_spacing*)*, 4
specify_math_token_spacing(), *12*
specify_reindention *(*reindention*)*, 5
specify_reindention(), *3*, *12*
start_comments_with_space(), *12*
style_dir, 6, *8*, *10, 11*
style_dir(), *2*
style_file, *7*, 7, *10, 11*
style_file(), *2*
style_guides(), *6*, *8, 9*, *11*
style_pkg, *7, 8*, 9, *11*
style_pkg(), *2*
style_text, *7, 8*, *10*, 10
style_text(), *2–4*
styler *(*styler-package*)*, 2
styler-package, 2
styler_addins, *2*, *7, 8*, *10, 11*

tidyverse_math_token_spacing
        *(*math_token_spacing*)*, 4
tidyverse_reindention *(*reindention*)*, 5
tidyverse_style, 11
tidyverse_style(), *4–6*, *8, 9*, *11*
two_cols_match, 12