

Package ‘styler’

November 20, 2018

Title Non-Invasive Pretty Printing of R Code

Version 1.1.0

Date 2018-11-20

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

License GPL-3

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

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

Imports backports, cli, magrittr, purrr (>= 0.2.3), rematch2, rlang, projroot, tibble (>= 1.4.2), tools, withr, xfun

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

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Collate 'addins.R' 'communicate.R' 'compat-dplyr.R' 'compat-tidyr.R' 'environments.R' 'expr-is.R' 'indent.R' 'initialize.R' 'io.R' 'nest.R' 'nested-to-tree.R' 'parse.R' 'reindent.R' 'token-define.R' 'relevel.R' 'roxygen-examples-add-remove.R' 'roxygen-examples-find.R' 'roxygen-examples-parse.R' 'roxygen-examples.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 2018-11-20 16:40:03 UTC

R topics documented:

styler-package	2
create_style_guide	3
math_token_spacing	4
print.vertical	5
reindentation	5
style_dir	6
style_file	7
style_pkg	9
style_text	11
tidyverse_style	12

Index	14
--------------	-----------

styler-package	<i>Non-invasive pretty printing of R code</i>
----------------	---

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

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

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)
```

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

<code>zero</code>	Character vector of tokens that should be surrounded with zero spaces.
<code>one</code>	Character vector with tokens that should be surrounded by at least one space (depending on <code>strict = TRUE</code> in the styling functions <code>style_text()</code> 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
)
```

print.vertical	<i>Print styled code</i>
----------------	--------------------------

Description

Print styled code

Usage

```
## S3 method for class 'vertical'
print(x, ...,
      colored = getOption("styler.colored_print.vertical"),
      style = prettycode::default_style())
```

Arguments

x	A character vector, one element corresponds to one line of code.
...	Not currently used.
colored	Whether or not the output should be colored with <code>prettycode::highlight()</code> .
style	Passed to <code>prettycode::highlight()</code> .

reindentation	<i>Specify what is re-indented how</i>
---------------	--

Description

This function returns a list that can be used as an input for the argument `reindentation` 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_reindentation(regex_pattern = NULL, indentation = 0,
                     comments_only = TRUE)

tidyverse_reindentation()
```

Arguments

regex_pattern	Character vector with regular expression patterns that are to be re-indented with spaces, NULL if no reindentation needed.
indentation	The indentation tokens should have if they match <code>regex_pattern</code> .
comments_only	Whether the <code>regex_reindentation_pattern</code> should only be matched against comments or against all tokens. Mainly added for performance.

Functions

- `specify_reindentation`: Allows to specify which tokens are reindented and how.
- `tidyverse_reindentation`: Simple forwarder to `specify_reindentation` with reindentation according to the tidyverse style guide.

Examples

```
style_text("a <- xyz", reindentation = specify_reindentation(
  regex_pattern = "xyz", indentation = 4, comments_only = FALSE
))
style_text("a <- xyz", reindentation = tidyverse_reindentation())
```

style_dir

Prettify arbitrary R code

Description

Performs various substitutions in all `.R`, `.Rmd` and/or `.Rnw` files in a directory (by default only `.R` files are styled - see `filetype` argument). Carefully examine the results after running this function!

Usage

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

Arguments

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

Examples

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

style_file

Style .R, .Rmd or .Rnw 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(...), include_roxygen_examples = TRUE)
```

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 <code>tidyverse_style()</code> (without the parentheses). Not used further except to construct the argument transformers. See <code>style_guides()</code> for details.
transformers	A set of transformer functions. This argument is most conveniently constructed via the style argument and <code>...</code> . See 'Examples'.
include_roxygen_examples	Whether or not to style code in roxygen examples.

Encoding

UTF-8 encoding is assumed. Please convert your code to UTF-8 if necessary before applying styler.

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")
xfun::write_utf8("1++1", file)
style_file(file, style = tidyverse_style, strict = TRUE)
style_file(file, transformers = tidyverse_style(strict = TRUE))
xfun::read_utf8(file)
unlink(file)
```

style_pkg

Prettify R source code

Description

Performs various substitutions in all .R files in a package (code and tests). One can also (optionally) style .Rmd and/or .Rnw files (vignettes and readme) by changing the `filetype` argument. Carefully examine the results after running this function!

Usage

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

Arguments

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

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	<i>Style a string</i>
------------	-----------------------

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(...), include_roxygen_examples = TRUE)
```

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'.
include_roxygen_examples	Whether or not to style code in roxygen 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	<i>The tidyverse style</i>
-----------------	----------------------------

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,
  reindentation = tidyverse_reindentation(),
  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 <code>strict = TRUE</code> . For example, <code>strict = TRUE</code> means force <i>one</i> space e.g. after <code>","</code> and <i>one</i> line break e.g. after a closing curly brace. <code>strict = FALSE</code> 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 indentation should be inserted after operators such as <code>'(</code> .
start_comments_with_one_space	Whether or not comments should start with only one space (see start_comments_with_space()).
reindentation	A list of parameters for regex re-indentation, most conveniently constructed using specify_reindentation() .
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.
- "indentation": In addition to "spaces", this option also manipulates the indentation level.
- "line_breaks": In addition to "indentation", 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 "<-"
```

Index

`create_style_guide`, 3

`math_token_spacing`, 4

`print.vertical`, 5

`reindentation`, 5

`specify_math_token_spacing`
 (`math_token_spacing`), 4

`specify_math_token_spacing()`, 12

`specify_reindentation` (`reindentation`), 5

`specify_reindentation()`, 3, 12

`start_comments_with_space()`, 12

`style_dir`, 6, 9–11

`style_dir()`, 2

`style_file`, 7, 7, 10, 11

`style_file()`, 2

`style_guides()`, 6, 8, 9, 11

`style_pkg`, 7, 9, 9, 11

`style_pkg()`, 2

`style_text`, 7, 9, 10, 11

`style_text()`, 2–4

`styler` (`styler-package`), 2

`styler-package`, 2

`styler_addins`, 2, 7, 9–11

`tidyverse_math_token_spacing`
 (`math_token_spacing`), 4

`tidyverse_reindentation` (`reindentation`), 5

`tidyverse_style`, 12

`tidyverse_style()`, 4–6, 8, 9, 11