Package ‘pillar’

November 20, 2020

Title Coloured Formatting for Columns
Version 1.4.7
Description Provides ‘pillar’ and ‘colonnade’ generics designed for formatting columns of data using the full range of colours provided by modern terminals.
License GPL-3
URL https://github.com/r-lib/pillar
BugReports https://github.com/r-lib/pillar/issues
Imports cli, crayon (>= 1.3.4), ellipsis, fansi, lifecycle, rlang (>= 0.3.0), utf8 (>= 1.1.0), vctrs (>= 0.2.0)
Suggests bit64, knitr, lubridate, testthat (>= 2.0.0), withr
Encoding UTF-8
LazyData true
RoxygenNote 7.1.1.9000
NeedsCompilation no
Author Kirill Müller [aut, cre], Hadley Wickham [aut], RStudio [cph]
Maintainer Kirill Müller <krlmlr+r@mailbox.org>
Repository CRAN
Date/Publication 2020-11-20 06:50:02 UTC

R topics documented:

pillar-package .......................................................... 2
align ................................................................. 3
colonnade ............................................................ 4
dim_desc ............................................................. 5
expect_known_display .................................................. 6
extra_cols ............................................................ 7
pillar-package

Description

Maturing Provides pillar and colonnade generics designed for formatting columns of data using the full range of colours provided by modern terminals.

Details

See pillar() for formatting a single column, and colonnade() for formatting multiple columns passed as a data frame.

Package options

- pillar.bold: Use bold font, e.g. for column headers? This currently defaults to FALSE, because many terminal fonts have poor support for bold fonts.
- pillar.subtle: Use subtle style, e.g. for row numbers and data types? Default: TRUE.
- pillar.subtle_num: Use subtle style for insignificant digits? Default: FALSE, is also affected by the pillar.subtle option.
- pillar.neg: Highlight negative numbers? Default: TRUE.
- pillar.sigfig: The number of significant digits that will be printed and highlighted, default: 3. Set the pillar.subtle option to FALSE to turn off highlighting of significant digits.
- pillar.min_title_chars: The minimum number of characters for the column title, default: 15. Column titles may be truncated up to that width to save horizontal space. Set to Inf to turn off truncation of column titles.
- pillar.min_chars: The minimum number of characters wide to display character columns, default: 0. Character columns may be truncated up to that width to save horizontal space. Set to Inf to turn off truncation of character columns.
**align**

**Author(s)**

Maintainer: Kirill Müller <krlmlr+r@mailbox.org>

Authors:

- Hadley Wickham

Other contributors:

- RStudio [copyright holder]

**See Also**

Useful links:

- [https://github.com/r-lib/pillar](https://github.com/r-lib/pillar)
- Report bugs at [https://github.com/r-lib/pillar/issues](https://github.com/r-lib/pillar/issues)

**Examples**

```
pillar(1:3)
pillar(c(1, 2, 3))
pillar(factor(letters[1:3]), title = "letters")
colonnade(iris[1:3, 1])
```

---

**align**

*Alignment helper*

**Description**

Facilitates easy alignment of strings within a character vector. Designed to help implementers of formatters for custom data types.

**Usage**

```
align(x, width = NULL, align = c("left", "right"))
```

**Arguments**

- **x**
  A character vector

- **width**
  The width that each string is padded to. If NULL, the maximum display width of the character vector is used (see `get_max_extent()`).

- **align**
  How should strings be aligned? If align = left then padding appears on the right, and vice versa.

**Examples**

```
align(c("abc", "de"), align = "left")
align(c("abc", "de"), align = "right")
```
Description

The vectors are formatted to fit horizontally into a user-supplied number of characters per row.

The colonnade() function doesn’t process the input but returns an object with a format() and a print() method. The implementations call squeeze() to create pillar objects and fit them to a given width.

The squeeze() function usually doesn’t need to be called manually. It returns an object suitable for printing and formatting at a fixed width with additional information about omitted columns, which can be retrieved via extra_cols().

Usage

colonnade(x, has_row_id = TRUE, width = NULL, ...)

squeeze(x, width = NULL, ...)

Arguments

x A list, which can contain matrices or data frames. If named, the names will be used as title for the pillars. Non-syntactic names will be escaped.

has_row_id Include a column indicating row IDs? Pass "*" to mark the row ID column with a star.

width Default width of the entire output, optional.

... These dots are for future extensions and must be empty.

Details

Pillars may be distributed over multiple tiers if width > getOption("width"). In this case each tier is at most getOption("width") characters wide. The very first step of formatting is to determine how many tiers are shown at most, and the width of each tier.

To avoid unnecessary computation for showing very wide colonnades, a first pass tries to fit all capitals into the tiers. For each pillar whose capital fits, it is then decided in which tier it is shown, if at all, and how much horizontal space it may use (either its minimum or its maximum width). Remaining space is then distributed proportionally to pillars that do not use their desired width.

For fitting pillars in one or more tiers, first a check is made if all pillars fit with their maximum width (e.g., option(tibble.width = Inf) or narrow colonnade). If yes, this is the resulting fit, no more work needs to be done. Otherwise, if the maximum width is too wide, the same test is carried out with the minimum width. If this is still too wide, this is the resulting fit. Otherwise, some tiers from the start will contain pillars with their maximum width, and the remaining tiers contain pillars with their minimum width. We determine the cut point where minimum and maximum assignment agree.
Fitting pillars into tiers is very similar to a word-wrapping algorithm. In a loop, new tiers are opened if the current tier overflows. If a column is too wide to fit a single tier, it will never be displayed, and the colonnade will be truncated there. This case should never occur with reasonable display widths larger than 30 characters. Truncation also happens if all available tiers are filled.

The remaining space is distributed from left to right. Each column gains space proportional to the fraction of missing and remaining space, rounded down. Any space remaining after rounding is distributed from left to right, one space per column.

Examples

```r
colonnade(list(a = 1:3, b = letters[1:3]))
long_string <- list(paste(letters, collapse = " "))
colonnade(long_string, width = 20)
colonnade(long_string, has_row_id = FALSE, width = 20)

# The width can also be overridden when calling format() or print():
print(colonnade(long_string), width = 20)

# If width is larger than getOption("width"), multiple tiers are created:
colonnade(rep(long_string, 4), width = Inf)
squeeze(colonnade(long_string), width = 20)
```

---

### dim_desc

**Format dimensions**

**Description**

Multi-dimensional objects are formatted as a x b x ..., for vectors the length is returned.

**Usage**

```r
dim_desc(x)
```

**Arguments**

- `x`: The object to format the dimensions for

**Examples**

```r
dim_desc(1:10)
dim_desc(Titanic)
```
**expect_known_display**  

*Test helpers*

**Description**

Expectation for packages that implement a data type with pillar support. Allows storing the desired result in a file, and comparing the output with the file contents. Note that this expectation sets options that affect the formatting of the pillar, see examples for usage.

**Usage**

```r
expect_known_display(object, file, ..., width = 80L, crayon = TRUE)
```

**Arguments**

- **object**: An object to check.
- **file**: File path where known value/output will be stored.
- **...**: Unused.
- **width**: The width of the output.
- **crayon**: Color the output?

**Examples**

```r
# The pillar is constructed after options have been set  
# (need two runs because reference file doesn't exist during the first run)
file <- tempfile("pillar", fileext = ".txt")
suppressWarnings(tryCatch(
  expect_known_display(pillar(1:3), file, crayon = FALSE),
  expectation_failure = function(e) {})
))
expect_known_display(pillar(1:3), file, crayon = FALSE)

# Good: Use tidyeval to defer construction
pillar_quo <- rlang::quo(pillar(1:3))
expect_known_display(!pillar_quo, file, crayon = FALSE)

## Not run:
# Bad: Options set in the active session may affect the display
integer_pillar <- pillar(1:3)
expect_known_display(integer_pillar, file, crayon = FALSE)

## End(Not run)
```
extra_cols

Retrieve information about columns that didn't fit the available width

Description
Formatting a colonnade object may lead to some columns being omitted due to width restrictions. This method returns a character vector that describes each of the omitted columns.

Usage
extra_cols(x, ...)

## S3 method for class 'pillar_squeezed_colonnade'
extra_cols(x, ..., n = Inf)

Arguments
x
The result of squeeze() on a colonnade object
...
Arguments passed to methods.
n
The number of extra columns to return; the returned vector will always contain as many elements as there are extra columns, but elements beyond n will be NA.

Examples
extra_cols(squeeze(colonnade(list(a = 1:3, b = 4:6), width = 8)))

format_type_sum
Format a type summary

Description
Called on values returned from type_sum() for defining the description in the capital. Implement this method for a helper class for custom formatting, and return an object of this helper class from your type_sum() implementation. See examples.

Usage
format_type_sum(x, width, ...)

## Default S3 method:
format_type_sum(x, width, ...)
get_extent

Arguments

x  A return value from type_sum()
width  The desired total width. If the returned string still is wider, it will be trimmed. Can be NULL.
...
Arguments passed to methods.

Examples

format_type_sum(1, NULL)
pillar(1)

type_sum.accel <- function(x) {
  structure("kg m/s^2", width = 8, class = "type_sum_accel")
}
format_type_sum.type_sum_accel <- function(x, width, ...) {
  if (!is.null(width) && width < attr(x, "width")) {
    x <- substr(x, 1, width)
  }
  style_subtle(x)
}
accel <- structure(9.81, class = "accel")
pillar(accel)

get_extent  Calculate display width

Description

get_extent() calculates the display width for each string in a character vector.
get_max_extent() calculates the maximum display width of all strings in a character vector, zero for empty vectors.

Usage

get_extent(x)

get_max_extent(x)

Arguments

x  A character vector.

Examples

get_extent(c("abc", "de"))
get_extent("\u904b\u6c23")
get_max_extent(c("abc", "de"))
**new_ornament**  
*Helper to define the contents of a pillar*

**Description**

This function is useful if your data renders differently depending on the available width. In this case, implement the `pillar_shaft()` method for your class to return a subclass of "pillar_shaft" and have the `format()` method for this subclass call `new_ornament()`. See the implementation of `pillar_shaft.numeric()` and `format.pillar_shaft_decimal()` for an example.

**Usage**

```r
call(new_ornament(x, width = NULL, align = NULL))
```

**Arguments**

- `x`: A character vector with formatting, see crayon
- `width`: An optional width of the resulting pillar, computed from `x` if missing
- `align`: Alignment, one of "left" or "right"

**Examples**

```r
call(new_ornament(c("abc", "de"), align = "right"))
```

---

**new_pillar_shaft**  
*Constructor for column data*

**Description**

**Stable**

The `new_pillar_shaft()` constructor creates objects of the "pillar_shaft" class. This is a virtual or abstract class, you must specify the `class` argument. By convention, this should be a string that starts with "pillar_shaft_". See vignette("extending",package = "tibble") for usage examples.

This method accepts a vector of arbitrary length and is expected to return an S3 object with the following properties:

- It has an attribute "width"
- It can have an attribute "min_width", if missing, "width" is used
- It must implement a method `format(x,width,...)` that can be called with any value between `min_width` and `width`
- This method must return an object that inherits from character and has attributes "align" (with supported values "left", "right", and "center") and "width"
The function `new_pillar_shaft()` returns such an object, and also correctly formats NA values. In many cases, the implementation of `pillar_shaft.your_class_name()` will format the data as a character vector (using color for emphasis) and simply call `new_pillar_shaft()`. See `pillar:::pillar_shaft.numeric` for a code that allows changing the display depending on the available width.

`new_pillar_shaft_simple()` provides an implementation of the `pillar_shaft` class suitable for output that has a fixed formatting, which will be truncated with a continuation character (ellipsis or ~) if it doesn’t fit the available width. By default, the required width is computed from the natural width of the `formatted` argument.

**Usage**

```r
new_pillar_shaft(
  x,
  ..., 
  width = NULL,  
  min_width = width,  
  class = NULL,  
  subclass = NULL
)

new_pillar_shaft_simple(
  formatted,  
  ..., 
  width = NULL, 
  align = "left",  
  min_width = NULL,  
  na = NULL,  
  na_indent = 0L
)
```

**Arguments**

- `x`: An object
- `...`: Additional attributes.
- `width`: The maximum column width.
- `min_width`: The minimum allowed column width, `width` if omitted.
- `class`: The name of the subclass.
- `subclass`: Deprecated, pass the `class` argument instead.
- `formatted`: An object coercible to `character`.
- `align`: Alignment of the column.
- `na`: String to use as NA value, defaults to "NA" styled with `style_na()` with fallback if color is not available.
- `na_indent`: Indentation of NA values.
new_pillar_title

Details
The formatted argument may also contain ANSI escapes to change color or other attributes of the text, see crayon.

Prepare a column title for formatting

Description
Call format() on the result to render column titles.

Usage
new_pillar_title(x, ...)

Arguments
x A character vector of column titles.
... These dots are for future extensions and must be empty.

Examples
format(new_pillar_title(names(iris)))

new_pillar_type

Prepare a column type for formatting

Description
Call format() on the result to render column types.

Usage
new_pillar_type(x, ...)

Arguments
x A vector for which the type is to be retrieved.
... These dots are for future extensions and must be empty.

Examples
format(new_pillar_type(iris$Species))
pillar

Format a vector suitable for tabular display

Description

Stable
pillar() formats a vector using one row for a title (if given), one row for the type, and length(x) rows for the data.

Usage

pillar(x, title = NULL, width = NULL, ...)

Arguments

- **x**: A vector to format.
- **title**: An optional title for the column. The title will be used "as is", no quoting will be applied.
- **width**: Default width, optional.
- **...**: Passed on to pillar_shaft().

Details

A pillar consists of a capital and a shaft.

The capital is constructed using the (currently internal) pillar_capital() function, which uses the title argument and calls type_sum() to format the type.

For the shaft, the pillar_shaft() generic is called with the object. The returned value is stored and processed with format() when displaying the pillar. The call to format() has a valid width argument. Depending on the implementation, the output representation can be computed eagerly right away (as done with new_pillar_shaft_simple()), or only when format() is called. The latter allows for adaptive shortening of the output depending on the available width, see pillar:::pillar_shaft.numeric for an example.

Examples

```r
x <- 123456789 * (10 ^ c(-1, -3, -5, NA, -8, -10))
pillar(x)
pillar(-x)
pillar(runif(10))
pillar(rcauchy(20))

# Special values are highlighted
pillar(c(runif(5), NA, NaN, Inf, -Inf))

# Very wide ranges will be displayed in scientific format
pillar(c(1e10, 1e-10), width = 20)
```
pillar(c(1e10, 1e-10))

x <- c(FALSE, NA, FALSE, FALSE, TRUE, FALSE, FALSE, TRUE, FALSE, TRUE)
pillar(x)

x <- c("This is string is rather long", NA, "?", "Short")
pillar(x)
pillar(x, width = 30)
pillar(x, width = 5)

date <- as.Date("2017-05-15")
pillar(date + c(1, NA, 3:5))
pillar(as.POSIXct(date) + c(30, NA, 600, 3600, 86400))

pillar_shaft

## S3 method for class 'pillar_shaft'
pillar(x, ...)  
## S3 method for class 'pillar_shaft'
print(x, width = NULL, ...)  
## S3 method for class 'pillar_shaft'
format(x, width, ...)  
## S3 method for class 'logical'
pillar_shaft(x, ...)  
## S3 method for class 'numeric'
pillar_shaft(x, ..., sigfig = NULL)  
## S3 method for class 'Date'
pillar_shaft(x, ...)

Description

Stable

Internal class for formatting the data for a column. pillar_shaft() is a coercion method that must be implemented for your data type to display it in a tibble.

This class comes with a default method for print() that calls format(). If print() is called without width argument, the natural width will be used when calling format(). Usually there's no need to implement this method for your subclass.

Your subclass must implement format(), the default implementation just raises an error. Your format() method can assume a valid value for the width argument.

Usage
pillar_shaft

## S3 method for class 'POSIXt'
pillar_shaft(x, ...)

## S3 method for class 'character'
pillar_shaft(x, ..., min_width = NULL)

## S3 method for class 'pillar_vertical'
pillar_shaft(x, ..., min_width = NULL, na_indent = 0L)

## S3 method for class 'list'
pillar_shaft(x, ...)

## S3 method for class 'factor'
pillar_shaft(x, ...)

## S3 method for class 'AsIs'
pillar_shaft(x, ...)

## Default S3 method:
pillar_shaft(x, ...)

Arguments

- **x** A vector to format
- **...** Arguments passed to methods.
- **width** Width for printing and formatting.
- **sigfig** Minimum number of significant figures to display, default: 3. Numbers larger than 1 will potentially show more significant figures than this.
- **min_width** Minimum number of characters to display, unless the string fits a shorter width.
- **na_indent** Indentation of NA values.

Details

The default method will currently format via `format()`, but you should not rely on this behavior.

Examples

```
pillar_shaft(1:3)
pillar_shaft(1.5:3.5)
pillar_shaft(NA)
pillar_shaft(c(1:3, NA))
```
**Description**

Functions that allow implementers of formatters for custom data types to maintain a consistent style with the default data types.

**Usage**

```r
style_num(x, negative, significant = rep_along(x, TRUE))
style_subtle(x)
style_subtle_num(x, negative)
style_bold(x)
style_na(x)
style_neg(x)
```

**Arguments**

- `x`: The character vector to style.
- `negative, significant`: Logical vector the same length as `x` that indicate if the values are negative and significant, respectively.

**Details**

- `style_subtle()` is affected by the `pillar.subtle` option.
- `style_subtle_num()` is affected by the `pillar.subtle_num` option, which is `FALSE` by default.
- `style_bold()` is affected by the `pillar.bold` option.
- `style_neg()` is affected by the `pillar.neg` option.

**See Also**

- `pillar-package` for a list of options

**Examples**

```r
style_num(
  c("123", "456"),
  negative = c(TRUE, FALSE)
)
style_num(
```
c("123", "456"),
negative = c(TRUE, FALSE),
significant = c(FALSE, FALSE)
)
style_subtle("text")
style_subtle_num(0.01 * 1:3, c(TRUE, FALSE, TRUE))
style_bold("Petal.Width")
style_na("NA")
style_neg("123")
Index

align, 3
character, 10
colonnade, 2, 4, 7
colonnade(), 2
crayon, 9, 11
dim_desc, 5
expect_known_display, 6
extra_cols, 7
extra_cols(), 4
format(), 4, 9, 11–14
format.pillar_shaft (pillar_shaft), 13
format_type_sum, 7
get_extent, 8
get_max_extent (get_extent), 8
get_max_extent(), 3
new_ornament, 9
new_pillar_shaft, 9
new_pillar_shaft(), 10
new_pillar_shaft_simple
  (new_pillar_shaft), 9
new_pillar_shaft_simple(), 12
new_pillar_title, 11
new_pillar_type, 11
pillar, 2, 4, 12
pillar(), 2
pillar-package, 2, 15
pillar_shaft, 13
pillar_shaft(), 9, 12
print(), 4, 13
print.pillar_shaft (pillar_shaft), 13
squeeze (colonnade), 4
squeeze(), 7
style_bold (style_num), 15
style_na (style_num), 15
style_na(), 10
style_neg (style_num), 15
style_num, 15
style_subtle (style_num), 15
style_subtle_num (style_num), 15
type_sum(), 7, 12