Package 'formatters'

September 12, 2024

Title ASCII Formatting for Values and Tables

Version 0.5.9

Date 2024-08-29

Description We provide a framework for rendering complex tables to ASCII, and a set of formatters for transforming values or sets of values into ASCII-ready display strings.

License Apache License 2.0

URL https://insightsengineering.github.io/formatters/, https://github.com/insightsengineering/formatters/

BugReports https://github.com/insightsengineering/formatters/issues

Depends methods, R (>= 2.10)

Imports checkmate (>= 2.1.0), grid, htmltools (>= 0.5.3), lifecycle (>= 0.2.0), stringi (>= 1.7.12)

Suggests dplyr (>= 1.0.9), gt (>= 0.10.0), huxtable (>= 2.0.0), knitr (>= 1.42), r2rtf (>= 0.3.2), rmarkdown (>= 2.23), testthat (>= 3.0.4), withr (>= 2.0.0)

VignetteBuilder knitr, rmarkdown

Config/Needs/verdepcheck mllg/checkmate, rstudio/htmltools, r-lib/lifecycle, tidyverse/dplyr, rstudio/gt, hughjonesd/huxtable, yihui/knitr, Merck/r2rtf, rstudio/rmarkdown, gagolews/stringi, r-lib/testthat, r-lib/withr

Config/Needs/website insightsengineering/nesttemplate

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.2

Collate 'data.R' 'format_value.R' 'matrix_form.R' 'generics.R' 'labels.R' 'mpf_exporters.R' 'package.R' 'page_size.R' 'pagination.R' 'tostring.R' 'utils.R' 'zzz.R'

NeedsCompilation no

Author Gabriel Becker [aut] (original creator of the package),
Adrian Waddell [aut],
Davide Garolini [ctb],
Emily de la Rua [ctb],
Abinaya Yogasekaram [ctb],
Joe Zhu [ctb, cre],
F. Hoffmann-La Roche AG [cph, fnd]

Maintainer Joe Zhu <joe.zhu@roche.com>

Repository CRAN

Date/Publication 2024-09-12 17:40:40 UTC

Contents

`basic_pagdf` *Basic/spoof pagination info data frame*

Description

Returns a minimal pagination info `data.frame` (with no info on siblings, footnotes, etc.).

Usage

```
basic_pagdf(
  rnames,
  labs = rnames,
  rnums = seq_along(rnames),
  extents = 1L,
  rclass = "DataRow",
  parent_path = NULL,
  paths = lapply(rnames, function(x) c(parent_path, x)),
  fontspec = font_spec()
)
```

Arguments

rnames	(character) vector of row names.
labs	(character) vector of row labels. Defaults to rnames.
rnums	(integer) vector of row numbers. Defaults to seq_along(rnames).
extents	(integer) number of lines each row requires to print. Defaults to 1 for all rows.
rclass	(character) class(es) for the rows. Defaults to "DataRow".
parent_path	(string) parent path that all rows should be "children of". Defaults to NULL, as usually this is not needed. It may be necessary to use "root", for some specific scenarios.
paths	(list) list of paths to the rows. Defaults to lapply(rnames, function(x) c(parent_path, x)).
fontspec	(font_spec) a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().

Value

A data.frame suitable for use in both the MatrixPrintForm constructor and the pagination machinery.

Examples

```
basic_pagdf(c("hi", "there"))
```

---

check_formats *Check if a format or alignment is supported*

Description

Utility functions for checking formats and alignments.

Usage

```
is_valid_format(x, stop_otherwise = FALSE)

check_aligns(algn)
```

Arguments

x	(string or function) format string or an object returned by sprintf_format()
stop_otherwise	(flag) whether an error should be thrown if x is not a valid format.
algn	(character) a character vector that indicates the requested cell alignments.

Value

- is_valid_format returns TRUE if x is NULL, a supported format string, or a function, and FALSE otherwise.
- check_aligns returns TRUE if the provided alignments are supported, otherwise, an error is thrown.

Note

If x is a function, no check is performed to verify that it returns a valid format.

Examples

```
is_valid_format("xx.x")
is_valid_format("fakeyfake")

check_aligns(c("decimal", "dec_right"))
```

---

decimal_align *Decimal alignment*

Description

Aligning decimal values of string matrix. Allowed alignments are: dec_left, dec_right, and decimal.

Usage

```
decimal_align(string_mat, align_mat)
```

Arguments

string_mat	(character matrix) "string" matrix component of MatrixPrintForm object.
align_mat	(character matrix) "aligns" matrix component of MatrixPrintForm object. Should contain either dec_left, dec_right, or decimal for values to be decimal aligned.

Details

Left and right decimal alignment (`dec_left` and `dec_right`) differ from center decimal alignment (`decimal`) only when there is padding present. This may occur if column widths are set wider via parameters `widths` in `toString` or `colwidths` in `paginate_*`. More commonly, it also occurs when column names are wider. Cell wrapping is not supported when decimal alignment is used.

Value

A processed string matrix of class `MatrixPrintForm` with decimal-aligned values.

See Also

`toString()`, `MatrixPrintForm()`

Examples

```
dfmf <- basic_matrix_form(mtcars[1:5, ])
aligns <- mf_aligns(dfmf)
aligns[, -c(1)] <- "dec_left"
decimal_align(mf_strings(dfmf), aligns)
```

---

default_horizontal_sep

Default horizontal separator

Description

The default horizontal separator character which can be displayed in the current charset for use in rendering table-like objects.

The default horizontal separator character which can be displayed in the current charset for use in rendering table-like objects.

Usage

```
default_hsep()

set_default_hsep(hsep_char)

default_hsep()

set_default_hsep(hsep_char)
```

Arguments

`hsep_char`	(`string`) character that will be set in the R environment options as the default horizontal separator. Must be a single character. Use `getOption("formatters_default_hsep")` to get its current value (`NULL` if not set).

Value

unicode 2014 (long dash for generating solid horizontal line) if in a locale that uses a UTF character set, otherwise an ASCII hyphen with a once-per-session warning.

unicode 2014 (long dash for generating solid horizontal line) if in a locale that uses a UTF character set, otherwise an ASCII hyphen with a once-per-session warning.

Examples

```
default_hsep()
set_default_hsep("o")
default_hsep()

default_hsep()
set_default_hsep("o")
default_hsep()
```

---

default_page_number *Default page number format*

Description

If set, the default page number string will appear on the bottom right of every page of a paginated table. The current cpp is used to position the string.

Usage

```
default_page_number()

set_default_page_number(page_number)
```

Arguments

page_number	(string) single string value to set the page number format. It should be formatted similarly to the following format: "page {i}/{n}". {i} will be replaced with the current page number, and {n} will be replaced with the total page number. Current cpp is used to position the string in the bottom right corner.

Value

The page number format string (NULL if not set).

Examples

```
default_page_number()
set_default_page_number("page {i} of {n}")
default_page_number()
```

divider_height *Divider height*

Description

Divider height

Usage

```
divider_height(obj)

## S4 method for signature 'ANY'
divider_height(obj)
```

Arguments

obj	(ANY) object.

Value

The height, in lines of text, of the divider between header and body. Currently returns 1L for the default method.

Examples

```
divider_height(mtcars)
```

DM *DM data*

Description

DM data

Usage

```
DM
```

Format

rds (data.frame)

do_forced_paginate *Generic for performing "forced" pagination*

Description

Forced pagination is pagination which happens regardless of position on page. The object is expected to have all information necessary to locate such page breaks, and the do_forced_pag method is expected to fully perform those paginations.

Usage

```
do_forced_paginate(obj)

## S4 method for signature 'ANY'
do_forced_paginate(obj)
```

Arguments

obj	(ANY) object to be paginated. The ANY method simply returns a list of length one, containing obj.

Value

A list of sub-objects, which will be further paginated by the standard pagination algorithm.

export_as_pdf *Export as PDF*

Description

The PDF output from this function is based on the ASCII output created with toString().

Usage

```
export_as_pdf(
  x,
  file,
  page_type = "letter",
  landscape = FALSE,
  pg_width = page_dim(page_type)[if (landscape) 2 else 1],
  pg_height = page_dim(page_type)[if (landscape) 1 else 2],
  width = lifecycle::deprecated(),
  height = lifecycle::deprecated(),
  margins = c(4, 4, 4, 4),
  min_siblings = 2,
```

```
  font_family = "Courier",
  font_size = 8,
  fontsize = font_size,
  lineheight = 1.2,
  paginate = TRUE,
  page_num = default_page_number(),
  lpp = NULL,
  cpp = NULL,
  hsep = "-",
  indent_size = 2,
  rep_cols = NULL,
  tf_wrap = TRUE,
  max_width = NULL,
  colwidths = NULL,
  fontspec = font_spec(font_family, font_size, lineheight),
  ttype_ok = FALSE
)
```

Arguments

x	(ANY) a table-like object to export. Must have an applicable `matrix_form` method.
file	(`string`) file to write to, must have `.pdf` extension.
page_type	(`string`) name of a page type. See `page_types`. Ignored when `pg_width` and `pg_height` are set directly.
landscape	(`flag`) whether the dimensions of `page_type` should be inverted for landscape orientation. Defaults to `FALSE`, ignored when `pg_width` and `pg_height` are set directly.
pg_width	(`numeric(1)`) page width in inches.
pg_height	(`numeric(1)`) page height in inches.
width	**[Deprecated]** Please use the `pg_width` argument or specify `page_type` instead.
height	**[Deprecated]** Please use the `pg_height` argument or specify `page_type` instead.
margins	(`numeric(4)`) the number of lines/characters of the margin on the bottom, left, top, and right sides of the page, respectively.
min_siblings	(`numeric`) minimum sibling rows which must appear on either side of pagination row for a mid-subtable split to be valid. Defaults to 2 for tables. It is automatically turned off (set to 0) for listings.
font_family	(`string`) name of a font family. An error will be thrown if the family named is not monospaced. Defaults to `"Courier"`.

font_size (numeric(1))
font size. Defaults to 12.

fontsize **[Deprecated]** Please use the font_size argument instead.

lineheight (numeric(1))
line height. Defaults to 1.

paginate (flag)
whether pagination should be performed. Defaults to TRUE if page size is specified (including the default).

page_num (string)
placeholder string for page numbers. See default_page_number for more information. Defaults to NULL.

lpp (numeric(1) or NULL)
lines per page. If NA (the default), this is calculated automatically based on the specified page size). NULL indicates no vertical pagination should occur.

cpp (numeric(1) or NULL)
width (in characters) per page. If NA (the default), this is calculated automatically based on the specified page size). NULL indicates no horizontal pagination should occur.

hsep (string)
character to repeat to create header/body separator line. If NULL, the object value will be used. If " ", an empty separator will be printed. See default_hsep() for more information.

indent_size (numeric(1))
indent size, in characters. Ignored when x is already a MatrixPrintForm object in favor of information there.

rep_cols (numeric(1))
number of *columns* (not including row labels) to be repeated on every page. Defaults to 0.

tf_wrap (flag)
whether the text for title, subtitles, and footnotes should be wrapped.

max_width (integer(1), string or NULL)
width that title and footer (including footnotes) materials should be word-wrapped to. If NULL, it is set to the current print width of the session (getOption("width")). If set to "auto", the width of the table (plus any table inset) is used. Parameter is ignored if tf_wrap = FALSE.

colwidths (numeric)
vector of column widths (in characters) for use in vertical pagination.

fontspec (font_spec)
a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().

ttype_ok (logical(1))
should truetype (non-monospace) fonts be allowed via fontspec. Defaults to FALSE. This parameter is primarily for internal testing and generally should not be set by end users.

Details

By default, pagination is performed with default cpp and lpp defined by specified page dimensions and margins. User-specified lpp and cpp values override this, and should be used with caution.

Title and footer materials are also word-wrapped by default (unlike when printed to the terminal), with cpp (as defined above) as the default max_width.

See Also

export_as_txt()

Examples

```
## Not run:
tf <- tempfile(fileext = ".pdf")
export_as_pdf(basic_matrix_form(mtcars), file = tf, pg_height = 4)

tf <- tempfile(fileext = ".pdf")
export_as_pdf(basic_matrix_form(mtcars), file = tf, lpp = 8)

## End(Not run)
```

---

export_as_rtf *Export as RTF*

Description

Experimental export to the rich text format (RTF) format.

Usage

```
export_as_rtf(
  x,
  file = NULL,
  colwidths = NULL,
  page_type = "letter",
  pg_width = page_dim(page_type)[if (landscape) 2 else 1],
  pg_height = page_dim(page_type)[if (landscape) 1 else 2],
  landscape = FALSE,
  margins = c(bottom = 0.5, left = 0.75, top = 0.5, right = 0.75),
  font_family = "Courier",
  font_size = 8,
  lineheight = 1,
  fontspec = font_spec(font_family, font_size, lineheight),
  ...
)
```

Arguments

x	(ANY) a table-like object to export. Must have an applicable matrix_form method.
file	(string or NULL) if non-NULL, the path to write a text file to containing x rendered as ASCII text.
colwidths	(numeric) vector of column widths (in characters) for use in vertical pagination.
page_type	(string) name of a page type. See page_types. Ignored when pg_width and pg_height are set directly.
pg_width	(numeric(1)) page width in inches.
pg_height	(numeric(1)) page height in inches.
landscape	(flag) whether the dimensions of page_type should be inverted for landscape orientation. Defaults to FALSE, ignored when pg_width and pg_height are set directly.
margins	(numeric(4)) named numeric vector containing "bottom", "left", "top", and "right" margins in inches. Defaults to .5 inches for both vertical margins and .75 for both horizontal margins.
font_family	(string) name of a font family. An error will be thrown if the family named is not monospaced. Defaults to "Courier".
font_size	(numeric(1)) font size. Defaults to 12.
lineheight	(numeric(1)) line height. Defaults to 1.
fontspec	(font_spec) a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().
...	additional parameters passed to paginate_to_mpfs().

Details

RTF export occurs via the following steps:

- The table is paginated to the specified page size (vertically and horizontally).
- Each separate page is converted to a MatrixPrintForm object and then to RTF-encoded text.
- Separate RTF text chunks are combined and written to a single RTF file.

Conversion of MatrixPrintForm objects to RTF is done via mpf_to_rtf().

`export_as_txt` *Export a table-like object to plain (ASCII) text with page breaks*

Description

This function converts `x` to a `MatrixPrintForm` object via `matrix_form()`, paginates it via `paginate_to_mpfs()`, converts each page to ASCII text via `toString()`, and outputs the strings, separated by `page_break`, to `file`.

Usage

```
export_as_txt(
  x,
  file = NULL,
  page_type = NULL,
  landscape = FALSE,
  pg_width = page_dim(page_type)[if (landscape) 2 else 1],
  pg_height = page_dim(page_type)[if (landscape) 1 else 2],
  font_family = "Courier",
  font_size = 8,
  lineheight = 1L,
  margins = c(top = 0.5, bottom = 0.5, left = 0.75, right = 0.75),
  paginate = TRUE,
  cpp = NA_integer_,
  lpp = NA_integer_,
  ...,
  hsep = NULL,
  indent_size = 2,
  tf_wrap = paginate,
  max_width = NULL,
  colwidths = NULL,
  min_siblings = 2,
  nosplitin = character(),
  rep_cols = NULL,
  verbose = FALSE,
  page_break = "\\s\\n",
  page_num = default_page_number(),
  fontspec = font_spec(font_family, font_size, lineheight),
  col_gap = 3
)
```

Arguments

`x`	(ANY) a table-like object to export. Must have an applicable `matrix_form` method.
`file`	(`string` or `NULL`) if non-`NULL`, the path to write a text file to containing `x` rendered as ASCII text.

page_type (string)
name of a page type. See page_types. Ignored when pg_width and pg_height are set directly.

landscape (flag)
whether the dimensions of page_type should be inverted for landscape orientation. Defaults to FALSE, ignored when pg_width and pg_height are set directly.

pg_width (numeric(1))
page width in inches.

pg_height (numeric(1))
page height in inches.

font_family (string)
name of a font family. An error will be thrown if the family named is not monospaced. Defaults to "Courier".

font_size (numeric(1))
font size. Defaults to 12.

lineheight (numeric(1))
line height. Defaults to 1.

margins (numeric(4))
named numeric vector containing "bottom", "left", "top", and "right" margins in inches. Defaults to .5 inches for both vertical margins and .75 for both horizontal margins.

paginate (flag)
whether pagination should be performed. Defaults to TRUE if page size is specified (including the default).

cpp (numeric(1) or NULL)
width (in characters) per page. If NA (the default), this is calculated automatically based on the specified page size). NULL indicates no horizontal pagination should occur.

lpp (numeric(1) or NULL)
lines per page. If NA (the default), this is calculated automatically based on the specified page size). NULL indicates no vertical pagination should occur.

... additional parameters passed to paginate_to_mpfs().

hsep (string)
character to repeat to create header/body separator line. If NULL, the object value will be used. If " ", an empty separator will be printed. See default_hsep() for more information.

indent_size (numeric(1))
indent size, in characters. Ignored when x is already a MatrixPrintForm object in favor of information there.

tf_wrap (flag)
whether the text for title, subtitles, and footnotes should be wrapped.

max_width (integer(1), string or NULL)
width that title and footer (including footnotes) materials should be word-wrapped to. If NULL, it is set to the current print width of the session (getOption("width")).

If set to "auto", the width of the table (plus any table inset) is used. Parameter is ignored if tf_wrap = FALSE.

colwidths (numeric)
vector of column widths (in characters) for use in vertical pagination.

min_siblings (numeric)
minimum sibling rows which must appear on either side of pagination row for a mid-subtable split to be valid. Defaults to 2 for tables. It is automatically turned off (set to 0) for listings.

nosplitin (character)
list of names of subtables where page breaks are not allowed, regardless of other considerations. Defaults to none.

rep_cols (numeric(1))
number of *columns* (not including row labels) to be repeated on every page. Defaults to 0.

verbose (flag)
whether additional informative messages about the search for pagination breaks should be shown. Defaults to FALSE.

page_break (string)
page break symbol (defaults to "\\n\\s").

page_num (string)
placeholder string for page numbers. See default_page_number for more information. Defaults to NULL.

fontspec (font_spec)
a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().

col_gap (numeric(1))
The number of spaces to be placed between columns in the rendered table (and assumed for horizontal pagination).

Details

If x has a num_rep_cols method, the value returned by it will be used for rep_cols by default. Otherwise, 0 will be used.

If x has an applicable do_forced_paginate method, it will be invoked during the pagination process.

Value

If file is NULL, the full paginated and concatenated string value is returned, otherwise the output is written to file and no value (invisible NULL) is returned.

Examples

```
export_as_txt(basic_matrix_form(mtcars), pg_height = 5, pg_width = 4)
```

ex_adsl *Simulated CDISC-like data for examples*

Description

Simulated CDISC-like data for examples

Usage

```
ex_adsl

ex_adae

ex_adaette

ex_adtte

ex_adcm

ex_adlb

ex_admh

ex_adqs

ex_adrs

ex_advs
```

Format

rds (data.frame)

An object of class tbl_df (inherits from tbl, data.frame) with 1934 rows and 48 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 1200 rows and 42 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 1200 rows and 42 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 1934 rows and 41 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 8400 rows and 59 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 1934 rows and 41 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 14000 rows and 49 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 2400 rows and 41 columns.

An object of class tbl_df (inherits from tbl, data.frame) with 16800 rows and 59 columns.

fmt_config *Format configuration*

Description

Format configuration

Usage

```
fmt_config(format = NULL, na_str = "NA", align = "center")
```

Arguments

format	(string or function) a format label (string) or formatter function.
na_str	(string) string that should be displayed in place of missing values.
align	(string) alignment values should be rendered with.

Value

An object of class fmt_config which contains the following elements:

- format
- na_str
- align

Examples

```
fmt_config(format = "xx.xx", na_str = "-", align = "left")
fmt_config(format = "xx.xx - xx.xx", align = "right")
```

font_spec *Font size specification*

Description

Font size specification

Usage

```
font_spec(font_family = "Courier", font_size = 8, lineheight = 1)
```

Arguments

font_family	(character(1)) font family to use during string width and lines-per-page calculations. You can specify "Times New Roman" as "Times" or "serif", regardless of OS. Beyond that, see family entry in graphics::par() for details.
font_size	(numeric(1)) font size to use during string width calculations and lines-per-page calculations.
lineheight	(numeric(1)) line height to use during lines-per-page calculations.

Details

Passing the output of this constructor to the rendering or pagination machinery defines a font for use when calculating word wrapping and pagination.

Note

Specifying font in this way to, e.g., export_as_txt() or toString() will not affect the font size of the output, as these are both raw text formats. export_as_pdf() will use the specified font.

See Also

nchar_ttype(), toString(), pagination_algo, export_as_pdf()

Examples

```
fspec <- font_spec("Courier", 8, 1)

lets <- paste(letters, collapse = "")

nchar_ttype(lets, fspec)

fspec2 <- font_spec("Times", 8, 1)

nchar_ttype(lets, fspec2)
```

---

format_value *Converts a (possibly compound) value into a string using the* format *information*

Description

Converts a (possibly compound) value into a string using the format information

Usage

```
format_value(x, format = NULL, output = c("ascii", "html"), na_str = "NA")
```

Arguments

x	(ANY) the value to be formatted.
format	(string or function) the format label (string) or formatter function to apply to x.
output	(string) output type.
na_str	(character) character vector to display when the values of x are missing. If only one string is provided, it is applied for all missing values. Defaults to "NA".

Details

A length-zero value for na_str will be interpreted as "NA".

Value

Formatted text representing the cell x.

See Also

round_fmt()

Examples

```
x <- format_value(pi, format = "xx.xx")
x

format_value(x, output = "ascii")

# na_str works with multiple values
format_value(c(NA, 1, NA), format = "xx.x (xx.x - xx.x)", na_str = c("NE", "<missing>"))
```

---

ifnotlen0 %||% *(if length-0) alternative operator*

Description

%||% (if length-0) alternative operator

Usage

```
a %||% b
```

Arguments

a	(ANY) element to select *only* if it is not of length 0.
b	(ANY) element to select if a has length 0.

Value

a if it is not of length 0, otherwise b.

Examples

```
6 %||% 10

character() %||% "hi"

NULL %||% "hi"
```

---

is.wholenumber *Check if a value is a whole number*

Description

Check if a value is a whole number

Usage

```
is.wholenumber(x, tol = .Machine$double.eps^0.5)
```

Arguments

x	(numeric(1)) a numeric value.
tol	(numeric(1)) a precision tolerance.

Value

TRUE if x is within tol of zero, FALSE otherwise.

Examples

```
is.wholenumber(5)
is.wholenumber(5.00000000000000001)
is.wholenumber(.5)
```

lab_name *Label, name, and format accessor generics*

Description

Getters and setters for basic, relatively universal attributes of "table-like" objects.

Usage

```
obj_name(obj)

obj_name(obj) <- value

obj_label(obj)

obj_label(obj) <- value

## S4 method for signature 'ANY'
obj_label(obj)

## S4 replacement method for signature 'ANY'
obj_label(obj) <- value

obj_format(obj)

## S4 method for signature 'ANY'
obj_format(obj)

## S4 method for signature 'fmt_config'
obj_format(obj)

obj_format(obj) <- value

## S4 replacement method for signature 'ANY'
obj_format(obj) <- value

## S4 replacement method for signature 'fmt_config'
obj_format(obj) <- value

obj_na_str(obj)

## S4 method for signature 'ANY'
obj_na_str(obj)

## S4 method for signature 'fmt_config'
obj_na_str(obj)
```

```
obj_na_str(obj) <- value

## S4 replacement method for signature 'ANY'
obj_na_str(obj) <- value

## S4 replacement method for signature 'fmt_config'
obj_na_str(obj) <- value

obj_align(obj)

## S4 method for signature 'ANY'
obj_align(obj)

## S4 method for signature 'fmt_config'
obj_align(obj)

obj_align(obj) <- value

## S4 replacement method for signature 'ANY'
obj_align(obj) <- value

## S4 replacement method for signature 'fmt_config'
obj_align(obj) <- value
```

Arguments

obj	(ANY) the object.
value	character(1). The new label

Value

The name, format, or label of `obj` for getters, or `obj` after modification for setters.

See Also

with_label

`list_formats` *List of currently supported formats and vertical alignments*

Description

We support `xx` style format labels grouped by 1d, 2d, and 3d. Currently valid format labels cannot be added dynamically. Format functions must be used for special cases.

Usage

```
list_valid_format_labels()

list_valid_aligns()
```

Value

- `list_valid_format_labels()` returns a nested list, with elements listing the supported 1d, 2d, and 3d format strings.
- `list_valid_aligns()` returns a character vector of valid vertical alignments.

Examples

```
list_valid_format_labels()

list_valid_aligns()
```

---

main_title *General title and footer accessors*

Description

General title and footer accessors

Usage

```
main_title(obj)

## S4 method for signature 'MatrixPrintForm'
main_title(obj)

main_title(obj) <- value

## S4 replacement method for signature 'MatrixPrintForm'
main_title(obj) <- value

subtitles(obj)

## S4 method for signature 'MatrixPrintForm'
subtitles(obj)

subtitles(obj) <- value

## S4 replacement method for signature 'MatrixPrintForm'
subtitles(obj) <- value
```

```
page_titles(obj)

## S4 method for signature 'MatrixPrintForm'
page_titles(obj)

## S4 method for signature 'ANY'
page_titles(obj)

page_titles(obj) <- value

## S4 replacement method for signature 'MatrixPrintForm'
page_titles(obj) <- value

main_footer(obj)

## S4 method for signature 'MatrixPrintForm'
main_footer(obj)

main_footer(obj) <- value

## S4 replacement method for signature 'MatrixPrintForm'
main_footer(obj) <- value

prov_footer(obj)

## S4 method for signature 'MatrixPrintForm'
prov_footer(obj)

prov_footer(obj) <- value

## S4 replacement method for signature 'MatrixPrintForm'
prov_footer(obj) <- value

all_footers(obj)

all_titles(obj)
```

Arguments

obj	(ANY) object to extract information from.
value	character. New value.

Value

A character scalar (main_title), character vector (main_footer), or vector of length zero or more (subtitles, page_titles, prov_footer) containing the relevant title/footer contents.

make_row_df *Make row layout summary data frames for use during pagination*

Description

All relevant information about table rows (e.g. indentations) is summarized in a `data.frame`. This function works **only** on `rtables` and `rlistings` objects, and not on their `print` counterparts (like `MatrixPrintForm`).

Usage

```
make_row_df(
  tt,
  colwidths = NULL,
  visible_only = TRUE,
  rownum = 0,
  indent = 0L,
  path = character(),
  incontent = FALSE,
  repr_ext = 0L,
  repr_inds = integer(),
  sibpos = NA_integer_,
  nsibs = NA_integer_,
  max_width = NULL,
  fontspec = font_spec(),
  col_gap = 3L
)

## S4 method for signature 'MatrixPrintForm'
make_row_df(
  tt,
  colwidths = NULL,
  visible_only = TRUE,
  rownum = 0,
  indent = 0L,
  path = character(),
  incontent = FALSE,
  repr_ext = 0L,
  repr_inds = integer(),
  sibpos = NA_integer_,
  nsibs = NA_integer_,
  max_width = NULL,
  fontspec = font_spec(),
  col_gap = mf_colgap(tt) %||% 3L
)
```

Arguments

tt	(ANY) object representing the table-like object to be summarized.
colwidths	(numeric) internal detail, do not set manually.
visible_only	(flag) should only visible aspects of the table structure be reflected in this summary. Defaults to TRUE. May not be supported by all methods.
rownum	(numeric(1)) internal detail, do not set manually.
indent	(integer(1)) internal detail, do not set manually.
path	(character) path to the (sub)table represented by tt. Defaults to character().
incontent	(flag) internal detail, do not set manually.
repr_ext	(integer(1)) internal detail, do not set manually.
repr_inds	(integer) internal detail, do not set manually.
sibpos	(integer(1)) internal detail, do not set manually.
nsibs	(integer(1)) internal detail, do not set manually.
max_width	(numeric(1) or NULL) maximum width for title/footer materials.
fontspec	(font_spec) a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().
col_gap	(numeric(1)) the gap to be assumed between columns, in number of spaces with font specified by fontspec.

Details

When visible_only is TRUE (the default), methods should return a data.frame with exactly one row per visible row in the table-like object. This is useful when reasoning about how a table will print, but does not reflect the full pathing space of the structure (though the paths which are given will all work as is).

If supported, when visible_only is FALSE, every structural element of the table (in row-space) will be reflected in the returned data.frame, meaning the full pathing-space will be represented but some rows in the layout summary will not represent printed rows in the table as it is displayed.

Most arguments beyond tt and visible_only are present so that make_row_df methods can call make_row_df recursively and retain information, and should not be set during a top-level call.

Value

A `data.frame` of row/column-structure information used by the pagination machinery.

Note

The technically present root tree node is excluded from the summary returned by both `make_row_df` and `make_col_df` (see relevant functions in`rtables`), as it is the row/column structure of `tt` and thus not useful for pathing or pagination.

Examples

```
# Expected error with matrix_form. For real case examples consult {rtables} documentation
mf <- basic_matrix_form(iris)
# make_row_df(mf) # Use table obj instead
```

MatrixPrintForm *Constructor for Matrix Print Form*

Description

Constructor for `MatrixPrintForm`, an intermediate representation for ASCII table printing.

Usage

```
MatrixPrintForm(
  strings = NULL,
  spans,
  aligns,
  formats,
  row_info,
  colpaths = NULL,
  line_grouping = seq_len(NROW(strings)),
  ref_fnotes = list(),
  nlines_header,
  nrow_header,
  has_topleft = TRUE,
  has_rowlabs = has_topleft,
  expand_newlines = TRUE,
  main_title = "",
  subtitles = character(),
  page_titles = character(),
  listing_keycols = NULL,
  main_footer = "",
  prov_footer = character(),
  header_section_div = NA_character_,
  horizontal_sep = default_hsep(),
```

```
  col_gap = 3,
  table_inset = 0L,
  colwidths = NULL,
  indent_size = 2,
  fontspec = font_spec(),
  rep_cols = 0L
)
```

Arguments

strings	(character matrix) matrix of formatted, ready-to-display strings organized as they will be positioned when rendered. Elements that span more than one column must be followed by the correct number of placeholders (typically either empty strings or repeats of the value).
spans	(numeric matrix) matrix of same dimension as strings giving the spanning information for each element. Must be repeated to match placeholders in strings.
aligns	(character matrix) matrix of same dimension as strings giving the text alignment information for each element. Must be repeated to match placeholders in strings. Must be a supported text alignment. See decimal_align for allowed values.
formats	(matrix) matrix of same dimension as strings giving the text format information for each element. Must be repeated to match placeholders in strings.
row_info	(data.frame) data frame with row-information necessary for pagination (see basic_pagdf() for more details).
colpaths	(list or NULL) NULL, or a list of paths to each leaf column, for use during horizontal pagination.
line_grouping	(integer) sequence of integers indicating how print lines correspond to semantic rows in the object. Typically this should not be set manually unless expand_newlines is set to FALSE.
ref_fnotes	(list) referential footnote information, if applicable.
nlines_header	(numeric(1)) number of lines taken up by the values of the header (i.e. not including the divider).
nrow_header	(numeric(1)) number of *rows* corresponding to the header.
has_topleft	(flag) does the corresponding table have "top left information" which should be treated differently when expanding newlines. Ignored if expand_newlines is FALSE.

has_rowlabs (flag)
do the matrices (strings, spans, aligns) each contain a column that corresponds with row labels (rather than with table cell values). Defaults to TRUE.

expand_newlines (flag)
whether the matrix form generated should expand rows whose values contain newlines into multiple 'physical' rows (as they will appear when rendered into ASCII). Defaults to TRUE.

main_title (string)
main title as a string.

subtitles (character)
subtitles, as a character vector.

page_titles (character)
page-specific titles, as a character vector.

listing_keycols (character)
. if matrix form of a listing, this contains the key columns as a character vector.

main_footer (character)
main footer, as a character vector.

prov_footer (character)
provenance footer information, as a character vector.

header_section_div (string)
divider to be used between header and body sections.

horizontal_sep (string)
horizontal separator to be used for printing divisors between header and table body and between different footers.

col_gap (numeric(1))
space (in characters) between columns.

table_inset (numeric(1))
table inset. See table_inset().

colwidths (numeric or NULL)
column rendering widths. If non-NULL, must have length equal to ncol(strings).

indent_size (numeric(1))
number of spaces to be used per level of indent (if supported by the relevant method). Defaults to 2.

fontspec (font_spec)
a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().

rep_cols (numeric(1))
number of columns to be repeated as context during horizontal pagination.

Value

An object of class `MatrixPrintForm`. Currently this is implemented as an S3 class inheriting from list with the following elements:

`strings` see argument.

`spans` see argument.

`aligns` see argument.

`display` logical matrix of same dimension as `strings` that specifies whether an element in `strings` will be displayed when the table is rendered.

`formats` see argument.

`row_info` see argument.

`line_grouping` see argument.

`ref_footnotes` see argument.

`main_title` see argument.

`subtitles` see argument.

`page_titles` see argument.

`main_footer` see argument.

`prov_footer` see argument.

`header_section_div` see argument.

`horizontal_sep` see argument.

`col_gap` see argument.

`table_inset` see argument.

as well as the following attributes:

`nlines_header` see argument.

`nrow_header` see argument.

`ncols` number of columns *of the table*, not including any row names/row labels

Note

The bare constructor for the `MatrixPrintForm` should generally only be called by `matrix_form` custom methods, and almost never from other code.

Examples

```
basic_matrix_form(iris) # calls matrix_form which calls this constructor
```

MatrixPrintForm-class *Class for Matrix Print Form*

Description

The `MatrixPrintForm` class, an intermediate representation for ASCII table printing.

matrix_form *Transform* `rtable` *to a list of matrices which can be used for outputting*

Description

Although `rtables` are represented as a tree data structure when outputting the table to ASCII or HTML, it is useful to map the `rtable` to an in-between state with the formatted cells in a matrix form.

Usage

```
matrix_form(
  obj,
  indent_rownames = FALSE,
  expand_newlines = TRUE,
  indent_size = 2,
  fontspec = NULL,
  col_gap = NULL
)

## S4 method for signature 'MatrixPrintForm'
matrix_form(
  obj,
  indent_rownames = FALSE,
  expand_newlines = TRUE,
  indent_size = 2,
  fontspec = NULL,
  col_gap = NULL
)
```

Arguments

`obj`	(ANY) object to be transformed into a ready-to-render form (a `MatrixPrintForm` object).

indent_rownames
: (flag)
if TRUE, the row names column in the strings matrix of obj will have indented row names (strings pre-fixed).

expand_newlines
: (flag)
whether the generated matrix form should expand rows whose values contain newlines into multiple 'physical' rows (as they will appear when rendered into ASCII). Defaults to TRUE.

indent_size
: (numeric(1))
number of spaces to be used per level of indent (if supported by the relevant method). Defaults to 2.

fontspec
: (font_spec)
a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().

col_gap
: (numeric(1))
the gap to be assumed between columns, in number of spaces with font specified by fontspec.

Value

A MatrixPrintForm classed list with an additional nrow_header attribute indicating the number of pseudo "rows" the column structure defines, with the following elements:

strings The content, as it should be printed, of the top-left material, column headers, row labels, and cell values of tt.

spans The column-span information for each print-string in the strings matrix.

aligns The text alignment for each print-string in the strings matrix.

display Whether each print-string in the strings matrix should be printed or not.

row_info The data.frame generated by basic_pagdf().

---

mf_strings *Getters and setters for aspects of* MatrixPrintForm *objects*

Description

Most of these functions, particularly the setters, are intended almost exclusively for internal use in, e.g., matrix_form methods, and should generally not be called by end users.

Usage

```
mf_strings(mf)

mf_spans(mf)
```

```
mf_aligns(mf)

mf_display(mf)

mf_formats(mf)

mf_rinfo(mf)

mf_cinfo(mf)

mf_has_topleft(mf)

mf_lgrouping(mf)

mf_rfnotes(mf)

mf_nlheader(mf)

mf_nrheader(mf)

mf_colgap(mf)

mf_fontspec(mf)

mf_fontspec(mf) <- value

mf_strings(mf) <- value

mf_spans(mf) <- value

mf_aligns(mf) <- value

mf_display(mf) <- value

mf_formats(mf) <- value

mf_rinfo(mf) <- value

mf_cinfo(mf) <- value

mf_lgrouping(mf) <- value

mf_rfnotes(mf) <- value

mf_nrheader(mf) <- value

mf_colgap(mf) <- value
```

```
mf_ncol(mf)

mf_nrow(mf)

mf_ncol(mf) <- value

## S4 method for signature 'MatrixPrintForm'
ncol(x)

mpf_has_rlabels(mf)

mf_has_rlabels(mf)
```

Arguments

mf	(MatrixPrintForm) a MatrixPrintForm object.
value	(ANY) the new value for the component in question.
x	MatrixPrintForm. The object.

Value

- Getters return the associated element of mf.
- Setters return the modified mf object.

---

mpf_to_rtf *Transform* MatrixPrintForm *to RTF*

Description

Experimental export to rich text format (RTF) via the r2rtf package.

Usage

```
mpf_to_rtf(
  mpf,
  colwidths = NULL,
  page_type = "letter",
  pg_width = page_dim(page_type)[if (landscape) 2 else 1],
  pg_height = page_dim(page_type)[if (landscape) 1 else 2],
  landscape = FALSE,
  margins = c(4, 4, 4, 4),
  font_family = "Courier",
  font_size = 8,
  lineheight = 1,
```

```
  fontspec = font_spec(font_family, font_size, lineheight),
  ...
)
```

Arguments

mpf	(MatrixPrintForm) a MatrixPrintForm object.
colwidths	(numeric) column widths.
page_type	(string) name of a page type. See page_types. Ignored when pg_width and pg_height are set directly.
pg_width	(numeric(1)) page width in inches.
pg_height	(numeric(1)) page height in inches.
landscape	(flag) whether the dimensions of page_type should be inverted for landscape orientation. Defaults to FALSE, ignored when pg_width and pg_height are set directly.
margins	(numeric(4)) named numeric vector containing "bottom", "left", "top", and "right" margins in inches. Defaults to .5 inches for both vertical margins and .75 for both horizontal margins.
font_family	(string) name of a font family. An error will be thrown if the family named is not monospaced. Defaults to "Courier".
font_size	(numeric(1)) font size. Defaults to 12.
lineheight	(numeric(1)) line height. Defaults to 1.
fontspec	(font_spec) a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().
...	additional parameters passed to individual methods.

Details

This function provides a low-level coercion of a MatrixPrintForm object into text containing the corresponding table in RTF. Currently, no pagination is done at this level, and should be done prior to calling this function, though that may change in the future.

Value

An RTF object.

nchar_ttype *Calculate font-specific string width*

Description

This function returns the width of each element `x` *as a multiple of the width of the space character for in declared font*, rounded up to the nearest integer. This is used extensively in the text rendering (`toString()`) and pagination machinery for calculating word wrapping, default column widths, lines per page, etc.

Usage

```
nchar_ttype(
  x,
  fontspec = font_spec(),
  tol = sqrt(.Machine$double.eps),
  raw = FALSE
)
```

Arguments

`x`	(`character`) the string(s) to calculate width(s) for.
`fontspec`	(`font_spec` or `NULL`) if non-NULL, the font to use for the calculations (as returned by `font_spec()`). Defaults to "Courier", which is a monospace font. If NULL, the width will be returned in number of characters by calling `nchar` directly.
`tol`	(`numeric(1)`) the tolerance to use when determining if a multiple needs to be rounded up to the next integer. See Details.
`raw`	(`logical(1)`) whether unrounded widths should be returned. Defaults to `FALSE`.

Details

String width is defined in terms of spaces within the specified font. For monospace fonts, this definition collapses to the number of characters in the string (`nchar()`), but for truetype fonts it does not.

For `raw = FALSE`, non-integer values (the norm in a truetype setting) for the number of spaces a string takes up is rounded up, *unless the multiple is less than* `tol` *above the last integer before it*. E.g., if `k - num_spaces < tol` for an integer `k`, `k` is returned instead of `k+1`.

See Also

`font_spec()`

Examples

```
nchar_ttype("hi there!")

nchar_ttype("hi there!", font_spec("Times"))
```

`nlines` *Number of lines required to print a value*

Description

Number of lines required to print a value

Usage

```
nlines(x, colwidths = NULL, max_width = NULL, fontspec, col_gap = NULL)

## S4 method for signature 'list'
nlines(x, colwidths = NULL, max_width = NULL, fontspec, col_gap = NULL)

## S4 method for signature 'NULL'
nlines(x, colwidths = NULL, max_width = NULL, fontspec, col_gap = NULL)

## S4 method for signature 'character'
nlines(x, colwidths = NULL, max_width = NULL, fontspec, col_gap = NULL)
```

Arguments

`x`	(ANY) the object to be printed.
`colwidths`	(numeric) column widths (if necessary). Principally used in `rtables`' method.
`max_width`	(numeric(1)) width that strings should be wrapped to when determining how many lines they require.
`fontspec`	(font_spec) a font_spec object specifying the font information to use for calculating string widths and heights, as returned by `font_spec()`.
`col_gap`	(numeric(1)) width of gap between columns in number of spaces. Only used by methods which must calculate span widths after wrapping.

Value

The number of lines needed to render the object `x`.

num_rep_cols *Number of repeated columns*

Description

When called on a table-like object using the formatters framework, this method returns the number of columns which are mandatorily repeated after each horizontal pagination.

Usage

```
num_rep_cols(obj)

## S4 method for signature 'ANY'
num_rep_cols(obj)

## S4 method for signature 'MatrixPrintForm'
num_rep_cols(obj)

num_rep_cols(obj) <- value

## S4 replacement method for signature 'ANY'
num_rep_cols(obj) <- value

## S4 replacement method for signature 'MatrixPrintForm'
num_rep_cols(obj) <- value
```

Arguments

obj	(ANY) a table-like object.
value	(numeric(1)) the new number of columns to repeat.

Details

Absent a class-specific method, this function returns 0, indicating no always-repeated columns.

Value

An integer.

Note

This number *does not* include row labels, the repetition of which is handled separately.

Examples

```
mpf <- basic_matrix_form(mtcars)
num_rep_cols(mpf)
lmpf <- basic_listing_mf(mtcars)
num_rep_cols(lmpf)
```

`open_font_dev` *Activate font state*

Description

Activate font state

Usage

```
open_font_dev(fontspec, silent = FALSE)

close_font_dev()

debug_font_dev()

undebug_font_dev()
```

Arguments

`fontspec`	(`font_spec`) a font_spec object specifying the font information to use for calculating string widths and heights, as returned by `font_spec()`.
`silent`	(`logical(1)`) If `FALSE`, the default, a warning will be emitted if this function switches away from an active graphics device.

Details

The font device state is an environment with four variables guaranteed to be set:

`open` (`logical(1)`)
: whether a device is already open with font info

`fontspec` (`font_spec`)
: the font specification, if any, that is currently active (`list()` if none is).

`spacewidth` (`numeric(1)`)
: the width of the space character in the currently active font.

`ismonospace` (`logical(1)`)
: whether the specified font is monospaced.

open_font_dev opens a pdf device with the specified font only if there is not one currently open with the same font. If a new device is opened, it caches spacewidth and ismonospace for use in nchar_ttype).

close_font_dev closes any open font state device and clears the cached values.

debug_font_dev and undebug_font_dev activate and deactivate, respectively, logging of where in the call stack font devices are being opened.

Value

- open_font_dev returns a logical value indicating whether a *new* pdf device was opened.
- close_font_dev, debug_font_dev and undebug_font_dev return NULL.

In all cases the value is returned invisibly.

Examples

```
open_font_dev(font_spec("Times"))
nchar_ttype("Hiya there", font_spec("Times"))
close_font_dev()
```

---

padstr *Pad a string and align within string*

Description

Pad a string and align within string

Usage

```
padstr(x, n, just = list_valid_aligns(), fontspec = font_spec())
```

Arguments

x	(string) a string.
n	(integer(1)) number of characters in the output string. If n < nchar(x), an error is thrown.
just	(string) text alignment justification to use. Defaults to "center". Must be one of "center", "right", "left", "dec_right", "dec_left", or "decimal".
fontspec	(font_spec) a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().

Value

x, padded to be a string of length n.

Examples

```
padstr("abc", 3)
padstr("abc", 4)
padstr("abc", 5)
padstr("abc", 5, "left")
padstr("abc", 5, "right")

## Not run:
# Expect error: "abc" has more than 1 characters
padstr("abc", 1)

## End(Not run)
```

---

pagdfrow *Create a row of a pagination data frame*

Description

Create a row of a pagination data frame

Usage

```
pagdfrow(
  row,
  nm = obj_name(row),
  lab = obj_label(row),
  rnum,
  pth,
  sibpos = NA_integer_,
  nsibs = NA_integer_,
  extent = nlines(row, colwidths, fontspec = fontspec),
  colwidths = NULL,
  repext = 0L,
  repind = integer(),
  indent = 0L,
  rclass = class(row),
  nrowrefs = 0L,
  ncellrefs = 0L,
  nreflines = 0L,
  force_page = FALSE,
  page_title = NA_character_,
  trailing_sep = NA_character_,
  fontspec
)
```

Arguments

row	(ANY) object representing the row, which is used for default values of nm, lab, extent, and rclass if provided. Must have methods for obj_name, obj_label, and nlines, to retrieve default values of nm, lab, and extent, respectively.
nm	(string) name.
lab	(string) label.
rnum	(numeric(1)) absolute row number.
pth	(character or NULL) path within larger table.
sibpos	(integer(1)) position among sibling rows.
nsibs	(integer(1)) number of siblings (including self).
extent	(numeric(1)) number of lines required to print the row.
colwidths	(numeric) column widths.
repext	(integer(1)) number of lines required to reprint all context for this row if it appears directly after pagination.
repind	(integer) vector of row numbers to be reprinted if this row appears directly after pagination.
indent	(integer) indent.
rclass	(string) class of row object.
nrowrefs	(integer(1)) number of row referential footnotes for this row.
ncellrefs	(integer(1)) number of cell referential footnotes for the cells in this row.
nreflines	(integer(1)) total number of lines required by all referential footnotes.
force_page	(flag) currently ignored.
page_title	(flag) currently ignored.

trailing_sep (string)
the string to use as a separator below this row during printing. If NA_character_, no separator is used.

fontspec (font_spec)
a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().

Value

A single row data.frame with the appropriate columns for a pagination info data frame.

---

page_lcpp *Determine lines per page (LPP) and characters per page (CPP) based on font and page type*

Description

Determine lines per page (LPP) and characters per page (CPP) based on font and page type

Usage

```
page_lcpp(
  page_type = page_types(),
  landscape = FALSE,
  font_family = "Courier",
  font_size = 8,
  lineheight = 1,
  margins = c(top = 0.5, bottom = 0.5, left = 0.75, right = 0.75),
  pg_width = NULL,
  pg_height = NULL,
  fontspec = font_spec(font_family, font_size, lineheight)
)
```

Arguments

page_type (string)
name of a page type. See page_types. Ignored when pg_width and pg_height are set directly.

landscape (flag)
whether the dimensions of page_type should be inverted for landscape orientation. Defaults to FALSE, ignored when pg_width and pg_height are set directly.

font_family (string)
name of a font family. An error will be thrown if the family named is not monospaced. Defaults to "Courier".

font_size (numeric(1))
font size. Defaults to 12.

lineheight (numeric(1))
line height. Defaults to 1.

margins (numeric(4))
named numeric vector containing "bottom", "left", "top", and "right" margins in inches. Defaults to .5 inches for both vertical margins and .75 for both horizontal margins.

pg_width (numeric(1))
page width in inches.

pg_height (numeric(1))
page height in inches.

fontspec (font_spec)
a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().

Value

A named list containing LPP (lines per page) and CPP (characters per page) elements suitable for use by the pagination machinery.

Examples

```
page_lcpp()
page_lcpp(font_size = 10)
page_lcpp("a4", font_size = 10)

page_lcpp(margins = c(top = 1, bottom = 1, left = 1, right = 1))
page_lcpp(pg_width = 10, pg_height = 15)
```

---

page_types *Supported named page types*

Description

List supported named page types.

Usage

```
page_types()

page_dim(page_type)
```

Arguments

page_type (string)
the name of a page size specification. Call page_types() for supported values.

Value

- `page_types` returns a character vector of supported page types
- `page_dim` returns the dimensions (width, then height) of the selected page type.

Examples

```
page_types()
page_dim("a4")
```

`paginate_indices` *Paginate a table-like object for rendering*

Description

These functions perform or diagnose bi-directional pagination on an object.

Usage

```
paginate_indices(
  obj,
  page_type = "letter",
  font_family = "Courier",
  font_size = 8,
  lineheight = 1,
  landscape = FALSE,
  pg_width = NULL,
  pg_height = NULL,
  margins = c(top = 0.5, bottom = 0.5, left = 0.75, right = 0.75),
  lpp = NA_integer_,
  cpp = NA_integer_,
  min_siblings = 2,
  nosplitin = list(rows = character(), cols = character()),
  colwidths = NULL,
  tf_wrap = FALSE,
  max_width = NULL,
  indent_size = 2,
  pg_size_spec = NULL,
  rep_cols = num_rep_cols(obj),
  col_gap = 3,
  fontspec = font_spec(font_family, font_size, lineheight),
  verbose = FALSE
)

paginate_to_mpfs(
  obj,
  page_type = "letter",
```

```
  font_family = "Courier",
  font_size = 8,
  lineheight = 1,
  landscape = FALSE,
  pg_width = NULL,
  pg_height = NULL,
  margins = c(top = 0.5, bottom = 0.5, left = 0.75, right = 0.75),
  lpp = NA_integer_,
  cpp = NA_integer_,
  min_siblings = 2,
  nosplitin = character(),
  colwidths = NULL,
  tf_wrap = FALSE,
  max_width = NULL,
  indent_size = 2,
  pg_size_spec = NULL,
  page_num = default_page_number(),
  rep_cols = NULL,
  col_gap = 3,
  fontspec = font_spec(font_family, font_size, lineheight),
  verbose = FALSE
)

diagnose_pagination(
  obj,
  page_type = "letter",
  font_family = "Courier",
  font_size = 8,
  lineheight = 1,
  landscape = FALSE,
  pg_width = NULL,
  pg_height = NULL,
  margins = c(top = 0.5, bottom = 0.5, left = 0.75, right = 0.75),
  lpp = NA_integer_,
  cpp = NA_integer_,
  min_siblings = 2,
  nosplitin = character(),
 colwidths = propose_column_widths(matrix_form(obj, TRUE), fontspec = fontspec),
  tf_wrap = FALSE,
  max_width = NULL,
  indent_size = 2,
  pg_size_spec = NULL,
  rep_cols = num_rep_cols(obj),
  col_gap = 3,
  verbose = FALSE,
  fontspec = font_spec(font_family, font_size, lineheight),
  ...
)
```

Arguments

obj	(ANY) object to be paginated. Must have a matrix_form() method.
page_type	(string) name of a page type. See page_types. Ignored when pg_width and pg_height are set directly.
font_family	(string) name of a font family. An error will be thrown if the family named is not monospaced. Defaults to "Courier".
font_size	(numeric(1)) font size. Defaults to 12.
lineheight	(numeric(1)) line height. Defaults to 1.
landscape	(flag) whether the dimensions of page_type should be inverted for landscape orientation. Defaults to FALSE, ignored when pg_width and pg_height are set directly.
pg_width	(numeric(1)) page width in inches.
pg_height	(numeric(1)) page height in inches.
margins	(numeric(4)) named numeric vector containing "bottom", "left", "top", and "right" margins in inches. Defaults to .5 inches for both vertical margins and .75 for both horizontal margins.
lpp	(numeric(1) or NULL) lines per page. If NA (the default), this is calculated automatically based on the specified page size). NULL indicates no vertical pagination should occur.
cpp	(numeric(1) or NULL) width (in characters) per page. If NA (the default), this is calculated automatically based on the specified page size). NULL indicates no horizontal pagination should occur.
min_siblings	(numeric) minimum sibling rows which must appear on either side of pagination row for a mid-subtable split to be valid. Defaults to 2 for tables. It is automatically turned off (set to 0) for listings.
nosplitin	(character) list of names of subtables where page breaks are not allowed, regardless of other considerations. Defaults to none.
colwidths	(numeric) vector of column widths (in characters) for use in vertical pagination.
tf_wrap	(flag) whether the text for title, subtitles, and footnotes should be wrapped.

max_width	(integer(1), string or NULL) width that title and footer (including footnotes) materials should be word-wrapped to. If NULL, it is set to the current print width of the session (getOption("width")). If set to "auto", the width of the table (plus any table inset) is used. Parameter is ignored if tf_wrap = FALSE.
indent_size	(numeric(1)) indent size, in characters. Ignored when x is already a MatrixPrintForm object in favor of information there.
pg_size_spec	(page_size_spec) . a pre-calculated page size specification. Typically this is not set by end users.
rep_cols	(numeric(1)) number of *columns* (not including row labels) to be repeated on every page. Defaults to 0.
col_gap	(numeric(1)) The number of spaces to be placed between columns in the rendered table (and assumed for horizontal pagination).
fontspec	(font_spec) a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().
verbose	(flag) whether additional informative messages about the search for pagination breaks should be shown. Defaults to FALSE.
page_num	(string) placeholder string for page numbers. See default_page_number for more information. Defaults to NULL.
...	additional parameters passed to individual methods.

Details

paginate_indices renders obj into a MatrixPrintForm (MPF), then uses that representation to calculate the rows and columns of obj corresponding to each page of the pagination of obj, but simply returns these indices rather than paginating obj itself (see Details for an important caveat).

paginate_to_mpfs renders obj into its MPF intermediate representation, then paginates that MPF into component MPFs each corresponding to an individual page and returns those in a list.

diagnose_pagination attempts pagination via paginate_to_mpfs, then returns diagnostic information which explains why page breaks were positioned where they were, or alternatively why no valid pagination could be found.

All three of these functions generally support all classes which have a corresponding matrix_form() method which returns a valid MatrixPrintForm object (including MatrixPrintForm objects themselves).

paginate_indices is directly called by paginate_to_mpfs (and thus diagnose_pagination). For most classes, and most tables represented by supported classes, calling paginate_to_mpfs is equivalent to a manual paginate_indices -> subset obj into pages -> matrix_form workflow.

The exception to this equivalence is objects which support "forced pagination", or pagination logic which is built into the object itself rather than being a function of space on a page. Forced pagination generally involves the creation of, e.g., page-specific titles which apply to these forced paginations. `paginate_to_mpfs` and `diagnose_pagination` support forced pagination by automatically calling the `do_forced_paginate()` generic on the object and then paginating each object returned by that generic separately. The assumption here, then, is that page-specific titles and such are handled by the class' `do_forced_paginate()` method.

`paginate_indices`, on the other hand, *does not support forced pagination*, because it returns only a set of indices for row and column subsetting for each page, and thus cannot retain any changes, e.g., to titles, done within `do_forced_paginate()`. `paginate_indices` does call `do_forced_paginate()`, but instead of continuing it throws an error in the case that the result is larger than a single "page".

`diagnose_pagination` attempts pagination and then, regardless of success or failure, returns diagnostic information about pagination attempts (if any) after each row and column.

The diagnostics data reflects the final time the pagination algorithm evaluated a page break at the specified location, regardless of how many times the position was assessed in total.

To get information about intermediate attempts, perform pagination with `verbose = TRUE` and inspect the messages in order.

Value

- `paginate_indices` returns a `list` with two elements of the same length: `pag_row_indices` and `pag_col_indices`.
- `paginate_to_mpfs` returns a `list` of `MatrixPrintForm` objects representing each individual page after pagination (including forced pagination if necessary).
- `diagnose_pagination` returns a `list` containing:

  `lpp_diagnostics` Diagnostic information regarding lines per page.

  `row_diagnostics` Basic information about rows, whether pagination was attempted after each row, and the final result of such an attempt, if made.

  `cpp_diagnostics` Diagnostic information regarding columns per page.

  `col_diagnostics` Very basic information about leaf columns, whether pagination was attempted after each leaf column, ad the final result of such attempts, if made.

Note

For `diagnose_pagination`, the column labels are not displayed in the `col_diagnostics` element due to certain internal implementation details; rather the diagnostics are reported in terms of absolute (leaf) column position. This is a known limitation, and may eventually be changed, but the information remains useful as it is currently reported.

`diagnose_pagination` is intended for interactive debugging use and *should not be programmed against*, as the exact content and form of the verbose messages it captures and returns is subject to change.

Because `diagnose_pagination` relies on `capture.output(type = "message")`, it cannot be used within the `testthat` (and likely other) testing frameworks, and likely cannot be used within `knitr`/`rmarkdown` contexts either, as this clashes with those systems' capture of messages.

Examples

```
mpf <- basic_matrix_form(mtcars)

paginate_indices(mpf, pg_width = 5, pg_height = 3)

paginate_to_mpfs(mpf, pg_width = 5, pg_height = 3)

diagnose_pagination(mpf, pg_width = 5, pg_height = 3)
clws <- propose_column_widths(mpf)
clws[1] <- floor(clws[1] / 3)
dgnost <- diagnose_pagination(mpf, pg_width = 5, pg_height = 3, colwidths = clws)
try(diagnose_pagination(mpf, pg_width = 1)) # fails
```

---

pagination_algo *Pagination*

Description

Pagination

Pagination Algorithm

Pagination is performed independently in the vertical and horizontal directions based solely on a *pagination data frame*, which includes the following information for each row/column:

- Number of lines/characters rendering the row will take **after word-wrapping** (`self_extent`)
- The indices (`reprint_inds`) and number of lines (`par_extent`) of the rows which act as **context** for the row
- The row's number of siblings and position within its siblings

Given `lpp` (`cpp`) is already adjusted for rendered elements which are not rows/columns and a data frame of pagination information, pagination is performed via the following algorithm with `start = 1`.

Core Pagination Algorithm:

1. Initial guess for pagination position is `start + lpp` (`start + cpp`)
2. While the guess is not a valid pagination position, and `guess > start`, decrement guess and repeat.
   - An error is thrown if all possible pagination positions between `start` and `start + lpp` (`start + cpp`) would be `< start` after decrementing
3. Retain pagination index
4. If pagination point was less than `NROW(tt)` (`ncol(tt)`), set `start` to `pos + 1`, and repeat steps (1) - (4).

Validating Pagination Position:

Given an (already adjusted) `lpp` or `cpp` value, a pagination is invalid if:

- The rows/columns on the page would take more than (adjusted) `lpp` lines/`cpp` characters to render **including**:
  - word-wrapping
  - (vertical only) context repetition
- (vertical only) footnote messages and/or section divider lines take up too many lines after rendering rows
- (vertical only) row is a label or content (row-group summary) row
- (vertical only) row at the pagination point has siblings, and it has less than `min_siblings` preceding or following siblings
- pagination would occur within a sub-table listed in `nosplitin`

`pag_indices_inner` *Find pagination indices from pagination info data frame*

Description

Pagination methods should typically call the `make_row_df` method for their object and then call this function on the resulting pagination info `data.frame`.

Usage

```
pag_indices_inner(
  pagdf,
  rlpp,
  lpp_or_cpp = NA_integer_,
  context_lpp_or_cpp = NA_integer_,
  min_siblings,
  nosplitin = character(),
  verbose = FALSE,
  row = TRUE,
  have_col_fnotes = FALSE,
  div_height = 1L,
  col_gap = 3L,
  has_rowlabels
)
```

Arguments

`pagdf`	(`data.frame`) a pagination info `data.frame` as created by either `make_rows_df` or `make_cols_df`.
`rlpp`	(`numeric`) maximum number of *row* lines per page (not including header materials), including (re)printed header and context rows.

lpp_or_cpp (numeric)
total maximum number of *row* lines or content (column-wise characters) per page (including header materials and context rows). This is only for informative results with verbose = TRUE. It will print NA if not specified by the pagination machinery.

context_lpp_or_cpp (numeric)
total number of context *row* lines or content (column-wise characters) per page (including header materials). Uses NA if not specified by the pagination machinery and is only for informative results with verbose = TRUE.

min_siblings (numeric)
minimum sibling rows which must appear on either side of pagination row for a mid-subtable split to be valid. Defaults to 2 for tables. It is automatically turned off (set to 0) for listings.

nosplitin (character)
list of names of subtables where page breaks are not allowed, regardless of other considerations. Defaults to none.

verbose (flag)
whether additional informative messages about the search for pagination breaks should be shown. Defaults to FALSE.

row (flag)
whether pagination is happening in row space (TRUE, the default) or column space (FALSE).

have_col_fnotes (flag)
whether the table-like object being rendered has column-associated referential footnotes.

div_height (numeric(1))
the height of the divider line when the associated object is rendered. Defaults to 1.

col_gap (numeric(1))
width of gap between columns, in same units as extent in pagdf (spaces under a particular font specification).

has_rowlabels (logical(1))
whether the object being paginated has row labels.

Details

pab_indices_inner implements the core pagination algorithm (see below) for a single direction (vertical if row = TRUE (the default), horizontal otherwise) based on the pagination data frame and (already adjusted for non-body rows/columns) lines (or characters) per page.

Value

A list containing a vector of row numbers, broken up by page.

Pagination Algorithm

Pagination is performed independently in the vertical and horizontal directions based solely on a *pagination data frame*, which includes the following information for each row/column:

- Number of lines/characters rendering the row will take **after word-wrapping** (`self_extent`)
- The indices (`reprint_inds`) and number of lines (`par_extent`) of the rows which act as **context** for the row
- The row's number of siblings and position within its siblings

Given `lpp` (`cpp`) is already adjusted for rendered elements which are not rows/columns and a data frame of pagination information, pagination is performed via the following algorithm with `start = 1`.

Core Pagination Algorithm:

1. Initial guess for pagination position is `start + lpp` (`start + cpp`)
2. While the guess is not a valid pagination position, and `guess > start`, decrement guess and repeat.
   - An error is thrown if all possible pagination positions between `start` and `start + lpp` (`start + cpp`) would be `< start` after decrementing
3. Retain pagination index
4. If pagination point was less than `NROW(tt)` (`ncol(tt)`), set `start` to `pos + 1`, and repeat steps (1) - (4).

Validating Pagination Position:

Given an (already adjusted) `lpp` or `cpp` value, a pagination is invalid if:

- The rows/columns on the page would take more than (adjusted) `lpp` lines/`cpp` characters to render **including**:
  - word-wrapping
  - (vertical only) context repetition
- (vertical only) footnote messages and/or section divider lines take up too many lines after rendering rows
- (vertical only) row is a label or content (row-group summary) row
- (vertical only) row at the pagination point has siblings, and it has less than `min_siblings` preceding or following siblings
- pagination would occur within a sub-table listed in `nosplitin`

Examples

```
mypgdf <- basic_pagdf(row.names(mtcars))

paginds <- pag_indices_inner(mypgdf, rlpp = 15, min_siblings = 0)
lapply(paginds, function(x) mtcars[x, ])
```

print,ANY-method *Print*

Description

Print an R object. See `print()`.

Usage

```
## S4 method for signature 'ANY'
print(x, ...)
```

Arguments

x	an object used to select a method.
...	further arguments passed to or from other methods.

propose_column_widths *Propose column widths based on the* MatrixPrintForm *of an object*

Description

Row names are also considered a column for the output.

Usage

```
propose_column_widths(x, indent_size = 2, fontspec = font_spec())
```

Arguments

x	(ANY) a MatrixPrintForm object, or an object with a matrix_form method.
indent_size	(numeric(1)) indent size, in characters. Ignored when x is already a MatrixPrintForm object in favor of information there.
fontspec	(font_spec) a font_spec object specifying the font information to use for calculating string widths and heights, as returned by `font_spec()`.

Value

A vector of column widths based on the content of x for use in printing and pagination.

Examples

```
mf <- basic_matrix_form(mtcars)
propose_column_widths(mf)
```

`ref_df_row` *Create a row for a referential footnote information data frame*

Description

Create a row for a referential footnote information data frame

Usage

```
ref_df_row(
  row_path = NA_character_,
  col_path = NA_character_,
  row = NA_integer_,
  col = NA_integer_,
  symbol = NA_character_,
  ref_index = NA_integer_,
  msg = NA_character_,
  max_width = NULL
)
```

Arguments

`row_path`	(`character`) row path (or `NA_character_` for none).
`col_path`	(`character`) column path (or `NA_character_` for none).
`row`	(`integer(1)`) integer position of the row.
`col`	(`integer(1)`) integer position of the column.
`symbol`	(`string`) symbol for the reference. `NA_character_` to use the `ref_index` automatically.
`ref_index`	(`integer(1)`) index of the footnote, used for ordering even when symbol is not `NA`.
`msg`	(`string`) the string message, not including the symbol portion (`{symbol} - `)
`max_width`	(`numeric(1)`) width that strings should be wrapped to when determining how many lines they require.

Value

A single row data frame with the appropriate columns.

round_fmt *Round and prepare a value for display*

Description

This function is used within `format_value()` to prepare numeric values within cells for formatting and display.

Usage

```
round_fmt(x, digits, na_str = "NA")
```

Arguments

`x`	(`numeric(1)`) value to format.
`digits`	(`numeric(1)`) number of digits to round to, or `NA` to convert to a character value with no rounding.
`na_str`	(`string`) the value to return if `x` is `NA`.

Details

This function combines the rounding behavior of R's standards-compliant `round()` function (see the Details section of that documentation) with the strict decimal display of `sprintf()`. The exact behavior is as follows:

1. If `x` is `NA`, the value of `na_str` is returned.
2. If `x` is non-`NA` but `digits` is `NA`, `x` is converted to a character and returned.
3. If `x` and `digits` are both non-NA, `round()` is called first, and then `sprintf()` is used to convert the rounded value to a character with the appropriate number of trailing zeros enforced.

Value

A character value representing the value after rounding, containing any trailing zeros required to display *exactly* `digits` elements.

Note

This differs from the base R `round()` function in that `NA` digits indicate `x` should be converted to character and returned unchanged whereas `round(x, digits=NA)` returns `NA` for all values of `x`.

This behavior will differ from `as.character(round(x, digits = digits))` in the case where there are not at least `digits` significant digits after the decimal that remain after rounding. It *may* differ from `sprintf("\%.Nf", x)` for values ending in 5 after the decimal place on many popular operating systems due to `round`'s stricter adherence to the IEC 60559 standard, particularly for R versions > 4.0.0 (see warning in `round()` documentation).

See Also

format_value(), round(), sprintf()

Examples

```
round_fmt(0, digits = 3)
round_fmt(.395, digits = 2)
round_fmt(NA, digits = 1)
round_fmt(NA, digits = 1, na_str = "-")
round_fmt(2.765923, digits = NA)
```

---

spans_to_viscell *Transform a vector of spans (with duplication) into a visibility vector*

Description

Transform a vector of spans (with duplication) into a visibility vector

Usage

```
spans_to_viscell(spans)
```

Arguments

spans	(numeric) a vector of spans, with each span value repeated for the cells it covers.

Details

The values of spans are assumed to be repeated such that each individual position covered by the span has the repeated value.

This means that each block of values in spans must be of a length at least equal to its value (i.e. two 2s, three 3s, etc).

This function correctly handles cases where two spans of the same size are next to each other; i.e., a block of four 2s represents two large cells each of which spans two individual cells.

Value

A logical vector the same length as spans indicating whether the contents of a string vector with those spans is valid.

Note

Currently no checking or enforcement is done to verify that the vector of spans is valid according to the specifications described in the Details section above.

Examples

```
spans_to_viscell(c(2, 2, 2, 2, 1, 3, 3, 3))
```

`split_word_ttype` *wrap string given a Truetype font*

Description

wrap string given a Truetype font

Usage

```
split_word_ttype(str, width, fontspec, min_ok_chars)

wrap_string_ttype(
  str,
  width,
  fontspec,
  collapse = NULL,
  min_ok_chars = min(floor(nchar(str)/2), 4, floor(width/2)),
  wordbreak_ok = TRUE
)
```

Arguments

`str`	(`string`, `character`, or `list`) string to be wrapped. If it is a `vector` or a `list`, it will be looped as a `list` and returned with `unlist(use.names = FALSE)`.
`width`	(`numeric(1)`) width, in characters, that the text should be wrapped to.
`fontspec`	(`font_spec`) a font_spec object specifying the font information to use for calculating string widths and heights, as returned by `font_spec()`.
`min_ok_chars`	(`numeric(1)`) number of minimum characters that remain on either side when a word is split.
`collapse`	(`string` or `NULL`) collapse character used to separate segments of words that have been split and should be pasted together. This is usually done internally with `"\n"` to update the wrapping along with other internal values.
`wordbreak_ok`	(`logical(1)`) should breaking within a word be allowed? If, `FALSE`, attempts to wrap a string to a width narrower than its widest word will result in an error.

Value

`str`, broken up into a word-wrapped vector

spread_integer *Spread an integer to a given length*

Description

Spread an integer to a given length

Usage

```
spread_integer(x, len)
```

Arguments

x	(integer(1)) number to spread.
len	(integer(1)) number of times to repeat x.

Value

If x is a scalar whole number value (see is.wholenumber()), the value x is repeated len times. Otherwise, an error is thrown.

Examples

```
spread_integer(3, 1)
spread_integer(0, 3)
spread_integer(1, 3)
spread_integer(2, 3)
spread_integer(3, 3)
spread_integer(4, 3)
spread_integer(5, 3)
spread_integer(6, 3)
spread_integer(7, 3)
```

sprintf_format *Specify text format via a* sprintf *format string*

Description

Specify text format via a sprintf format string

Usage

```
sprintf_format(format)
```

Arguments

format	(string) a format string passed to sprintf().

Value

A formatting function which wraps and applies the specified sprintf-style format to string format.

See Also

sprintf()

Examples

```
fmtfun <- sprintf_format("(N=%i")
format_value(100, format = fmtfun)

fmtfun2 <- sprintf_format("%.4f - %.2f")
format_value(list(12.23456, 2.724))
```

---

table_inset *Access or (recursively) set table inset*

Description

Table inset is the amount of characters that the body of a table, referential footnotes, and main footer material are inset from the left-alignment of the titles and provenance footer materials.

Usage

```
table_inset(obj)

## S4 method for signature 'MatrixPrintForm'
table_inset(obj)

table_inset(obj) <- value

## S4 replacement method for signature 'MatrixPrintForm'
table_inset(obj) <- value
```

Arguments

obj	(ANY) object to get or (recursively if necessary) set table inset for.
value	(string) string to use as new header/body separator.

Value

- table_inset returns the integer value that the table body (including column heading information and section dividers), referential footnotes, and main footer should be inset from the left alignment of the titles and provenance footers during rendering.
- table_inset<- returns obj with the new table_inset value applied recursively to it and all its subtables.

---

test_matrix_form *Create spoof matrix form from a data frame*

Description

Useful functions for writing tests and examples, and a starting point for more sophisticated custom matrix_form methods.

Usage

```
basic_matrix_form(
  df,
  indent_rownames = FALSE,
  parent_path = NULL,
  ignore_rownames = FALSE,
  add_decoration = FALSE,
  fontspec = font_spec(),
  split_labels = NULL,
  data_labels = NULL,
  num_rep_cols = 0L
)

basic_listing_mf(
  df,
  keycols = names(df)[1],
  add_decoration = TRUE,
  fontspec = font_spec()
)
```

Arguments

df	(data.frame) a data frame.
indent_rownames	(flag) whether row names should be indented. Being this used for testing purposes, it defaults to FALSE. If TRUE, it assigns label rows on even lines (also format is "-" and value strings are ""). Indentation works only if split labels are used (see parameters split_labels and data_labels).

parent_path (string)
: parent path that all rows should be "children of". Defaults to NULL, as usually this is not needed. It may be necessary to use "root", for some specific scenarios.

ignore_rownames (flag)
: whether row names should be ignored.

add_decoration (flag)
: whether adds title and footer decorations should be added to the matrix form.

fontspec (font_spec)
: a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().

split_labels (string)
: indicates which column to use as split labels. If NULL, no split labels are used.

data_labels (string)
: indicates which column to use as data labels. It is ignored if no split_labels is present and is automatically assigned to "Analysis method" when split_labels is present, but data_labels is NULL. Its direct column name is used as node name in "DataRow" pathing. See mf_rinfo() for more information.

num_rep_cols (numeric(1))
: Number of columns to be treated as repeating columns. Defaults to 0 for basic_matrix_form and length(keycols) for basic_listing_mf. Note repeating columns are separate from row labels if present.

keycols (character)
: a vector of df column names that are printed first and for which repeated values are assigned "". This format is characteristic of a listing matrix form.

Details

If some of the column has a obj_format assigned, it will be respected for all column values except for label rows, if present (see parameter split_labels).

Value

A valid MatrixPrintForm object representing df that is ready for ASCII rendering.

A valid MatrixPrintForm object representing df as a listing that is ready for ASCII rendering.

Functions

- basic_listing_mf(): Create a MatrixPrintForm object from data frame df that respects the default formats for a listing object.

Examples

```
mform <- basic_matrix_form(mtcars)
cat(toString(mform))
```

```
# Advanced test case with label rows
library(dplyr)
iris_output <- iris %>%
  group_by(Species) %>%
  summarize("all obs" = round(mean(Petal.Length), 2)) %>%
  mutate("DataRow_label" = "Mean")
mf <- basic_matrix_form(iris_output,
  indent_rownames = TRUE,
  split_labels = "Species", data_labels = "DataRow_label"
)
cat(toString(mf))

mform <- basic_listing_mf(mtcars)
cat(toString(mform))
```

---

toString *Transform objects into string representations*

Description

Transform a complex object into a string representation ready to be printed or written to a plain-text file.

All objects that are printed to console pass via `toString`. This function allows fundamental formatting specifications to be applied to final output, like column widths and relative wrapping (`width`), title and footer wrapping (`tf_wrap = TRUE` and `max_width`), and horizontal separator character (e.g. `hsep = "+"`).

Usage

```
toString(x, ...)

## S4 method for signature 'MatrixPrintForm'
toString(
  x,
  widths = NULL,
  tf_wrap = FALSE,
  max_width = NULL,
  col_gap = mf_colgap(x),
  hsep = NULL,
  fontspec = font_spec(),
  ttype_ok = FALSE
)
```

Arguments

`x`	`(ANY)` object to be prepared for rendering.

...	additional parameters passed to individual methods.
widths	(numeric or NULL) Proposed widths for the columns of x. The expected length of this numeric vector can be retrieved with ncol(x) + 1 as the column of row names must also be considered.
tf_wrap	(flag) whether the text for title, subtitles, and footnotes should be wrapped.
max_width	(integer(1), string or NULL) width that title and footer (including footnotes) materials should be word-wrapped to. If NULL, it is set to the current print width of the session (getOption("width")). If set to "auto", the width of the table (plus any table inset) is used. Parameter is ignored if tf_wrap = FALSE.
col_gap	(numeric(1)) space (in characters) between columns.
hsep	(string) character to repeat to create header/body separator line. If NULL, the object value will be used. If " ", an empty separator will be printed. See default_hsep() for more information.
fontspec	(font_spec) a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().
ttype_ok	(logical(1)) should truetype (non-monospace) fonts be allowed via fontspec. Defaults to FALSE. This parameter is primarily for internal testing and generally should not be set by end users.

Details

Manual insertion of newlines is not supported when tf_wrap = TRUE and will result in a warning and undefined wrapping behavior. Passing vectors of already split strings remains supported, however in this case each string is word-wrapped separately with the behavior described above.

Value

A character string containing the ASCII rendering of the table-like object represented by x.

See Also

wrap_string()

Examples

```
mform <- basic_matrix_form(mtcars)
cat(toString(mform))
```

var_labels *Get label attributes of variables in a* data.frame

Description

Variable labels can be stored as a label attribute for each variable. This functions returns a named character vector with the variable labels (or empty strings if not specified).

Usage

```
var_labels(x, fill = FALSE)
```

Arguments

x	(data.frame) a data frame object.
fill	(flag) whether variable names should be returned for variables for which the label attribute does not exist. If FALSE, these variables are filled with NAs instead.

Value

a named character vector of variable labels from x, with names corresponding to variable names.

Examples

```
x <- iris
var_labels(x)
var_labels(x) <- paste("label for", names(iris))
var_labels(x)
```

var_labels<- *Set label attributes of all variables in a* data.frame

Description

Variable labels can be stored as the label attribute for each variable. This functions sets all non-missing (non-NA) variable labels in a data.frame.

Usage

```
var_labels(x) <- value
```

Arguments

x	(data.frame) a data frame object.
value	(character) a vector of new variable labels. If any values are NA, the label for that variable is removed.

Value

x with modified variable labels.

Examples

```
x <- iris
var_labels(x)
var_labels(x) <- paste("label for", names(iris))
var_labels(x)

if (interactive()) {
  View(x) # in RStudio data viewer labels are displayed
}
```

---

var_labels_remove *Remove variable labels of a* data.frame

Description

Remove label attribute from all variables in a data frame.

Usage

```
var_labels_remove(x)
```

Arguments

x	(data.frame) a data.frame object.

Value

x with its variable labels stripped.

Examples

```
x <- var_labels_remove(iris)
```

var_relabel *Copy and change variable labels of a* data.frame

Description

Relabel a subset of the variables.

Usage

```
var_relabel(x, ...)
```

Arguments

x	(data.frame) a data frame object.
...	name-value pairs, where each name corresponds to a variable name in x and the value to the new variable label.

Value

A copy of x with labels modified according to ...

Examples

```
x <- var_relabel(iris, Sepal.Length = "Sepal Length of iris flower")
var_labels(x)
```

vert_pag_indices *Find column indices for vertical pagination*

Description

Find column indices for vertical pagination

Usage

```
vert_pag_indices(
  obj,
  cpp = 40,
  colwidths = NULL,
  verbose = FALSE,
  rep_cols = 0L,
  fontspec,
  nosplitin = character()
)
```

Arguments

obj	(ANY) object to be paginated. Must have a matrix_form() method.
cpp	(numeric(1)) number of characters per page (width).
colwidths	(numeric) vector of column widths (in characters) for use in vertical pagination.
verbose	(flag) whether additional informative messages about the search for pagination breaks should be shown. Defaults to FALSE.
rep_cols	(numeric(1)) number of *columns* (not including row labels) to be repeated on every page. Defaults to 0.
fontspec	(font_spec) a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().
nosplitin	(character) list of names of subtables where page breaks are not allowed, regardless of other considerations. Defaults to none.

Value

A list partitioning the vector of column indices into subsets for 1 or more horizontally paginated pages.

Examples

```
mf <- basic_matrix_form(df = mtcars)
colpaginds <- vert_pag_indices(mf, fontspec = font_spec())
lapply(colpaginds, function(j) mtcars[, j, drop = FALSE])
```

---

with_label *Return an object with a label attribute*

Description

Return an object with a label attribute

Usage

```
with_label(x, label)
```

Arguments

x	(ANY) an object.
label	(string) label attribute to attach to x.

Value

x labeled by label. Note that the exact mechanism of labeling should be considered an internal implementation detail, but the label can always be retrieved via obj_label.

Examples

```
x <- with_label(c(1, 2, 3), label = "Test")
obj_label(x)
```

---

wrap_string *Wrap a string to a precise width*

Description

Core wrapping functionality that preserves whitespace. Newline character "\n" is not supported by core functionality stringi::stri_wrap(). This is usually solved beforehand by matrix_form(). If the width is smaller than any large word, these will be truncated after width characters. If the split leaves trailing groups of empty spaces, they will be dropped.

Usage

```
wrap_string(str, width, collapse = NULL, fontspec = font_spec())

wrap_txt(str, width, collapse = NULL, fontspec = font_spec())
```

Arguments

str	(string, character, or list) string to be wrapped. If it is a vector or a list, it will be looped as a list and returned with unlist(use.names = FALSE).
width	(numeric(1)) width, in characters, that the text should be wrapped to.
collapse	(string or NULL) collapse character used to separate segments of words that have been split and should be pasted together. This is usually done internally with "\n" to update the wrapping along with other internal values.
fontspec	(font_spec) a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().

Details

Word wrapping happens similarly to `stringi::stri_wrap()` with the following difference: individual words which are longer than `max_width` are broken up in a way that fits with other word wrapping.

Value

A string if `str` is one element and if `collapse = NULL`. Otherwise, a list of elements (if `length(str) > 1`) that can contain strings or vectors of characters (if `collapse = NULL`).

Functions

- `wrap_txt()`: Deprecated function. Please use `wrap_string()` instead.

Examples

```
str <- list(
  "  , something really  \\tnot  very good", # \t needs to be escaped
  "  but I keep it12   "
)
wrap_string(str, 5, collapse = "\n")

wrap_txt(str, 5, collapse = NULL)
```

Index