Package 'formatters'

December 6, 2023

Title ASCII Formatting for Values and Tables

Version 0.5.5

Date 2023-12-05

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://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)

Suggests dplyr (>= 1.0.9), gt (>= 0.7.0), huxtable (>= 2.0.0), knitr (>= 1.42), r2rtf (>= 0.3.2), stringi (>= 1.6), testthat (>= 3.0.4)

VignetteBuilder knitr

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

Config/Needs/website insightsengineering/nesttemplate

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.2.3

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'

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 2023-12-06 09:20:06 UTC

R topics documented:

`formatters-package` *formatters Package*

Description

Package to format tables and listings in a flexible way.

Author(s)

Maintainer: Joe Zhu <joe.zhu@roche.com> [contributor]

Authors:

- Gabriel Becker <gabembecker@gmail.com> (original creator of the package)
- Adrian Waddell <adrian.waddell@gene.com>

Other contributors:

- Davide Garolini <davide.garolini@roche.com> [contributor]
- Emily de la Rua <emily.de_la_rua@contractors.roche.com> [contributor]
- Abinaya Yogasekaram <abinaya.yogasekaram@contractors.roche.com> [contributor]
- F. Hoffmann-La Roche AG [copyright holder, funder]

See Also

Useful links:

- https://github.com/insightsengineering/formatters
- Report bugs at https://github.com/insightsengineering/formatters/issues

basic_matrix_form *Create spoof matrix form from a data.frame*

Description

This is useful primarily for writing testing/examples, and as a starting point for more sophisticated custom `matrix_form` methods

Usage

```
basic_matrix_form(df, parent_path = "root")
```

Arguments

`df`	data.frame
`parent_path`	character. parent path that all rows should be "children of", defaults to `"root"`, and generally should not matter to end users.

Value

A valid `MatrixPrintForm` object representing `df`, ready for ASCII rendering

Examples

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

basic_pagdf *Basic/spoof pagination info dataframe*

Description

Returns a minimal pagination info data.frame (with no sibling/footnote/etc info).

Usage

```
basic_pagdf(
  rnames,
  labs = rnames,
  rnums = seq_along(rnames),
  extents = 1L,
  rclass = "NA",
  parent_path = "root"
)
```

Arguments

rnames	character. Vector of row names
labs	character. Vector of row labels (defaults to names)
rnums	integer. Vector of row numbers. Defaults to seq_along(rnames).
extents	integer. Number of lines each row will take to print, defaults to 1 for all rows
rclass	character. Class(es) for the rows. Defaults to "NA"
parent_path	character. parent path that all rows should be "children of", defaults to "root", and generally should not matter to end users.

Value

A data.frame suitable for use in both the matrix_print_form 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	either format string or an object returned by sprintf_format
stop_otherwise	logical, if x is not a format should an error be thrown
algn	vector of characters that indicates the requested cell alignments.

Value

- is_valid_format: TRUE if x is NULL, a supported format string, or a function; FALSE otherwise.
- check_aligns: TRUE if it passes the check.

Note

No check if the function is actually a `formatter` is performed.

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 matrix print form object.
`align_mat`	character matrix. Aligns matrix component of matrix print form object. Should contain either `dec_left`, `dec_right` or `decimal` for values to be decimal aligned.

Details

Decimal alignment left and right (`dec_left` and `dec_right`) are different to center decimal alignment `decimal` only in the case some padding is present. This may happen if column widths are wider by setting parameters `widths` in `toString` or `colwidths` in `paginate_*` accordingly. It will be also the case (more common) of wider column names. Decimal alignment is not supported along with cell wrapping.

Value

Processed string matrix of matrix print form with decimal aligned values.

See Also

toString and 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-likes.

Usage

```
default_hsep()

set_default_hsep(hsep_char)
```

Arguments

`hsep_char`	character(1). Character that will be set in the R environment options as default for creating the horizontal separator. It needs to be 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.

Examples

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

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

Value

a list of subobjects, which will be further paginated by the standard pagination algorithm.

export_as_pdf *Export as PDF*

Description

The PDF output 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 = NULL,
  height = NULL,
  margins = c(4, 4, 4, 4),
  min_siblings = 2,
```

```
  font_family = "Courier",
  font_size = 8,
  fontsize = font_size,
  paginate = TRUE,
  lpp = NULL,
  cpp = NULL,
  hsep = "-",
  indent_size = 2,
  tf_wrap = TRUE,
  max_width = NULL,
  colwidths = propose_column_widths(x)
)
```

Arguments

x	ANY. The table-like object to export. Must have an applicable matrix_form method.
file	file to write, must have .pdf extension
page_type	character(1). Name of a page type. See page_types. Ignored when pg_width and pg_height are set directly.
landscape	logical(1). Should the dimensions of page_type be inverted for landscape? 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 pg_width or specify page_type. The width of the graphics region in inches
height	Deprecated, please use pg_height or specify page_type. The height of the graphics region in inches
margins	numeric(4). The number of lines/characters of margin on the bottom, left, top, and right sides of the page.
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.
font_family	character(1). 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 font_size. The size of text (in points)
paginate	logical(1). Whether pagination should be performed, defaults to TRUE if page size is specified (including the default).
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	character(1). Characters to repeat to create header/body separator line. If NULL, the object value will be used. If " ", an empty separator will be printed. Check 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	logical(1). Should the texts for title, subtitle, and footnotes be wrapped?
max_width	integer(1), character(1) 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. Ignored completely if tf_wrap is FALSE.
colwidths	numeric vector. Column widths (in characters) for use with vertical pagination.

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 table to* RTF

Description

Experimental export to the RTF format.

Usage

```
export_as_rtf(
  x,
  file = NULL,
  colwidths = propose_column_widths(matrix_form(x, TRUE)),
  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_size = 8,
  font_family = "Courier",
  ...
)
```

Arguments

x	ANY. The table-like object to export. Must have an applicable matrix_form method.
file	character(1) or NULL. If non-NULL, the path to write a text file to containing the x rendered as ASCII text,
colwidths	numeric vector. Column widths (in characters) for use with vertical pagination.
page_type	character(1). 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	logical(1). Should the dimensions of page_type be inverted for landscape? 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_size	numeric(1). Font size, defaults to 12.
font_family	character(1). Name of a font family. An error will be thrown if the family named is not monospaced. Defaults to Courier.
...	Passed to individual methods.

Details

RTF export occurs by via the following steps

- the table is paginated to the page size (Vertically and horizontally)
- Each separate page is converted to a MatrixPrintForm and from there to RTF-encoded text
- Separate RTFs text chunks are combined and written out as 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 break*

Description

This function converts x to a MatrixPrintForm object via matrix_form, paginates it via paginate, converts each page to ASCII text via toString, and emits the strings to file, separated by page_break.

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 = num_rep_cols(x),
  verbose = FALSE,
  page_break = "\\s\\n"
)
```

Arguments

x	ANY. The table-like object to export. Must have an applicable matrix_form method.
file	character(1) or NULL. If non-NULL, the path to write a text file to containing the x rendered as ASCII text,
page_type	character(1). Name of a page type. See page_types. Ignored when pg_width and pg_height are set directly.

landscape	logical(1). Should the dimensions of page_type be inverted for landscape? 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	character(1). 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	logical(1). 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.
...	Passed to individual methods.
hsep	character(1). Characters to repeat to create header/body separator line. If NULL, the object value will be used. If " ", an empty separator will be printed. Check 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	logical(1). Should the texts for title, subtitle, and footnotes be wrapped?
max_width	integer(1), character(1) 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. Ignored completely if tf_wrap is FALSE.
colwidths	numeric vector. Column widths (in characters) for use with 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.
nosplitin	character. List of names of sub-tables 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	logical(1). Should additional informative messages about the search for pagination breaks be shown. Defaults to FALSE.
page_break	character(1). Page break symbol (defaults to outputting "\n\s").

Details

if x has an num_rep_cols method, the value returned by it will be used for rep_cols by default, if not, 0 will be used.

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

Value

if file is NULL, the total paginated and then concatenated string value, otherwise the file that was written.

Examples

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

---

ex_adsl *Simulated* CDISC *Alike Data for Examples*

Description

Simulated CDISC Alike 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	character(1) or function. A format label (string) or formatter function.
na_str	character(1). String that should be displayed in place of missing values.
align	character(1). 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")
```

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	character(1) or function. The format label (string) or formatter function to apply to x.
output	character(1). output type
na_str	character(1). String that should be displayed when the value of x is missing. Defaults to "NA".

Details

A length-zero value for na_str will be interpreted as "NA", as will any missing values within a non-length-zero na_str vector.

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")
```

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 length 0
b	ANY. Element to select if a is length 0

Value

a, unless it is length 0, in which case b (even in the case b is also length 0)

Examples

```
6 %||% 10

character() %||% "hi"

NULL %||% "hi"
```

is.wholenumber is.wholenumber

Description

is.wholenumber

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(5.0000000000000001)
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 with currently supported formats and vertical alignments*

Description

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

Usage

```
list_valid_format_labels()

list_valid_aligns()
```

Value

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

Examples

```
list_valid_format_labels()

list_valid_aligns()
```

main_title *General title/footer accessors*

Description

General title/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), a 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.frames. 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
)

## 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
)
```

Arguments

tt	ANY. Object representing the table-like object to be summarized.
colwidths	numeric. Internal detail do not set manually.
visible_only	logical(1). 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	logical(1). 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.

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 rtables::make_col_df), as it is simply the row/column structure of tt and thus not useful for pathing or pagination.

MatrixPrintForm *Matrix Print Form - Intermediate Representation for ASCII Table Printing*

Description

Matrix Print Form - Intermediate Representation for ASCII Table Printing

Usage

```
MatrixPrintForm(
  strings = NULL,
  spans,
  aligns,
  formats,
  row_info,
  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(),
  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
)

matrix_print_form(
  strings = NULL,
  spans,
  aligns,
  formats,
  row_info,
  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(),
  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
)
```

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 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 (XXX document exactly what that is).
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 expact_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	logical(1). Does the corresponding table have 'top left information' which should be treated differently when expanding newlines. Ignored if expand_newlines is FALSE.
has_rowlabs	logical(1). 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`
: logical(1). Should the matrix form generated expand rows whose values contain newlines into multiple 'physical' rows (as they will appear when rendered into ASCII). Defaults to `TRUE`

`main_title`
: character(1). Main title as a string.

`subtitles`
: character. Subtitles, as a character vector.

`page_titles`
: character. Page-specific titles, as a character vector.

`main_footer`
: character(1). Main footer as a string.

`prov_footer`
: character. Provenance footer information as a character vector.

`header_section_div`
: character(1). Divider to be used between header and body sections.

`horizontal_sep`
: character(1). 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. NULL, or a vector of 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.

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.

`MatrixPrintForm-class` *Matrix Print Form - Intermediate Representation for ASCII Table Printing*

Description

Matrix Print Form - 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
)

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

Arguments

obj	ANY. Object to be transformed into a ready-to-render form (a MatrixPrintForm object)
indent_rownames	logical(1), if TRUE the column with the row names in the strings matrix of has indented row names (strings pre-fixed)
expand_newlines	logical(1). Should the matrix form generated 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.

Details

The strings in the return object are defined as follows: row labels are those determined by summarize_rows and cell values are determined using get_formatted_cells. (Column labels are calculated using a non-exported internal function.

Value

A MatrixPrintForm classed list 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 summarize_rows(tt)

With an additional nrow_header attribute indicating the number of pseudo "rows" the column structure defines.

---

mf_strings Setters *and* getters *for aspects of* MatrixPrintForm *Objects*

Description

Most of these functions, particularly the settters, 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_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(1). A MatrixPrintForm object
value	ANY. The new value for the component in question.
x	MatrixPrintForm. The object.

Value

The element of the MatrixPrintForm associated with the getter, or the modified MatrixPrintForm object in the case of a setter.

---

mpf_to_rtf *Transform* MPF *to* RTF

Description

Experimental export to 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_size = 8,
  ...
)
```

Arguments

mpf	MatrixPrintForm. MatrixPrintForm object.
colwidths	character(1). Column widths.
page_type	character(1). 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	logical(1). Should the dimensions of page_type be inverted for landscape? 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_size	numeric(1). Font size, defaults to 12.
...	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

---

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)

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

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

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

Arguments

x	ANY. The object to be printed
colwidths	numeric. Column widths (if necessary).
max_width	numeric(1). Width strings should be wrapped to when determining how many lines they require.

Value

A scalar numeric indicating 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 should return 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)
```

Arguments

obj	ANY. A table-like object.

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)
```

padstr *Pad a string and align within string*

Description

Pad a string and align within string

Usage

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

Arguments

x	string
n	number of character of the output string, if n < nchar(x) an error is thrown
just	character(1). Text alignment justification to use. Defaults to center. Must be center, right, left, dec_right, dec_left or decimal.

Value

x, padded to be a string of n characters

Examples

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

if (interactive()) {
  padstr("abc", 1)
}
```

pagdfrow *Create row of pagination data frame*

Description

Create row of 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),
  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_
)
```

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, respectively, for default values of nm, lab and extent to be retrieved, respectively.
nm	character(1). Name
lab	character(1). 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	character(1). 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	logical(1). Currently Ignored.
page_title	logical(1). Currently Ignored.
trailing_sep	character(1). The string to used as a separator below this row during printing (or NA_character_ for no separator).

Value

a single row data.frame with the columns appropriate 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
)
```

Arguments

page_type	character(1). Name of a page type. See page_types. Ignored when pg_width and pg_height are set directly.
landscape	logical(1). Should the dimensions of page_type be inverted for landscape? Defaults to FALSE, ignored when pg_width and pg_height are set directly.
font_family	character(1). 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.

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* TypesList *supported named page types*

Description

Supported Named Page TypesList supported named page types

Usage

```
page_types()

page_dim(page_type)
```

Arguments

page_type	character(1). The name of a page size specification. Call page_types for supported values.

Value

for page_types a character vector of supported page types, for page_dim 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.

`paginate_to_mpfs` renders `obj` into the `MatrixPrintForm` (`MPF`) intermediate representation, and then paginates that `MPF` into component `MPF`s each corresponding to an individual page and returns those in a list.

`paginate_indices` renders `obj` into an `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).

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

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 = 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,
  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,
  rep_cols = num_rep_cols(obj),
  col_gap = 2,
  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)),
  tf_wrap = FALSE,
  max_width = NULL,
  indent_size = 2,
  pg_size_spec = NULL,
  rep_cols = num_rep_cols(obj),
  col_gap = 2,
  verbose = FALSE,
  ...
)
```

Arguments

obj	ANY. object to be paginated. Must have a matrix_form method.

page_type	character(1). Name of a page type. See page_types. Ignored when pg_width and pg_height are set directly.
font_family	character(1). 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	logical(1). Should the dimensions of page_type be inverted for landscape? 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.
nosplitin	character. List of names of sub-tables where page-breaks are not allowed, regardless of other considerations. Defaults to none.
colwidths	numeric vector. Column widths (in characters) for use with vertical pagination.
tf_wrap	logical(1). Should the texts for title, subtitle, and footnotes be wrapped?
max_width	integer(1), character(1) 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. Ignored completely if tf_wrap is 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 in end user code.
rep_cols	numeric(1). Number of *columns* (not including row labels) to be repeated on every page. Defaults to 0
col_gap	numeric(1). Currently unused.
verbose	logical(1). Should additional informative messages about the search for pagination breaks be shown. Defaults to FALSE.
...	Passed to individual methods.

Details

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 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_pagination 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_pagination 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 more 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 total.

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

Value

for paginate_indices a list with two elements of the same length: pag_row_indices, and pag_col_indices. For paginate_to_mpfs, a list of MatrixPrintForm objects representing each individual page after pagination (including forced pagination if necessary).

For diagnose_pagination 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} \item{**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) already adjusted for rendered elements which are not rows/columns and a dataframe of pagination information, pagination is performed via the following algorithm, and with a start = 1:

Core Pagination Algorithm:

1. Initial guess for pagination point 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 ever be `< start` after decrementing

1. Retain pagination index
2. 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 Dataframe*

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,
  min_siblings,
  nosplitin = character(),
  verbose = FALSE,
  row = TRUE,
  have_col_fnotes = FALSE,
  div_height = 1L
)
```

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
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.
nosplitin	character. List of names of sub-tables where page-breaks are not allowed, regardless of other considerations. Defaults to none.
verbose	logical(1). Should additional informative messages about the search for pagination breaks be shown. Defaults to FALSE.
row	logical(1). Is pagination happening in row space (TRUE, the default) or column space (FALSE)
have_col_fnotes	logical(1). Does the table-like object being rendered have column-associated referential footnotes.
div_height	numeric(1). The height of the divider line when the associated object is rendered. Defaults to 1.

Details

pab_indices_inner implements the Core Pagination Algorithm for a single direction (vertical if row = TRUE, the default, horizontal otherwise) based on the pagination dataframe and (already adjusted for non-body rows/columns) lines (or characters) per page.

Value

A list containing the 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) already adjusted for rendered elements which are not rows/columns and a dataframe of pagination information, pagination is performed via the following algorithm, and with a start = 1:

Core Pagination Algorithm:

1. Initial guess for pagination point 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 ever be `< start` after decrementing

1. Retain pagination index
2. 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 `[base::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 an object's* MatrixPrintForm *form*

Description

The row names are also considered a column for the output

Usage

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

Arguments

x	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.

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 dataframe*

Description

Create a row for a referential footnote information dataframe

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 (NA_character_ for none)
col_path	character. column path (NA_character_ for none)
row	integer(1). Integer position of the row.
col	integer(1). Integer position of the column.
symbol	character(1). Symbol for the reference. NA_character_ to use the ref_index automatically.
ref_index	integer(1). The index of the footnote, used for ordering even when symbol is not NA
msg	character(1). The string message, not including the symbol portion ({symbol} -)
max_width	numeric(1). Width 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	character(1). The value to return if x is NA.

Details

This function combines the rounding behavior of R's standards-complaint 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 containing any trailling zeros required to display *exactly* `digits` elements.

Note

This differs from the base R `round` function in that `NA` digits indicate `x` should be passed 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

`link{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 vectors of spans (with duplication) to Visibility vector*

Description

Transform vectors of spans (with duplication) to 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 to such that each individual position covered by the span has the repeated value.

This means that each block of values in span 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 span two individual cells.

Value

a logical vector the same length as spans indicating whether the contents of a string vector with those spans

Note

Currently no checking or enforcement is done that the vector of spans is valid in the sense described in the Details section above.

Examples

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

---

spread_integer *spread* x *into* len *elements*

Description

spread x into len elements

Usage

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

Arguments

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

Value

if x is a scalar "whole number" value (see is.wholenumber), the value x repeated len times. If not, 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	character(1). A format string passed to sprintf.

Value

A formatting function which wraps and will apply the specified printf style format 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	character(1). String to use as new header/body separator.

Value

for table_inset 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. For table_inset<-, the obj, with the new table_inset value applied recursively to it and all its subtables.

toString *toString*

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 by toString. This function allows fundamental formatting specifications for the final output, like column widths and relative wrapping (width), title and footer wrapping (tf_wrap = TRUE and max_width), or 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
)
```

Arguments

x	ANY. Object to be prepared for rendering.
...	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() + 1 as the column of row names must also be considered.
tf_wrap	logical(1). Should the texts for title, subtitle, and footnotes be wrapped?
max_width	integer(1), character(1) 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. Ignored completely if tf_wrap is FALSE.
col_gap	numeric(1). Space (in characters) between columns
hsep	character(1). Characters to repeat to create header/body separator line. If NULL, the object value will be used. If " ", an empty separator will be printed. Check default_hsep() for more information.

Details

Manual insertion of newlines is not supported when tf_wrap is on 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 (empty sting if not specified)

Usage

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

Arguments

x	a data.frame object
fill	boolean in case the label attribute does not exist if TRUE the variable names is returned, otherwise NA

Value

a named character vector with the variable labels, the names correspond to the 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 a 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	a data.frame object
value	new variable labels, NA removes the variable label

Value

modifies the variable labels of x

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

Removing labels attributes from a variables in a data frame

Usage

```
var_labels_remove(x)
```

Arguments

x	a data.frame object

Value

the same data frame as x stripped of variable labels

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

Value

a copy of x with changed labels 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
)
```

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. Column widths (in characters) for use with vertical pagination.
verbose	logical(1). Should additional informative messages about the search for pagination breaks 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

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)
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	an object
label	label attribute to to attached to x

Value

x labeled by label. Note: the exact mechanism of labeling should be considered an internal implementation detail, but the label will 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 within a precise width*

Description

Core wrapping functionality that preserve white spaces. Only "\n" is not supported by core functionality stringi::stri_wrap(). This is usually solved before hand 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)

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

Arguments

str	character. String to be wrapped. If it is a character 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 at.
collapse	character(1) or NULL. If the words that have been split should be pasted together with the collapse character. This is usually done internally with "\n" to have the wrapping updated along with other internal values.

Details

Word wrapping happens as with stringi::stri_wrap() with the following exception: individual words which are longer than max_width are broken up in a way that fits with the rest of the word wrapping.

Value

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

Functions

- wrap_txt(): function that flattens the list of wrapped strings with unist(str, use.names = FALSE). This is deprecated, 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