Package ‘formatters’

May 25, 2023

Title ASCII Formatting for Values and Tables
Version 0.5.0
Date 2023-04-19
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, grid, htmltools
Suggests dplyr, gt (>= 0.7.0), huxtable, knitr, r2rtf, rmarkdown, testthat
VignetteBuilder knitr
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' 'page_size.R' 'pagination.R'
'tostring.R' 'utils.R'
NeedsCompilation no
Author Gabriel Becker [aut, cre],
Adrian Waddell [aut],
Davide Garolini [ctb],
F. Hoffmann-La Roche AG [cph, fnd]
Maintainer Gabriel Becker <gabembecker@gmail.com>
Repository CRAN
Date/Publication 2023-05-25 07:40:06 UTC
# R topics documented:

- `basic_matrix_form` .................................................. 3
- `basic_pagdf` ......................................................... 3
- `default_hsep` ......................................................... 4
- `divider_height` ....................................................... 5
- `DM` ................................................................ 5
- `do_forced_paginate` .................................................. 6
- `export_as_rtf` .......................................................... 6
- `export_as_txt` ........................................................... 7
- `ex_adsl` ................................................................ 10
- `font_lcpi` ................................................................ 11
- `format_value` ............................................................. 12
- `ifnotlen0` ................................................................ 13
- `is.wholenumber` .......................................................... 13
- `is_valid_format` ........................................................... 14
- `lab_name` ................................................................ 15
- `list_valid_format_labels` .............................................. 16
- `main_title` ................................................................ 16
- `make_row_df` ............................................................... 18
- `MatrixPrintForm` ......................................................... 20
- `MatrixPrintForm-class` ................................................. 23
- `matrix_form` ............................................................... 23
- `mf_strings` ................................................................. 24
- `mpf_to_rtf` ................................................................. 26
- `nlines` .................................................................... 27
- `num_rep_cols` ............................................................. 28
- `padstr` .................................................................... 28
- `pagdfrow` .................................................................. 29
- `page_lcpp` ................................................................. 31
- `page_types` ................................................................. 32
- `paginate_indices` ......................................................... 33
- `pagination_algo` ......................................................... 37
- `pag_indices_inner` ..................................................... 38
- `print,ANY-method` .................................................... 40
- `propose_column_widths` .............................................. 41
- `ref_df_row` ................................................................. 41
- `round_fmt` ................................................................. 42
- `spans_to_viscell` ......................................................... 43
- `spread_integer` ........................................................... 44
- `sprintf_format` ........................................................... 45
- `table_inset` ............................................................... 46
- `toString` ................................................................. 46
- `var_labels` ................................................................. 48
- `var_labels<-` ............................................................. 48
- `var_labels_remove` ..................................................... 49
- `var_relabel` ............................................................... 50
- `vert_pag_indices` ........................................................ 50
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 data.frame

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"))
```

---

**default_hsep**

__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()
```

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()
```
**divider_height**

---

<table>
<thead>
<tr>
<th>divider_height</th>
<th>Divider Height</th>
</tr>
</thead>
</table>

**Description**

Divider Height

**Usage**

```r
divider_height(obj)
```

```r
# 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**

```r
divider_height(mtcars)
```

---

<table>
<thead>
<tr>
<th>DM</th>
<th>DM data</th>
</tr>
</thead>
</table>

**Description**

DM data

**Usage**

```r
DM
```

**Format**

```r
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)
```

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_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",
  ...
)
```
export_as_txt

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 rtf's 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

```r
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 = default_hsep(),
  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.
export_as_txt

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.

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)
Simulated CDISC Alike Data for Examples

Description
Simulated CDISC Alike Data for Examples

Usage
ex_ads1
ex_adae
ex_adette
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.
font_lcpi

Calculate lines per inch and characters per inch for font

Description

Calculate lines per inch and characters per inch for font

Usage

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

Arguments

- `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.

Details

This function creates opens pdf graphics device writing to an temporary file, then utilizes `grid::convertWidth()` and `grid::convertHeight()` to calculate lines per inch and characters per inch for the specified font family, size, and line height.

An error is thrown if the font is not monospaced (determined by comparing the effective widths of the M and . glyphs).

Value

named list with `cpi` and `lpi`, the characters and lines per inch, respectively.

Examples

```r
font_lcpi()
font_lcpi(font_size = 8)
font_lcpi(font_size = 8, lineheight = 1.1)
```
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.
is_valid_format

Examples

is.wholenumber(5)
is.wholenumber(5.0000000000000001)
is.wholenumber(.5)

is_valid_format(x, stop_otherwise = FALSE)

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

Value

TRUE if x is NULL, a supported format string, or a function; FALSE otherwise.

Note

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

Examples

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

Getters and setters for basic, relatively universal attributes of “table-like” objects

Usage

```r
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)

obj_format(obj) <- value

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

obj_na_str(obj)

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

obj_na_str(obj) <- value

## S4 replacement method for signature 'ANY'
obj_na_str(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_valid_format_labels

List with currently support 'xx' style format labels grouped by 1d, 2d and 3d

---

Description

Currently valid format labels can not be added dynamically. Format functions must be used for special cases

Usage

list_valid_format_labels()  

Value

A nested list, with elements listing the supported 1d, 2d, and 3d format strings.

Examples

list_valid_format_labels()  

---

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, 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 and column layout summary data.frames for use during pagination

Description

Make row and column layout summary data.frames for use during pagination

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
)
**make_row_df**

**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 dby both `make_row_df` and `make_col_df`, as it is simply the row/column structure of tt and thus not useful for pathing or pagination.
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(),
    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,
MatrixPrintForm

main_title = "",  
subtitles = character(),
page_titles = character(),
main_footer = "",  
prov_footer = character(),
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.

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 expect_newlines is set to FALSE.

ref_fnotes  list. Referential footnote information if applicable.

nenlines_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.
MatrixPrintForm

- 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.
- 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-NUL, 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
- 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.
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.
**mf_strings**

**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 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_strings

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.
\textit{mpf_to_rtf}

\begin{itemize}
\item \textbf{x} MatrixPrintForm. The object.
\end{itemize}

\section*{Value}

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

\section*{mpf_to_rtf \quad Transform MPF to RTF}

\section*{Description}

Experimental export to RTF via the \texttt{r2rtf} package

\section*{Usage}

\begin{verbatim}
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,  
  ...
)
\end{verbatim}

\section*{Arguments}

\begin{itemize}
\item \texttt{mpf} MatrixPrintForm. MatrixPrintForm object.
\item \texttt{colwidths} character(1). Column widths.
\item \texttt{page_type} character(1). Name of a page type. See \texttt{page_types}. Ignored when \texttt{pg_width} and \texttt{pg_height} are set directly.
\item \texttt{pg_width} numeric(1). Page width in inches.
\item \texttt{pg_height} numeric(1). Page height in inches.
\item \texttt{landscape} logical(1). Should the dimensions of \texttt{page_type} be inverted for landscape? Defaults to FALSE, ignored when \texttt{pg_width} and \texttt{pg_height} are set directly.
\item \texttt{margins} numeric(4). Named numeric vector containing \texttt{"bottom"}, \texttt{"left"}, \texttt{"top"}, and \texttt{"right"} margins in inches. Defaults to .5 inches for both vertical margins and .75 for both horizontal margins.
\item \texttt{font_size} numeric(1). Font size, defaults to 12.
\item \ldots Passed to individual methods.
\end{itemize}
**nlines**

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

```r
nlines(x, colwidths = NULL, max_width = NULL)
nlines(x, colwidths = NULL, max_width = NULL)
nlines(x, colwidths = NULL, max_width = NULL)
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**

```r
num_rep_cols(obj)
```

```r
## 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**

```r
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**

```r
padstr(x, n, just = c("center", "left", "right"))
```
Arguments

- \texttt{x}: \text{string}  \\
  number of character of the output string, if \( n < nchar(x) \) an error is thrown

- \texttt{just}: \text{character(1)}  \\
  Text alignment justification to use. Defaults to center. Must be center, right or left.

Value

- \texttt{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)
}

---

\texttt{pagdfrow} \hspace{1cm} \textit{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 rownumber
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
replext 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 Types

List supported named page types

Description

Supported Named Page Types

List 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 MPFs 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",
  ...
paginate_indices

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.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>page_type</td>
<td>character(1). Name of a page type. See page_types. Ignored when pg_width</td>
</tr>
<tr>
<td></td>
<td>and pg_height are set directly.</td>
</tr>
<tr>
<td>font_family</td>
<td>character(1). Name of a font family. An error will be thrown if the family</td>
</tr>
<tr>
<td></td>
<td>named is not monospaced. Defaults to Courier.</td>
</tr>
<tr>
<td>font_size</td>
<td>numeric(1). Font size, defaults to 12.</td>
</tr>
<tr>
<td>lineheight</td>
<td>numeric(1). Line height, defaults to 1.</td>
</tr>
<tr>
<td>landscape</td>
<td>logical(1). Should the dimensions of page_type be inverted for landscape?</td>
</tr>
<tr>
<td></td>
<td>Defaults to FALSE, ignored when pg_width and pg_height are set directly.</td>
</tr>
<tr>
<td>pg_width</td>
<td>numeric(1). Page width in inches.</td>
</tr>
<tr>
<td>pg_height</td>
<td>numeric(1). Page height in inches.</td>
</tr>
<tr>
<td>margins</td>
<td>numeric(4). Named numeric vector containing 'bottom', 'left', 'top', and</td>
</tr>
<tr>
<td></td>
<td>'right' margins in inches. Defaults to .5 inches for both vertical margins</td>
</tr>
<tr>
<td></td>
<td>and .75 for both horizontal margins.</td>
</tr>
<tr>
<td>lpp</td>
<td>numeric(1) or NULL. Lines per page. if NA (the default, this is calculated</td>
</tr>
<tr>
<td></td>
<td>automatically based on the specified page size). NULL indicates no vertical</td>
</tr>
<tr>
<td></td>
<td>pagination should occur.</td>
</tr>
<tr>
<td>cpp</td>
<td>numeric(1) or NULL. Width in characters per page. if NA (the default, this is</td>
</tr>
<tr>
<td></td>
<td>calculated automatically based on the specified page size). NULL indicates</td>
</tr>
<tr>
<td></td>
<td>no horizontal pagination should occur.</td>
</tr>
<tr>
<td>min_siblings</td>
<td>numeric. Minimum sibling rows which must appear on either side of pagination</td>
</tr>
<tr>
<td></td>
<td>row for a mid-subtable split to be valid. Defaults to 2.</td>
</tr>
<tr>
<td>nosplitin</td>
<td>character. List of names of sub-tables where page-breaks are not allowed,</td>
</tr>
<tr>
<td></td>
<td>regardless of other considerations. Defaults to none.</td>
</tr>
<tr>
<td>colwidths</td>
<td>numeric vector. Column widths (in characters) for use with vertical pagination</td>
</tr>
<tr>
<td>tf_wrap</td>
<td>logical(1). Should the texts for title, subtitle, and footnotes be Wrapped?</td>
</tr>
<tr>
<td>max_width</td>
<td>integer(1), character(1) or NULL. Width that title and footer (including</td>
</tr>
<tr>
<td></td>
<td>footnotes) materials should be word-wrapped to. If NULL, it is set to the</td>
</tr>
<tr>
<td></td>
<td>current print width of the session (getOption(&quot;width&quot;)). If set to &quot;auto&quot;,</td>
</tr>
<tr>
<td></td>
<td>the width of the table (plus any table inset) is used. Ignored completely if</td>
</tr>
<tr>
<td></td>
<td>tf_wrap is FALSE.</td>
</tr>
</tbody>
</table>
| indent_size     | numeric(1). Indent size in characters. Ignored when x is already a MatrixPrint-
|                 | Form 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-pagination`. `paginate_indices` does call `do_forced-pagination`, 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
  - `col_diagnostics` (very) basic information about leaf columns, whether pagination was attempted after each leaf column, and 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

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

pag_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

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

---

**Description**

Print an R object. see [base::print()]

**Usage**

```r
## 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 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 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

```r
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

```r
spans_to_viscell(spans)
```

Arguments

spans numeric. 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(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.
### sprintf_format

Specify text format via a sprintf format string

#### Description

Specify text format via a sprintf format string

#### Usage

```r
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

```r
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)
```

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

```
    table_inset(obj) <- value
```

```r
    # 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.
Usage

toString(x, ...)

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

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.

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

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 function returns a named character vector with the variable labels (empty sting if not specified).

**Usage**

```r
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**

```r
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 function sets all non-missing (non-NA) variable labels in a data.frame

**Usage**

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

**Arguments**

- `x`: a data.frame object
- `value`: new variable labels, NA removes the variable label
var_labels_remove

Value

modifies the variable labels of x

Examples

```r
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**

```r
var_labels_remove(x)
```

**Arguments**

- **x** a data.frame object

**Value**

the same data frame as x stripped of variable labels

**Examples**

```r
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
)
```
with_label

Arguments

obj \quad \text{ANY. object to be paginated. Must have a matrix_form method.}
cpp \quad \text{numeric(1). Number of characters per page (width) }
colwidths \quad \text{numeric vector. Column widths (in characters) for use with vertical pagination.}
verbose \quad \text{logical(1). Should additional informative messages about the search for pagination breaks be shown. Defaults to FALSE.}
rep_cols \quad \text{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

```r
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

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

Arguments

x \quad \text{an object}
label \quad \text{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

```r
x <- with_label(c(1, 2, 3), label = "Test")
obj_label(x)
```
Wrap a string to within a maximum width

Description

Wrap a string to within a maximum width

Usage

wrap_string(str, max_width, hard = FALSE)

wrap_txt(txt, max_width, hard = FALSE)

Arguments

str character(1). String to be wrapped
max_width numeric(1). Maximum width, in characters, that the text should be wrapped at.
hard logical(1). Should hard wrapping (embedding newlines in the incoming strings) or soft (breaking wrapped strings into vectors of length >1) be used. Defaults to FALSE (i.e. soft wrapping).
txt character. Vector of strings that should be (independently) text-wrapped.

Details

Word wrapping happens as with base::strwrap 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 (wrap_string or character vector (wrap_txt) containing the hard or soft word-wrapped content.
Index

* datasets
  DM, 5
  ex_adsl, 10

all_footers (main_title), 16
all_titles (main_title), 16

base::strwrap, 52
basic_matrix_form, 3
basic_pagdf, 3

default_hsep, 4
diagnose_pagination (paginate_indices), 33
divider_height, 5
divider_height, ANY-method
  (divider_height), 5
DM, 5
do_forced_paginate, 6
do_forced_paginate, ANY-method
  (do_forced_paginate), 6

ex_aqae (ex_adsl), 10
ex_adadette (ex_adsl), 10
ex_adcm (ex_adsl), 10
ex_adlb (ex_adsl), 10
ex_adhs (ex_adsl), 10
ex_adhs (ex_adsl), 10
ex_adsl, 10
ex_adte (ex_adsl), 10
ex_adtes (ex_adsl), 10
export_as_rtf, 6
export_as_txt, 7

font_lcp, 11
format_value, 12, 42

grid::convertHeight(), 11
grid::convertWidth(), 11

ifnotlen0, 13
is.wholenumber, 13, 44
is_valid_format, 14

lab_name, 15
list_valid_format_labels, 16

main_footer (main_title), 16
main_footer,MatrixPrintForm-method
  (main_title), 16
main_footer<-(main_title), 16
main_footer<-,MatrixPrintForm-method
  (main_title), 16
main_title, 16
main_title,MatrixPrintForm-method
  (main_title), 16
main_title<-(main_title), 16
main_title<-,MatrixPrintForm-method
  (main_title), 16
make_row_df, 18
matrix_form, 23, 34, 51
matrix_form,MatrixPrintForm-method
  (matrix_form), 23
matrix_print_form (MatrixPrintForm), 20
MatrixPrintForm, 20
MatrixPrintForm-class, 23
mf_aligns (mf_strings), 24
mf_aligns<-(mf_strings), 24
mf_cinfo (mf_strings), 24
mf_cinfo<-(mf_strings), 24
mf_colgap (mf_strings), 24
mf_colgap<-(mf_strings), 24
mf_display (mf_strings), 24
mf_display<-(mf_strings), 24
mf_formats (mf_strings), 24
mf_formats<-(mf_strings), 24
mf_has_rlabels (mf_strings), 24
mf_has_topleft (mf_strings), 24
mf_lgrouping (mf_strings), 24
mf_lgrouping<-(mf_strings), 24
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>mf_ncol(mf_strings)</code></td>
<td>24</td>
</tr>
<tr>
<td><code>mf_ncol&lt;-</code> (mf_strings)</td>
<td>24</td>
</tr>
<tr>
<td><code>mf_nlheader(mf_strings)</code></td>
<td>24</td>
</tr>
<tr>
<td><code>mf_nrheader(mf_strings)</code></td>
<td>24</td>
</tr>
<tr>
<td><code>mf_nrheader&lt;-</code> (mf_strings)</td>
<td>24</td>
</tr>
<tr>
<td><code>mf_nrow(mf_strings)</code></td>
<td>24</td>
</tr>
<tr>
<td><code>mf_rfnotes(mf_strings)</code></td>
<td>24</td>
</tr>
<tr>
<td><code>mf_rfnotes&lt;-</code> (mf_strings)</td>
<td>24</td>
</tr>
<tr>
<td><code>mf_rinfo(mf_strings)</code></td>
<td>24</td>
</tr>
<tr>
<td><code>mf_rinfo&lt;-</code> (mf_strings)</td>
<td>24</td>
</tr>
<tr>
<td><code>mf_spans(mf_strings)</code></td>
<td>24</td>
</tr>
<tr>
<td><code>mf_spans&lt;-</code> (mf_strings)</td>
<td>24</td>
</tr>
<tr>
<td><code>mf_strings</code></td>
<td>24</td>
</tr>
<tr>
<td><code>mf_strings&lt;-</code> (mf_strings)</td>
<td>24</td>
</tr>
<tr>
<td><code>mpf_has_rlabels(mf_strings)</code></td>
<td>24</td>
</tr>
<tr>
<td><code>mpf_to_rtf()</code></td>
<td>7</td>
</tr>
<tr>
<td><code>ncol</code>, <code>MatrixPrintForm-method</code></td>
<td>24</td>
</tr>
<tr>
<td><code>nlines</code>, <code>character-method(nlines)</code></td>
<td>27</td>
</tr>
<tr>
<td><code>nlines</code>, <code>list-method(nlines)</code></td>
<td>27</td>
</tr>
<tr>
<td><code>nlines</code>, <code>NULL-method(nlines)</code></td>
<td>27</td>
</tr>
<tr>
<td><code>num_rep_cols</code>, <code>ANY-method(num_rep_cols)</code></td>
<td>28</td>
</tr>
<tr>
<td><code>obj_format(lab_name)</code></td>
<td>15</td>
</tr>
<tr>
<td><code>obj_format</code>, <code>ANY-method(lab_name)</code></td>
<td>15</td>
</tr>
<tr>
<td><code>obj_format&lt;-</code>, <code>ANY-method(lab_name)</code></td>
<td>15</td>
</tr>
<tr>
<td><code>obj_label(lab_name)</code></td>
<td>15</td>
</tr>
<tr>
<td><code>obj_label</code>, <code>ANY-method(lab_name)</code></td>
<td>15</td>
</tr>
<tr>
<td><code>obj_label&lt;-</code> (lab_name)</td>
<td>15</td>
</tr>
<tr>
<td><code>obj_label&lt;-</code>, <code>ANY-method(lab_name)</code></td>
<td>15</td>
</tr>
<tr>
<td><code>obj_na_str(lab_name)</code></td>
<td>15</td>
</tr>
<tr>
<td><code>obj_na_str</code>, <code>ANY-method(lab_name)</code></td>
<td>15</td>
</tr>
<tr>
<td><code>obj_na_str&lt;-</code> (lab_name)</td>
<td>15</td>
</tr>
<tr>
<td><code>obj_na_str&lt;-</code>, <code>ANY-method(lab_name)</code></td>
<td>15</td>
</tr>
<tr>
<td><code>obj_name(lab_name)</code></td>
<td>15</td>
</tr>
<tr>
<td><code>obj_name&lt;-</code> (lab_name)</td>
<td>15</td>
</tr>
<tr>
<td><code>padstr</code></td>
<td>28</td>
</tr>
<tr>
<td><code>pag_indices_inner</code></td>
<td>38</td>
</tr>
<tr>
<td><code>pagdFrow</code></td>
<td>29</td>
</tr>
<tr>
<td><code>page_dim(page_types)</code></td>
<td>32</td>
</tr>
<tr>
<td><code>page_lcpp</code></td>
<td>31</td>
</tr>
<tr>
<td><code>page_titles</code>, <code>ANY-method(main_title)</code></td>
<td>16</td>
</tr>
<tr>
<td><code>page_titles</code>, <code>MatrixPrintForm-method</code></td>
<td>16</td>
</tr>
<tr>
<td><code>page_titles</code>, <code>MatrixPrintForm-method</code></td>
<td>16</td>
</tr>
<tr>
<td><code>page_titles&lt;-</code> (main_title)</td>
<td>16</td>
</tr>
<tr>
<td><code>page_titles&lt;-</code>, <code>MatrixPrintForm-method</code></td>
<td>16</td>
</tr>
<tr>
<td><code>page_types</code></td>
<td>32</td>
</tr>
<tr>
<td><code>paginate</code>, <code>paginate_indices</code></td>
<td>33</td>
</tr>
<tr>
<td><code>paginate</code>, <code>paginate_indices</code></td>
<td>33</td>
</tr>
<tr>
<td><code>pagination</code>, <code>paginate_indices</code></td>
<td>33</td>
</tr>
<tr>
<td><code>pagination_algo</code></td>
<td>37</td>
</tr>
<tr>
<td><code>print</code>, <code>ANY-method</code></td>
<td>40</td>
</tr>
<tr>
<td><code>propose_column_widths</code></td>
<td>41</td>
</tr>
<tr>
<td><code>prov_footer</code>, <code>main_title</code></td>
<td>16</td>
</tr>
<tr>
<td><code>prov_footer</code>, <code>main_title</code></td>
<td>16</td>
</tr>
<tr>
<td><code>prov_footer&lt;-</code> (main_title)</td>
<td>16</td>
</tr>
<tr>
<td><code>prov_footer&lt;-</code>, <code>MatrixPrintForm-method</code></td>
<td>16</td>
</tr>
<tr>
<td><code>ref_df_row</code></td>
<td>41</td>
</tr>
<tr>
<td><code>round</code>, 42, 43</td>
<td></td>
</tr>
<tr>
<td><code>round</code>, <code>round_fmt()</code></td>
<td>42</td>
</tr>
<tr>
<td><code>round</code>, <code>round_fmt()</code></td>
<td>42</td>
</tr>
<tr>
<td><code>rounding</code>, <code>round_fmt()</code></td>
<td>42</td>
</tr>
<tr>
<td><code>spans_to_viscell</code></td>
<td>43</td>
</tr>
<tr>
<td><code>spread_integer</code></td>
<td>44</td>
</tr>
<tr>
<td><code>sprintf</code>, 42, 43, 45</td>
<td></td>
</tr>
<tr>
<td><code>sprintf</code>, <code>sprintf_format</code></td>
<td>45</td>
</tr>
<tr>
<td><code>subtitles</code>, <code>main_title</code></td>
<td>16</td>
</tr>
<tr>
<td><code>subtitles</code>, <code>main_title</code></td>
<td>16</td>
</tr>
<tr>
<td><code>subtitles&lt;-</code>, <code>MatrixPrintForm-method</code></td>
<td>16</td>
</tr>
<tr>
<td><code>subtitles&lt;-</code>, <code>MatrixPrintForm-method</code></td>
<td>16</td>
</tr>
<tr>
<td><code>table_inset</code>, 22, 46</td>
<td></td>
</tr>
<tr>
<td><code>table_inset</code>, <code>MatrixPrintForm-method</code></td>
<td>46</td>
</tr>
<tr>
<td><code>table_inset&lt;-</code>, <code>table_inset</code></td>
<td>46</td>
</tr>
<tr>
<td><code>table_inset&lt;-</code>, <code>MatrixPrintForm-method</code></td>
<td>46</td>
</tr>
<tr>
<td><code>toString</code>, 46</td>
<td></td>
</tr>
<tr>
<td><code>toString</code>, <code>MatrixPrintForm-method</code></td>
<td>46</td>
</tr>
<tr>
<td><code>toString</code>, <code>MatrixPrintForm-method</code></td>
<td>46</td>
</tr>
<tr>
<td><code>var_labels</code></td>
<td>48</td>
</tr>
</tbody>
</table>