

# Package ‘formatters’

August 25, 2023

**Title** ASCII Formatting for Values and Tables

**Version** 0.5.2

**Date** 2023-08-24

**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] (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** 2023-08-25 09:30:02 UTC

## R topics documented:

basic_matrix_form . . . . .	3
basic_pagdf . . . . .	4
check_formats . . . . .	5
decimal_align . . . . .	6
default_hsep . . . . .	7
divider_height . . . . .	7
DM . . . . .	8
do_forced_paginate . . . . .	8
export_as_rtf . . . . .	9
export_as_txt . . . . .	10
ex_adsl . . . . .	12
fnt_config . . . . .	13
font_lcp . . . . .	14
format_value . . . . .	15
ifnotlen0 . . . . .	16
is.wholenumber . . . . .	16
lab_name . . . . .	17
list_formats . . . . .	19
main_title . . . . .	19
make_row_df . . . . .	21
MatrixPrintForm . . . . .	22
MatrixPrintForm-class . . . . .	25
matrix_form . . . . .	26
mf_strings . . . . .	27
mpf_to_rtf . . . . .	29
nlines . . . . .	30
num_rep_cols . . . . .	30
padstr . . . . .	31
pagdfrow . . . . .	32
page_lcpp . . . . .	33
page_types . . . . .	35
paginate_indices . . . . .	35
pagination_algo . . . . .	40
pag_indices_inner . . . . .	41
print,ANY-method . . . . .	43
propose_column_widths . . . . .	43
ref_df_row . . . . .	44
round_fmt . . . . .	45
spans_to_viscell . . . . .	46
spread_integer . . . . .	47
sprintf_format . . . . .	48

<i>basic_matrix_form</i>	3
table_inset . . . . .	48
toString . . . . .	49
var_labels . . . . .	50
var_labels<- . . . . .	51
var_labels_remove . . . . .	52
var_relabel . . . . .	52
vert_pag_indices . . . . .	53
with_label . . . . .	54
wrap_string . . . . .	54
<b>Index</b>	<b>56</b>

---

<code>basic_matrix_form</code>	<i>Create spoof matrix form from a data.frame</i>
--------------------------------	---

---

### 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

<code>df</code>	data.frame
<code>parent_path</code>	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**

<code>rnames</code>	character. Vector of row names
<code>labs</code>	character. Vector of row labels (defaults to names)
<code>rnums</code>	integer. Vector of row numbers. Defaults to <code>seq_along(rnames)</code> .
<code>extents</code>	integer. Number of lines each row will take to print, defaults to 1 for all rows
<code>rclass</code>	character. Class(es) for the rows. Defaults to "NA"
<code>parent_path</code>	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	<i>Check if a format or alignment is supported</i>
---------------	--

---

**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**

<code>string_mat</code>	character matrix. String matrix component of matrix print form object.
<code>align_mat</code>	character matrix. Aligns matrix component of matrix print form object. Should contain either <code>dec_left</code> , <code>dec_right</code> or <code>decimal</code> 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_hsep	<i>Default horizontal Separator</i>
--------------	-------------------------------------

---

**Description**

The default horizontal separator character which can be displayed in the current char set 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	<i>Divider Height</i>
----------------	-----------------------

---

**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 <i>data</i>
----	----------------

---

**Description**

DM data

**Usage**

DM

**Format**

rds (data.frame)

---

do_forced_paginate	<i>Generic for Performing "Forced Pagination"</i>
--------------------	---

---

**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_rtf	<i>Export table to RTF</i>
---------------	----------------------------

---

## 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

<code>x</code>	ANY. The table-like object to export. Must have an applicable <code>matrix_form</code> method.
<code>file</code>	character(1) or NULL. If non-NULL, the path to write a text file to containing the <code>x</code> rendered as ASCII text.
<code>colwidths</code>	numeric vector. Column widths (in characters) for use with vertical pagination.
<code>page_type</code>	character(1). Name of a page type. See <code>page_types</code> . Ignored when <code>pg_width</code> and <code>pg_height</code> are set directly.
<code>pg_width</code>	numeric(1). Page width in inches.
<code>pg_height</code>	numeric(1). Page height in inches.
<code>landscape</code>	logical(1). Should the dimensions of <code>page_type</code> be inverted for landscape? Defaults to FALSE, ignored when <code>pg_width</code> and <code>pg_height</code> are set directly.
<code>margins</code>	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.
<code>font_size</code>	numeric(1). Font size, defaults to 12.
<code>font_family</code>	character(1). Name of a font family. An error will be thrown if the family named is not monospaced. Defaults to Courier.
<code>...</code>	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 = 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 <code>matrix_form</code> 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 <code>page_types</code> . Ignored when <code>pg_width</code> and <code>pg_height</code> are set directly.
landscape	logical(1). Should the dimensions of <code>page_type</code> be inverted for landscape? Defaults to FALSE, ignored when <code>pg_width</code> and <code>pg_height</code> 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.
indent_size	numeric(1). Indent size in characters. Ignored when x is already a <code>MatrixPrintForm</code> 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 ( <code>getOption("width")</code> ). If set to "auto", the width of the table (plus any table inset) is used. Ignored completely if <code>tf_wrap</code> 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 <i>columns</i> (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\_ads1

*Simulated CDISC Alike Data for Examples*

---

**Description**

Simulated CDISC Alike Data for Examples

**Usage**

ex\_ads1

ex\_adae

ex\_adaette

ex\_adtte

ex\_adcm

ex\_adlb

ex\_admh

ex\_adqs

ex\_adrs

ex\_adv

**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	<i>Format Configuration</i>
------------	-----------------------------

---

**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")
```

---

font_lcp	<i>Calculate lines per inch and characters per inch for font</i>
----------	--

---

### Description

Calculate lines per inch and characters per inch for font

### Usage

```
font_lcp(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

```
font_lcp()
font_lcp(font_size = 8)
font_lcp(font_size = 8, lineheight = 1.1)
```

---

format_value	<i>Converts a (possibly compound) value into a string using the format information</i>
--------------	--

---

### 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(5.000000000000000001)
is.wholenumber(.5)
```

---

lab_name	<i>Label, Name and Format accessor generics</i>
----------	---

---

**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**

<code>obj</code>	ANY. The object.
<code>value</code>	<code>character(1)</code> . 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	<i>List with currently supported formats and vertical alignments</i>
--------------	--

---

**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	<i>General title/footer accessors</i>
------------	---------------------------------------

---

**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	<i>Make row and column layout summary data.frames for use during pagination</i>
-------------	---

---

**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
)
```

**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.

<code>repr_ext</code>	integer(1). Internal detail do not set manually.
<code>repr_inds</code>	integer. Internal detail do not set manually.
<code>sibpos</code>	integer(1). Internal detail do not set manually.
<code>nsibs</code>	integer(1). Internal detail do not set manually.
<code>max_width</code>	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`, 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(),
    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(),
  col_gap = 3,
  table_inset = 0L,
  colwidths = NULL,
  indent_size = 2
)

```

### Arguments

<code>strings</code>	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).
<code>spans</code>	numeric matrix. Matrix of same dimension as <code>strings</code> giving the spanning information for each element. Must be repeated to match placeholders in <code>strings</code> .

<code>aligns</code>	character matrix. Matrix of same dimension as <code>strings</code> giving the text alignment information for each element. Must be repeated to match placeholders in <code>strings</code> . Must be a supported text alignment. See <a href="#">decimal_align</a> allowed values.
<code>formats</code>	matrix. Matrix of same dimension as <code>strings</code> giving the text format information for each element. Must be repeated to match placeholders in <code>strings</code> .
<code>row_info</code>	<code>data.frame</code> . <code>Data.frame</code> with row-information necessary for pagination (XXX document exactly what that is).
<code>line_grouping</code>	integer. Sequence of integers indicating how print lines correspond to semantic rows in the object. Typically this should not be set manually unless <code>expand_newlines</code> is set to <code>FALSE</code> .
<code>ref_fnotes</code>	list. Referential footnote information if applicable.
<code>nlines_header</code>	numeric(1). Number of lines taken up by the values of the header (i.e. not including the divider).
<code>nrow_header</code>	numeric(1). Number of <code>rows</code> corresponding to the header.
<code>has_topleft</code>	logical(1). Does the corresponding table have 'top left information' which should be treated differently when expanding newlines. Ignored if <code>expand_newlines</code> is <code>FALSE</code> .
<code>has_rowlabs</code>	logical(1). Do the matrices ( <code>strings</code> , <code>spans</code> , <code>aligns</code> ) each contain a column that corresponds with row labels (Rather than with table cell values). Defaults to <code>TRUE</code> .
<code>expand_newlines</code>	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 <code>TRUE</code>
<code>main_title</code>	character(1). Main title as a string.
<code>subtitles</code>	character. Subtitles, as a character vector.
<code>page_titles</code>	character. Page-specific titles, as a character vector.
<code>main_footer</code>	character(1). Main footer as a string.
<code>prov_footer</code>	character. Provenance footer information as a character vector.
<code>col_gap</code>	numeric(1). Space (in characters) between columns
<code>table_inset</code>	numeric(1). Table inset. See <a href="#">table_inset</a>
<code>colwidths</code>	numeric. <code>NULL</code> , or a vector of column rendering widths. if non- <code>NULL</code> , must have length equal to <code>ncol(strings)</code>
<code>indent_size</code>	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.

---

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

---

**Description**

Matrix Print Form - Intermediate Representation for ASCII Table Printing

---

matrix_form	<i>Transform rtable to a list of matrices which can be used for outputting</i>
-------------	--

---

### 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 setters, are intended almost exclusively for internal use in, e.g., matrix\_form methods, and should generally not be called by end users.

**Usage**

mf\_strings(mf)

mf\_spans(mf)

mf\_aligns(mf)

mf\_display(mf)

mf\_formats(mf)

mf\_rinfo(mf)

mf\_cinfo(mf)

mf\_has\_topleft(mf)

mf\_lgrouping(mf)

mf\_rfnotes(mf)

mf\_nlheader(mf)

mf\_nrheader(mf)

```
mf_colgap(mf)
mf_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	<i>Transform MPF to RTF</i>
------------	-----------------------------

---

**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	<i>Number of lines required to print a value</i>
--------	--

---

### 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	<i>Number of repeated columns</i>
--------------	-----------------------------------

---

### 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  $n < \text{nchar}(x)$  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,
  reptext = 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
repxt	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	<i>Determine lines per page (LPP) and characters per page (CPP) based on font and page type</i>
-----------	---

---

**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\_mpf 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\_mpf 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 <code>matrix_form</code> method.
page_type	character(1). Name of a page type. See <code>page_types</code> . Ignored when <code>pg_width</code> and <code>pg_height</code> 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 <code>page_type</code> be inverted for landscape? Defaults to FALSE, ignored when <code>pg_width</code> and <code>pg_height</code> 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 ( <code>getOption("width")</code> ). If set to "auto", the width of the table (plus any table inset) is used. Ignored completely if <code>tf_wrap</code> is FALSE.
indent_size	numeric(1). Indent size in characters. Ignored when <code>x</code> is already a <code>MatrixPrintForm</code> object in favor of information there.
pg_size_spec	<code>page_size_spec</code> . A pre-calculated page size specification. Typically this is not set in end user code.
rep_cols	numeric(1). Number of <i>columns</i> (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_mpf`s, 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, 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_mpf(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 *nosplit*



---

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 <code>make_rows_df</code> or <code>make_cols_df</code> .
rlpp	numeric. Maximum number of <i>row</i> 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**

```
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	<i>Round and prepare a value for display</i>
-----------	--

---

### 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

<code>x</code>	numeric(1). Value to format
<code>digits</code>	numeric(1). Number of digits to round to, or NA to convert to a character value with no rounding.
<code>na_str</code>	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 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**

```
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	<i>Transform vectors of spans (with duplication) to Visibility vector</i>
------------------	---

---

**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	<i>spread x into len elements</i>
----------------	-----------------------------------

---

**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	<i>Specify text format via a sprintf format string</i>
----------------	--

---

### 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	<i>Access or (recursively) set table inset.</i>
-------------	---

---

### 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

**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 <code>ncol() + 1</code> 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 ( <code>getOption("width")</code> ). If set to "auto", the width of the table (plus any table inset) is used. Ignored completely if <code>tf_wrap</code> 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 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	<i>Remove Variable Labels of a data.frame</i>
-------------------	---

---

**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	<i>Copy and Change Variable Labels of a data.frame</i>
-------------	--

---

**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	<i>Find Column Indices for Vertical Pagination</i>
------------------	--

---

### 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 <a href="#">matrix_form</a> 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 <i>columns</i> (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	<i>Return an object with a label attribute</i>
------------	--

---

**Description**

Return an object with a label attribute

**Usage**

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

**Arguments**

x	an object
label	label attribute to be 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	<i>Wrap a string to within a maximum width</i>
-------------	--

---

**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. A 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, 8
  - ex\_adsl, 12
  
- all\_footers (main\_title), 19
- all\_titles (main\_title), 19
  
- base::strwrap, 55
- basic\_matrix\_form, 3
- basic\_pagdf, 4
  
- check\_aligns (check\_formats), 5
- check\_formats, 5
  
- decimal\_align, 6, 24
- default\_hsep, 7
- diagnose\_pagination (paginate\_indices), 35
- divider\_height, 7
- divider\_height, ANY-method (divider\_height), 7
- DM, 8
- do\_forced\_paginate, 8
- do\_forced\_paginate, ANY-method (do\_forced\_paginate), 8
  
- ex\_adae (ex\_adsl), 12
- ex\_adaette (ex\_adsl), 12
- ex\_adcm (ex\_adsl), 12
- ex\_adlb (ex\_adsl), 12
- ex\_admh (ex\_adsl), 12
- ex\_adqs (ex\_adsl), 12
- ex\_adrs (ex\_adsl), 12
- ex\_adsl, 12
- ex\_adtte (ex\_adsl), 12
- ex\_advz (ex\_adsl), 12
- export\_as\_rtf, 9
- export\_as\_txt, 10
  
- fmt\_config, 13
- font\_lcp\_i, 14
  
- format\_value, 15, 45
  
- grid::convertHeight(), 14
- grid::convertWidth(), 14
  
- ifnotlen0, 16
- is.wholenumber, 16, 47
- is\_valid\_format (check\_formats), 5
  
- lab\_name, 17
- list\_formats, 19
- list\_valid\_aligns (list\_formats), 19
- list\_valid\_format\_labels (list\_formats), 19
  
- main\_footer (main\_title), 19
- main\_footer, MatrixPrintForm-method (main\_title), 19
- main\_footer<- (main\_title), 19
- main\_footer<-, MatrixPrintForm-method (main\_title), 19
- main\_title, 19
- main\_title, MatrixPrintForm-method (main\_title), 19
- main\_title<- (main\_title), 19
- main\_title<-, MatrixPrintForm-method (main\_title), 19
  
- make\_row\_df, 21
- matrix\_form, 26, 37, 53
- matrix\_form, MatrixPrintForm-method (matrix\_form), 26
- matrix\_print\_form (MatrixPrintForm), 22
- MatrixPrintForm, 6, 22
- MatrixPrintForm-class, 25
- mf\_aligns (mf\_strings), 27
- mf\_aligns<- (mf\_strings), 27
- mf\_cinfo (mf\_strings), 27
- mf\_cinfo<- (mf\_strings), 27
- mf\_colgap (mf\_strings), 27
- mf\_colgap<- (mf\_strings), 27



- mf\_display (mf\_strings), 27
- mf\_display<- (mf\_strings), 27
- mf\_formats (mf\_strings), 27
- mf\_formats<- (mf\_strings), 27
- mf\_has\_rlabels (mf\_strings), 27
- mf\_has\_topleft (mf\_strings), 27
- mf\_lgrouping (mf\_strings), 27
- mf\_lgrouping<- (mf\_strings), 27
- mf\_ncol (mf\_strings), 27
- mf\_ncol<- (mf\_strings), 27
- mf\_nlheader (mf\_strings), 27
- mf\_nrheader (mf\_strings), 27
- mf\_nrheader<- (mf\_strings), 27
- mf\_nrow (mf\_strings), 27
- mf\_rfnodes (mf\_strings), 27
- mf\_rfnodes<- (mf\_strings), 27
- mf\_rinfo (mf\_strings), 27
- mf\_rinfo<- (mf\_strings), 27
- mf\_spans (mf\_strings), 27
- mf\_spans<- (mf\_strings), 27
- mf\_strings, 27
- mf\_strings<- (mf\_strings), 27
- mpf\_has\_rlabels (mf\_strings), 27
- mpf\_to\_rtf, 29
- mpf\_to\_rtf(), 10
  
- ncol, MatrixPrintForm-method  
(mf\_strings), 27
- nlines, 30
- nlines, character-method (nlines), 30
- nlines, list-method (nlines), 30
- nlines, NULL-method (nlines), 30
- num\_rep\_cols, 30
- num\_rep\_cols, ANY-method (num\_rep\_cols),  
30
  
- obj\_align (lab\_name), 17
- obj\_align, ANY-method (lab\_name), 17
- obj\_align, fmt\_config-method (lab\_name),  
17
- obj\_align<- (lab\_name), 17
- obj\_align<- , ANY-method (lab\_name), 17
- obj\_align<- , fmt\_config-method  
(lab\_name), 17
- obj\_format (lab\_name), 17
- obj\_format, ANY-method (lab\_name), 17
- obj\_format, fmt\_config-method  
(lab\_name), 17
- obj\_format<- (lab\_name), 17
  
- obj\_format<- , ANY-method (lab\_name), 17
- obj\_format<- , fmt\_config-method  
(lab\_name), 17
- obj\_label (lab\_name), 17
- obj\_label, ANY-method (lab\_name), 17
- obj\_label<- (lab\_name), 17
- obj\_label<- , ANY-method (lab\_name), 17
- obj\_na\_str (lab\_name), 17
- obj\_na\_str, ANY-method (lab\_name), 17
- obj\_na\_str, fmt\_config-method  
(lab\_name), 17
- obj\_na\_str<- (lab\_name), 17
- obj\_na\_str<- , ANY-method (lab\_name), 17
- obj\_na\_str<- , fmt\_config-method  
(lab\_name), 17
- obj\_name (lab\_name), 17
- obj\_name<- (lab\_name), 17
  
- padstr, 31
- pag\_indices\_inner, 41
- pagdfrow, 32
- page\_dim (page\_types), 35
- page\_lcpp, 33
- page\_titles (main\_title), 19
- page\_titles, ANY-method (main\_title), 19
- page\_titles, MatrixPrintForm-method  
(main\_title), 19
- page\_titles<- (main\_title), 19
- page\_titles<- , MatrixPrintForm-method  
(main\_title), 19
- page\_types, 35
- paginate (paginate\_indices), 35
- paginate\_indices, 35
- paginate\_to\_mpf (paginate\_indices), 35
- pagination (paginate\_indices), 35
- pagination\_algo, 40
- print, ANY-method, 43
- propose\_column\_widths, 43
- prov\_footer (main\_title), 19
- prov\_footer, MatrixPrintForm-method  
(main\_title), 19
- prov\_footer<- (main\_title), 19
- prov\_footer<- , MatrixPrintForm-method  
(main\_title), 19
  
- ref\_df\_row, 44
- round, 45, 46
- round\_fmt, 45
- round\_fmt(), 15

rounding(round\_fmt), 45

spans\_to\_viscell, 46

spread\_integer, 47

sprintf, 45, 46, 48

sprintf\_format, 48

subtitles(main\_title), 19

subtitles,MatrixPrintForm-method  
    (main\_title), 19

subtitles<- (main\_title), 19

subtitles<-,MatrixPrintForm-method  
    (main\_title), 19

table\_inset, 24, 48

table\_inset,MatrixPrintForm-method  
    (table\_inset), 48

table\_inset<- (table\_inset), 48

table\_inset<-,MatrixPrintForm-method  
    (table\_inset), 48

toString, 6, 49

toString,MatrixPrintForm-method  
    (toString), 49

var\_labels, 50

var\_labels<-, 51

var\_labels\_remove, 52

var\_relabel, 52

vert\_pag\_indices, 53

with\_label, 54

wrap\_string, 54

wrap\_txt(wrap\_string), 54