Package 'formatters'

March 29, 2022

Title ASCII Formatting for Values and Tables

Date 2022-03-24

Version 0.2.0

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

Depends methods, R (>= 2.10)

Imports htmltools

Suggests dplyr, tinytest, knitr, rmarkdown

Encoding UTF-8

LazyData true

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

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

RoxygenNote 7.1.2

VignetteBuilder knitr

NeedsCompilation no

Author Gabriel Becker [aut, cre],
Adrian Waddell [aut],
F. Hoffmann-La Roche [cph]

Maintainer Gabriel Becker <gabembecker@gmail.com>

Repository CRAN

Date/Publication 2022-03-29 18:30:02 UTC

R topics documented:

`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)
```

Arguments

`df`	data.frame

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"
)
```

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,d efaults to 1 for all rows
`rclass`	character. Class(es) for the rows. Defaults to "NA"

Value

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

Examples

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

divider_height *Divider Height*

Description

Divider Height

Usage

```
divider_height(obj)

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

Arguments

obj	ANY. Object.

Value

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

Examples

```
divider_height(mtcars)
```

DM *DM data*

Description

DM data

Usage

```
DM
```

Format

rds (data.frame)

ex_adsl *Simulated CDISC Alike Data for Examples*

Description

Simulated CDISC Alike Data for Examples

Usage

```
ex_adsl

ex_adae

ex_adaette

ex_adtte

ex_adcm

ex_adlb

ex_admh

ex_adqs

ex_adrs

ex_advs
```

Format

rds (data.frame)

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

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

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

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

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

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

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

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

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

`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"`.

Value

formatted text representing the cell `x`.

See Also

[round_fmt()]

Examples

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

format_value(x, output = "ascii")
```

ifnotlen0 *'%||%' If length-0 alternative operator*

Description

'%||%' If length-0 alternative operator

Usage

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

Arguments

a	ANY. Element to select only if it is not length 0
b	ANY. Element to select if a is length 0

Value

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

Examples

```
6 %||% 10

character() %||% "hi"

NULL %||% "hi"
```

is.wholenumber *is.wholenumber*

Description

is.wholenumber

Usage

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

Arguments

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

Value

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

Examples

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

`is_valid_format` *Check if a format is supported*

Description

Check if a format is supported

Usage

```
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")
```

lab_name *Label, Name and Format accessor generics Getters and setters for basic, relatively universal attributes of "table-like" objects"*

Description

Label, Name and Format accessor generics

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)

obj_format(obj) <- value

## S4 replacement method for signature 'ANY'
obj_format(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

subtitles(obj)

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

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_
)
```

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()
	@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

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.

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 returne 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.

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

Description

This should generally only be called by ‘matrix_form‘ custom methods, and almost never from other code.

Usage

```
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()
)
```

```
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()
)
```

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 expact_newlines is set to FALSE.
ref_fnotes	list. Referential footnote information if applicable.
nlines_header	numeric(1). Number of lines taken up by the values of the header (ie not including the divider).
nrow_header	numeric(1). Number of *rows* corresponding to the header.
has_topleft	logical(1). Does the corresponding table have 'top left information' which should be treated differently when expanding newlines. Ignored if expand_newlines is FALSE.

`has_rowlabs` logical(1). Do the matrices (`strings`, `spans`, `aligns`) each contain a column that corresponds with row labels (Rather than with table cell values). Defaults to `TRUE`.

`expand_newlines` logical(1). Should the matrix form generated expand rows whose values contain newlines into multiple 'physical' rows (as they will appear when rendered into ASCII). Defaults to `TRUE`

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

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

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

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

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

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 arugment

`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

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

`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)

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

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)

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 funciton.

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.

nlines *Number of lines required to print a value*

Description

Number of lines required to print a value

Usage

```
nlines(x, colwidths)

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

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

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

Arguments

x	ANY. The object to be printed
colwidths	numeric. Column widths (if necessary)

Value

A scalar numeric indicating the number of lines needed to render the object x.

padstr *Pad a string and align within string*

Description

Pad a string and align within string

Usage

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

Arguments

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

Value

'x', padded to be a string of 'n' characters

Examples

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

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

---

pagdfrow *Create row of pagination data frame*

Description

Create row of pagination data frame

Usage

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

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 amongst sibling rows
nsibs	integer(1). Number of siblings
extent	numberic(1). Number of lines required to print the row
colwidths	numeric. Column widths
repext	integer(1). Number of lines required to reprint all context for this row if it appears directly after pagination.
repind	integer. Vector of row numbers to be reprinted if this row appears directly after pagination.
indent	integer. Indent
rclass	character(1). Class of row object.
nrowrefs	integer(1). Number of row referential footnotes for this row
ncellrefs	integer(1). Number of cell referential footnotes for the cells in this row
nreflines	integer(1). Total number of lines required by all referential footnotes
force_page	logical(1). Currently Ignored.
page_title	logical(1). Currently Ignored.

Value

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

---

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
)
```

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.

Value

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

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

Description

Print

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)
```

Arguments

x	MatrixPrintForm object

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)
```

---

round_fmt *Round and prepare a value for display*

Description

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

Usage

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

Arguments

x	numeric(1). Value to format
digits	numeric(1). Number of digits to round to, or NA to convert to a character value with no rounding.
na_str	character(1). The value to return if x is NA.

Details

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

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

Value

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

Note

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

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

See Also

link{format_value} round sprintf

Examples

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

---

spans_to_viscell *Transform vectors of spans (with dupblication) to Visibility vector*

Description

Transform vectors of spans (with dupblication) to Visibility vector

Usage

```
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 (ie two 2s, three 3s, etc).

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

Value

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

Note

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

Examples

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

---

spread_integer *spread x into len elements*

Description

spread x into len elements

Usage

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

Arguments

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

Value

if x is a scalar "whole number" value (see is.wholenumber), the value x repeated len times. If not, an error is thrown.

Examples

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

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

Description

Specify text format via a sprintf format string

Usage

```
sprintf_format(format)
```

Arguments

`format`	character(1). A format string passed to sprintf.

Value

A formating 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))
```

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

Description

toString

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, col_gap = 3, linesep = "")
```

Arguments

x	ANY. Object to be prepared for rendering.
...	Passed to individual methods.
widths	(proposed) widths for the columns of x
col_gap	numeric(1). Space between columns
linesep	character(1). Characters to repeat to create header/body separator line.

Value

A character string containing the ASCII rendering of the table-like object reprseented 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 *Remove Variable Labels of a* data.frame

Description

Removing labels attributes from a variables in a data frame

Usage

```
var_labels_remove(x)
```

Arguments

x	a data.frame object

Value

the same data frame as x stripped of variable labels

Examples

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

var_relabel *Copy and Change Variable Labels of a* data.frame

Description

Relabel a subset of the variables

Usage

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

Arguments

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

Value

a copy of x with changed labels according to ...

Examples

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

vert_pag_indices *Find Column Indicies for Vertical Pagination*

Description

Find Column Indicies for Vertical Pagination

Usage

```
vert_pag_indices(obj, cpp = 40, verbose = FALSE)
```

Arguments

obj	ANY. object to be paginated. Must have a matrix_form method.
cpp	numeric(1). Number of columns per page
verbose	logical(1). Should additional informative messages about the search for pagination breaks be shown. Defaults to FALSE.

Value

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

Examples

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

with_label *Return an object with a label attribute*

Description

Return an object with a label attribute

Usage

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

Arguments

x	an object
label	label attribute to to attached to x

Value

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

Examples

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

Index