Package 'rlistings'

June 15, 2025

Title Clinical Trial Style Data Readout Listings

Version 0.2.12

Date 2025-05-28

Description Listings are often part of the submission of clinical trial data in regulatory settings. We provide a framework for the specific formatting features often used when displaying large datasets in that context.

License Apache License 2.0

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

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

Depends formatters (>= 0.5.11), methods, tibble (>= 2.0.0)

Imports checkmate (>= 2.1.0), grDevices, grid, stats, utils

Suggests dplyr (>= 1.0.2), knitr (>= 1.42), lifecycle (>= 0.2.0), rmarkdown (>= 2.23), stringi (>= 1.6), testthat (>= 3.1.5), withr (>= 2.0.0)

VignetteBuilder knitr, rmarkdown

Config/Needs/verdepcheck insightsengineering/formatters, tidyverse/tibble, mllg/checkmate, tidyverse/dplyr, yihui/knitr, r-lib/lifecycle, rstudio/rmarkdown, gagolews/stringi, r-lib/testthat, r-lib/withr

Config/Needs/website insightsengineering/nesttemplate

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.3.2

NeedsCompilation no

Author Gabriel Becker [aut] (original creator of the package),
Adrian Waddell [aut],
Joe Zhu [aut, cre] (ORCID: <https://orcid.org/0000-0001-7566-2787>),
Davide Garolini [aut] (ORCID: <https://orcid.org/0000-0002-1445-1369>),
Emily de la Rua [aut] (ORCID: <https://orcid.org/0009-0000-8738-5561>),
Abinaya Yogasekaram [ctb] (ORCID:
<https://orcid.org/0009-0005-2083-1105>),
F. Hoffmann-La Roche AG [cph, fnd]

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

Repository CRAN

Date/Publication 2025-06-15 14:30:02 UTC

Contents

`as_listing` *Create a listing from a* `data.frame` *or* `tibble`

Description

[Experimental]

Create listings displaying `key_cols` and `disp_cols` to produce a compact and elegant representation of the input `data.frame` or `tibble`.

Usage

```
as_listing(
  df,
  key_cols = names(df)[1],
  disp_cols = NULL,
  non_disp_cols = NULL,
  sort_cols = key_cols,
  unique_rows = FALSE,
  default_formatting = list(all = fmt_config()),
  col_formatting = NULL,
  align_colnames = FALSE,
  add_trailing_sep = NULL,
  trailing_sep = " ",
```

```
  main_title = NULL,
  subtitles = NULL,
  main_footer = NULL,
  prov_footer = NULL,
  split_into_pages_by_var = NULL,
  spanning_col_labels = no_spans_df
)

spanning_col_label_df(df)

spanning_col_label_df(df) <- value

as_keycol(vec)

is_keycol(vec)

get_keycols(df)

listing_dispcols(df)

add_listing_dispcol(df, new)

listing_dispcols(df) <- value

align_colnames(df)

align_colnames(df) <- value

add_listing_col(
  df,
  name,
  fun = NULL,
  format = NULL,
  na_str = "NA",
  align = "left"
)
```

Arguments

df	(data.frame or listing_df) the data.frame to be converted to a listing or listing_df to be modified.
key_cols	(character) vector of names of columns which should be treated as *key columns* when rendering the listing. Key columns allow you to group repeat occurrences.
disp_cols	(character or NULL) vector of names of non-key columns which should be displayed when the listing is rendered. Defaults to all columns of df not named in key_cols or non_disp_cols.
non_disp_cols	(character or NULL)

vector of names of non-key columns to be excluded as display columns. All other non-key columns are treated as display columns. Ignored if `disp_cols` is non-`NULL`.

`sort_cols` (`character` or `NULL`)
vector of names of columns (in order) which should be used to sort the listing. Defaults to `key_cols`. If `NULL`, no sorting will be performed.

`unique_rows` (`flag`)
whether only unique rows should be included in the listing. Defaults to `FALSE`.

`default_formatting`
(`list`)
a named list of default column format configurations to apply when rendering the listing. Each name-value pair consists of a name corresponding to a data class (or "numeric" for all unspecified numeric classes) and a value of type `fmt_config` with the format configuration that should be implemented for columns of that class. If named element "all" is included in the list, this configuration will be used for all data classes not specified. Objects of type `fmt_config` can take 3 arguments: `format`, `na_str`, and `align`.

`col_formatting` (`list`)
a named list of custom column formatting configurations to apply to specific columns when rendering the listing. Each name-value pair consists of a name corresponding to a column name and a value of type `fmt_config` with the formatting configuration that should be implemented for that column. Objects of type `fmt_config` can take 3 arguments: `format`, `na_str`, and `align`. Defaults to `NULL`.

`align_colnames` (`flag`)
whether the column titles should have the same alignment as their columns. All titles default to `"center"` alignment if `FALSE` (default). This can be changed with `align_colnames()`.

`add_trailing_sep`
(`character` or `numeric` or `NULL`)
If it is assigned to one or more column names, a trailing separator will be added between groups with identical values for that column. Numeric option allows the user to specify in which rows it can be added. Defaults to `NULL`.

`trailing_sep` (`character(1)`)
The separator to be added between groups. The character will be repeated to fill the row.

`main_title` (`string` or `NULL`)
the main title for the listing, or `NULL` (the default).

`subtitles` (`character` or `NULL`)
a vector of subtitles for the listing, or `NULL` (the default).

`main_footer` (`character` or `NULL`)
a vector of main footer lines for the listing, or `NULL` (the default).

`prov_footer` (`character` or `NULL`)
a vector of provenance footer lines for the listing, or `NULL` (the default). Each string element is placed on a new line.

split_into_pages_by_var
: (character or NULL)
the name of a variable for on the listing should be split into pages, with each page corresponding to one unique value/level of the variable. See split_into_pages_by_var() for more details.

spanning_col_labels
: (data.frame)
A data.frame with the columns span_level, label, start, and span defining 0 or more levels of addition spanning (ie grouping) of columns. Defaults to no additional spanning labels.

value
: (string)
new value.

vec
: (string)
name of a column vector from a listing_df object to be annotated as a key column.

new
: (character)
vector of names of columns to be added to the set of display columns.

name
: (string)
name of the existing or new column to be displayed when the listing is rendered.

fun
: (function or NULL)
a function which accepts df and returns the vector for a new column, which is added to df as name, or NULL if marking an existing column as a listing column.

format
: (string or function)
a format label (string) or formatter function.

na_str
: (string)
string that should be displayed in place of missing values.

align
: (string)
alignment values should be rendered with.

Details

At its core, a listing_df object is a tbl_df object with a customized print method and support for the formatting and pagination machinery provided by the formatters package.

listing_df objects have two 'special' types of columns: key columns and display columns.

Key columns act as indexes, which means a number of things in practice.

All key columns are also display columns.

listing_df objects are always sorted by their set of key columns at creation time. Any listing_df object which is not sorted by its full set of key columns (e.g., one whose rows have been reordered explicitly during creation) is invalid and the behavior when rendering or paginating that object is undefined.

Each value of a key column is printed only once per page and per unique combination of values for all higher-priority (i.e., to the left of it) key columns. Locations where a repeated value would have been printed within a key column for the same higher-priority-key combination on the same page are rendered as empty space. Note, determination of which elements to display within a key column

at rendering is based on the underlying value; any non-default formatting applied to the column has no effect on this behavior.

Display columns are columns which should be rendered, but are not key columns. By default this is all non-key columns in the incoming data, but in need not be. Columns in the underlying data which are neither key nor display columns remain within the object available for computations but *are not rendered during printing or export of the listing*.

Spanning column labels are displayed centered above the individual labels of the columns they span across. `span_level` 1 is placed directly above the column labels, with higher "span_levels‘ displayed above it in ascending order.

IF spanning column labels are present, a single spanning label cannot span across both key and non-key displayed columns simultaneously due to key columns' repetition after page breaks during horizontal pagination. Attempting to set a spanning column label which does so will result in an error.

Value

A `listing_df` object, sorted by its key columns.

`df` with `name` created (if necessary) and marked for display during rendering.

Note

Unlike in the `rtables` sister package, spanning labels here are purely decorative and do not reflect any structure among the columns modeled by `rlistings`. Thus, we cannot, e.g., use pathing to select columns under a certain spanning column label, or restrict horizontal pagination to leave 'groups' of columns implied by a spanning label intact.

Examples

```
dat <- ex_adae

# This example demonstrates the listing with key_cols (values are grouped by USUBJID) and
# multiple lines in prov_footer
lsting <- as_listing(dat[1:25, ],
  key_cols = c("USUBJID", "AESOC"),
  main_title = "Example Title for Listing",
  subtitles = "This is the subtitle for this Adverse Events Table",
  main_footer = "Main footer for the listing",
  prov_footer = c(
    "You can even add a subfooter", "Second element is place on a new line",
    "Third string"
  )
) %>%
  add_listing_col("AETOXGR") %>%
  add_listing_col("BMRKR1", format = "xx.x") %>%
 add_listing_col("AESER / AREL", fun = function(df) paste(df$AESER, df$AREL, sep = " / "))

mat <- matrix_form(lsting)

cat(toString(mat))
```

```
# This example demonstrates the listing table without key_cols
# and specifying the cols with disp_cols.
dat <- ex_adae
lsting <- as_listing(dat[1:25, ],
  disp_cols = c("USUBJID", "AESOC", "RACE", "AETOXGR", "BMRKR1")
)

mat <- matrix_form(lsting)

cat(toString(mat))

# This example demonstrates a listing with format configurations specified
# via the default_formatting and col_formatting arguments
dat <- ex_adae
dat$AENDY[3:6] <- NA
lsting <- as_listing(dat[1:25, ],
  key_cols = c("USUBJID", "AESOC"),
  disp_cols = c("STUDYID", "SEX", "ASEQ", "RANDDT", "ASTDY", "AENDY"),
  default_formatting = list(
    all = fmt_config(align = "left"),
    numeric = fmt_config(
      format = "xx.xx",
      na_str = "<No data>",
      align = "right"
    )
  )
) %>%
  add_listing_col("BMRKR1", format = "xx.x", align = "center")

mat <- matrix_form(lsting)

cat(toString(mat))
```

`listing_methods` *Methods for* `listing_df` *objects*

Description

See core documentation in formatters::formatters-package for descriptions of these functions.

Usage

```
## S3 method for class 'listing_df'
print(
  x,
  widths = NULL,
  tf_wrap = FALSE,
  max_width = NULL,
  fontspec = NULL,
```

```
  col_gap = 3L,
  round_type = c("iec", "sas"),
  ...
)

## S4 method for signature 'listing_df'
toString(
  x,
  widths = NULL,
  fontspec = NULL,
  col_gap = 3L,
  round_type = c("iec", "sas"),
  ...
)

## S4 method for signature 'listing_df'
x[i, j, drop = FALSE]

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

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

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

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

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

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

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

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

## S4 method for signature 'listing_df'
num_rep_cols(obj)
```

Arguments

x	(listing_df) the listing.

widths	(numeric or NULL) Proposed widths for the columns of x. The expected length of this numeric vector can be retrieved with ncol(x) + 1 as the column of row names must also be considered.
tf_wrap	(flag) whether the text for title, subtitles, and footnotes should be wrapped.
max_width	(integer(1), string or NULL) width that title and footer (including footnotes) materials should be word-wrapped to. If NULL, it is set to the current print width of the session (getOption("width")). If set to "auto", the width of the table (plus any table inset) is used. Parameter is ignored if tf_wrap = FALSE.
fontspec	(font_spec) a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().
col_gap	(numeric(1)) space (in characters) between columns.
round_type	("iec" or "sas") the type of rounding to perform. iec, the default, peforms rounding compliant with IEC 60559 (see details), while sas performs nearest-value rounding consistent with rounding within SAS.
...	additional parameters passed to formatters::toString().
i	(any) object passed to base [methods.
j	(any) object passed to base [methods.
drop	relevant for matrices and arrays. If TRUE the result is coerced to the lowest possible dimension (see the examples). This only works for extracting elements, not for the replacement. See drop for further details.
obj	(listing_df) the listing.
value	typically an array-like R object of a similar class as x.

Value

- Accessor methods return the value of the aspect of obj.
- Setter methods return obj with the relevant element of the listing updated.

Examples

```
lsting <- as_listing(mtcars)
main_title(lsting) <- "Hi there"

main_title(lsting)
```

`make_row_df,listing_df-method`
Make pagination data frame for a listing

Description

Make pagination data frame for a listing

Usage

```
## S4 method for signature 'listing_df'
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_,
  fontspec = dflt_courier,
  round_type = c("iec", "sas")
)
```

Arguments

`tt`	(`listing_df`) the listing to be rendered.
`colwidths`	(`numeric`) internal detail, do not set manually.
`visible_only`	(`flag`) ignored, as listings do not have non-visible structural elements.
`rownum`	(`numeric(1)`) internal detail, do not set manually.
`indent`	(`integer(1)`) internal detail, do not set manually.
`path`	(`character`) path to the (sub)table represented by `tt`. Defaults to `character()`.
`incontent`	(`flag`) internal detail, do not set manually.
`repr_ext`	(`integer(1)`) internal detail, do not set manually.

repr_inds (integer)
internal detail, do not set manually.

sibpos (integer(1))
internal detail, do not set manually.

nsibs (integer(1))
internal detail, do not set manually.

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

round_type ("iec" or "sas")
the type of rounding to perform. iec, the default, peforms rounding compliant with IEC 60559 (see details), while sas performs nearest-value rounding consistent with rounding within SAS.

Value

a data.frame with pagination information.

See Also

formatters::make_row_df()

Examples

```
lsting <- as_listing(mtcars)
mf <- matrix_form(lsting)
```

---

matrix_form,listing_df-method

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

```
## S4 method for signature 'listing_df'
matrix_form(
  obj,
  indent_rownames = FALSE,
  expand_newlines = TRUE,
```

```
  fontspec = font_spec,
  col_gap = 3L,
  round_type = c("iec", "sas")
)
```

Arguments

obj	(ANY) object to be transformed into a ready-to-render form (a MatrixPrintForm object).
indent_rownames	(flag) silently ignored, as listings do not have row names nor indenting structure.
expand_newlines	(flag) this should always be TRUE for listings. We keep it for debugging reasons.
fontspec	(font_spec) a font_spec object specifying the font information to use for calculating string widths and heights, as returned by font_spec().
col_gap	(numeric(1)) the gap to be assumed between columns, in number of spaces with font specified by fontspec.
round_type	("iec" or "sas") the type of rounding to perform. iec, the default, peforms rounding compliant with IEC 60559 (see details), while sas performs nearest-value rounding consistent with rounding within SAS.

Value

a formatters::MatrixPrintForm object.

See Also

formatters::matrix_form()

Examples

```
lsting <- as_listing(mtcars)
mf <- matrix_form(lsting)
```

paginate_listing *Paginate listings*

Description

[Experimental]

Pagination of a listing. This can be vertical for long listings with many rows and/or horizontal if there are many columns. This function is a wrapper of `formatters::paginate_to_mpfs()` and it is mainly meant for exploration and testing.

Usage

```
paginate_listing(
  lsting,
  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_,
  colwidths = NULL,
  tf_wrap = !is.null(max_width),
  rep_cols = NULL,
  max_width = NULL,
  col_gap = 3,
  fontspec = font_spec(font_family, font_size, lineheight),
  verbose = FALSE,
  print_pages = TRUE
)
```

Arguments

`lsting`	(`listing_df` or `list`) the listing or list of listings to paginate.
`page_type`	(`string`) name of a page type. See `page_types`. Ignored when `pg_width` and `pg_height` are set directly.
`font_family`	(`string`) name of a font family. An error will be thrown if the family named is not monospaced. Defaults to `"Courier"`.
`font_size`	(`numeric(1)`) font size. Defaults to `12`.

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

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

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

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

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

lpp (numeric(1) or NULL)
number of rows/lines (excluding titles and footers) to include per page. Standard is 70 while NULL disables vertical pagination.

cpp (numeric(1) or NULL)
width (in characters) of the pages for horizontal pagination. NULL (the default) indicates no horizontal pagination should be done.

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

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

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

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

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

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

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

print_pages (flag)
whether the paginated listing should be printed to the console (cat(toString(x))).

Value

A list of listing_df objects where each list element corresponds to a separate page.

Examples

```
dat <- ex_adae
lsting <- as_listing(dat[1:25, ], disp_cols = c("USUBJID", "AESOC", "RACE", "AETOXGR", "BMRKR1"))
mat <- matrix_form(lsting)
cat(toString(mat))

paginate_listing(lsting, lpp = 10)

paginate_listing(lsting, cpp = 100, lpp = 40)

paginate_listing(lsting, cpp = 80, lpp = 40, verbose = TRUE)
```

---

split_into_pages_by_var

Split Listing by Values of a Variable

Description

[Experimental]

Split is performed based on unique values of the given parameter present in the listing. Each listing can only be split by variable once. If this function is applied prior to pagination, parameter values will be separated by page.

Usage

```
split_into_pages_by_var(lsting, var, page_prefix = var)
```

Arguments

`lsting`	(`listing_df`) the listing to split.
`var`	(`string`) name of the variable to split on. If the column is a factor, the resulting list follows the order of the levels.
`page_prefix`	(`string`) prefix to be appended with the split value (`var` level), at the end of the subtitles, corresponding to each resulting list element (listing).

Value

A list of `lsting_df` objects each corresponding to a unique value of `var`.

Note

This function should only be used after the complete listing has been created. The listing cannot be modified further after applying this function.

Examples

```
dat <- ex_adae[1:20, ]

lsting <- as_listing(
  dat,
  key_cols = c("USUBJID", "AGE"),
  disp_cols = "SEX",
  main_title = "title",
  main_footer = "footer"
) %>%
  add_listing_col("BMRKR1", format = "xx.x") %>%
  split_into_pages_by_var("SEX")

lsting
```

Index