Package 'skimr'

December 23, 2022

Title Compact and Flexible Summaries of Data

Version 2.1.5

Description A simple to use summary function that can be used with pipes and displays nicely in the console. The default summary statistics may be modified by the user as can the default formatting. Support for data frames and vectors is included, and users can implement their own skim methods for specific object types as described in a vignette. Default summaries include support for inline spark graphs. Instructions for managing these on specific operating systems are given in the ``Using skimr'' vignette and the README.

License GPL-3

URL https://docs.ropensci.org/skimr/ (website), https://github.com/ropensci/skimr/

BugReports https://github.com/ropensci/skimr/issues

Depends R (>= 3.1.2)

Imports cli, dplyr (>= 0.8.0), knitr (>= 1.2), magrittr (>= 1.5), pillar (>= 1.6.4), purrr, repr, rlang (>= 0.3.1), stats, stringr (>= 1.1), tibble (>= 2.0.0), tidyr (>= 1.0), tidyselect (>= 1.0.0), vctrs

Suggests covr, crayon, data.table, dtplyr, extrafont, haven, lubridate, rmarkdown, testthat (>= 2.0.0), withr

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.2.3

Collate 'deprecated.R' 'dplyr.R' 'stats.R' 'skim_with.R' 'get_skimmers.R' 'reshape.R' 'sfl.R' 'skim.R' 'skim_obj.R' 'skim_print.R' 'skimr-package.R' 'summary.R' 'utils.R' 'vctrs.R'

NeedsCompilation no

Author Elin Waring [cre, aut],
Michael Quinn [aut],
Amelia McNamara [aut],
Eduardo Arino de la Rubia [aut],
Hao Zhu [aut],
Julia Lowndes [ctb],
Shannon Ellis [aut],
Hope McLeod [ctb],
Hadley Wickham [ctb],
Kirill Müller [ctb],
RStudio, Inc. [cph] (Spark functions),
Connor Kirkpatrick [ctb],
Scott Brenstuhl [ctb],
Patrick Schratz [ctb],
lbusett [ctb],
Mikko Korpela [ctb],
Jennifer Thompson [ctb],
Harris McGehee [ctb],
Mark Roepke [ctb],
Patrick Kennedy [ctb],
Daniel Possenriede [ctb],
David Zimmermann [ctb],
Kyle Butts [ctb],
Bastian Torges [ctb],
Rick Saporta [ctb],
Henry Morgan Stewart [ctb]

Maintainer Elin Waring <elin.waring@gmail.com>

Repository CRAN

Date/Publication 2022-12-23 11:10:02 UTC

R topics documented:

`skimr-package` *Skim a data frame*

Description

This package provides an alternative to the default summary functions within R. The package's API is tidy, functions take data frames, return data frames and can work as part of a pipeline. The returned `skimr` object is subsettable and offers a human readable output.

Details

`skimr` is opinionated, providing a strong set of summary statistics that are generated for a variety of different data types. It is also provides an API for customization. Users can change both the functions dispatched and the way the results are formatted.

`deprecated-v1` *Deprecated functions from skimr v1*

Description

Skimr used to offer functions that combined skimming with a secondary effect, like reshaping the data, building a list or printing the results. Some of these behaviors are no longer necessary. `skim()` always returns a wide data frame. Others have been replaced by functions that do a single thing. `partition()` creates a list-like object from a skimmed data frame.

Usage

```
skim_to_wide(.data, ...)

skim_to_list(.data, ...)

skim_format(...)
```

Arguments

`.data`	A tibble, or an object that can be coerced into a tibble.
`...`	Columns to select for skimming. When none are provided, the default is to skim all columns.

Value

Either A `skim_df` or a `skim_list` object.

Functions

- `skim_to_wide()`: `skim()` always produces a wide data frame.
- `skim_to_list()`: `partition()` creates a list.
- `skim_format()`: `print()` and `skim_with()` set options.

`fix_windows_histograms` *Fix unicode histograms on Windows*

Description

This functions changes your session's locale to address issues with printing histograms on Windows.

Usage

```
fix_windows_histograms()
```

Details

There are known issues with printing the spark-histogram characters when printing a data frame, appearing like this: "<U+2582><U+2585><U+2587>". This longstanding problem originates in the low-level code for printing dataframes.

See Also

`skim_without_charts()`

`focus` *Only show a subset of summary statistics after skimming*

Description

This function is a variant of `dplyr::select()` designed to work with `skim_df` objects. When using `focus()`, `skimr` metadata columns are kept, and `skimr` print methods are still utilized. Otherwise, the signature and behavior is identical to `dplyr::select()`.

Usage

```
focus(.data, ...)
```

Arguments

.data	A skim_df object.
...	One or more unquoted expressions separated by commas. Variable names can be used as if they were positions in the data frame, so expressions like x:y can be used to select a range of variables.

Examples

```
# Compare
iris %>%
  skim() %>%
  dplyr::select(n_missing)

iris %>%
  skim() %>%
  focus(n_missing)

# This is equivalent to
iris %>%
  skim() %>%
  dplyr::select(skim_variable, skim_type, n_missing)
```

---

get_default_skimmers *View default skimmer names and functions*

Description

These utility functions look up the currently-available defaults for one or more skim_type's. They work with all defaults in the skimr package, as well as the defaults in any package that extends skimr. See get_skimmers() for writing lookup methods for different.

Usage

```
get_default_skimmers(skim_type = NULL)

get_one_default_skimmer(skim_type)

get_default_skimmer_names(skim_type = NULL)

get_one_default_skimmer_names(skim_type)

get_sfl(skim_type)
```

Arguments

skim_type	The class of the column being skimmed.

Details

The functions differ in return type and whether or not the result is in a list. `get_default_skimmers()` and `get_one_default_skimmer()` return functions. The former returns functions within a typed list, i.e. `list(numeric = list(...functions...))`.

The functions differ in return type and whether or not the result is in a list. `get_default_skimmer_names()` and `get_one_default_skimmer_names()` return a list of character vectors or a single character vector.

`get_sfl()` returns the skimmer function list for a particular `skim_type`. It differs from `get_default_skimmers()` in that the returned `sfl` contains a list of functions and a `skim_type`.

Functions

- `get_one_default_skimmer()`: Get the functions associated with one `skim_type`.
- `get_default_skimmer_names()`: Get the names of the functions used in one or more `skim_type`'s.
- `get_one_default_skimmer_names()`: Get the names of the functions used in one `skim_type`.
- `get_sfl()`: Get the `sfl` for a `skim_type`.

`get_skimmers` *Retrieve the summary functions for a specific data type*

Description

These functions are used to set the default skimming functions for a data type. They are combined with the base skim function list (`sfl`) in `skim_with()`, to create the summary tibble for each type.

Usage

```
get_skimmers(column)

## Default S3 method:
get_skimmers(column)

## S3 method for class 'numeric'
get_skimmers(column)

## S3 method for class 'factor'
get_skimmers(column)

## S3 method for class 'character'
get_skimmers(column)

## S3 method for class 'logical'
get_skimmers(column)

## S3 method for class 'complex'
```

```
get_skimmers(column)

## S3 method for class 'Date'
get_skimmers(column)

## S3 method for class 'POSIXct'
get_skimmers(column)

## S3 method for class 'difftime'
get_skimmers(column)

## S3 method for class 'Timespan'
get_skimmers(column)

## S3 method for class 'ts'
get_skimmers(column)

## S3 method for class 'list'
get_skimmers(column)

## S3 method for class 'AsIs'
get_skimmers(column)

## S3 method for class 'haven_labelled'
get_skimmers(column)

modify_default_skimmers(skim_type, new_skim_type = NULL, new_funs = list())
```

Arguments

column	An atomic vector or list. A column from a data frame.
skim_type	A character scalar. The class of the object with default skimmers.
new_skim_type	The type to assign to the looked up set of skimmers.
new_funs	Replacement functions for those in

Details

When creating your own set of skimming functions, call `sfl()` within a `get_skimmers()` method for your particular type. Your call to `sfl()` should also provide a matching class in the `skim_type` argument. Otherwise, it will not be possible to dynamically reassign your default functions when working interactively.

Call `get_default_skimmers()` to see the functions for each type of summary function currently supported. Call `get_default_skimmer_names()` to just see the names of these functions. Use `modify_default_skimmers()` for a method for changing the `skim_type` or functions for a default `sfl`. This is useful for creating new default `sfl`'s.

Value

A skim_function_list object.

Methods (by class)

- get_skimmers(default): The default method for skimming data. Only used when a column's data type doesn't match currently installed types. Call get_default_skimmer_names to see these defaults.
- get_skimmers(numeric): Summary functions for numeric columns, covering both double() and integer() classes: mean(), sd(), quantile() and inline_hist().
- get_skimmers(factor): Summary functions for factor columns: is.ordered(), n_unique() and top_counts().
- get_skimmers(character): Summary functions for character columns. Also, the default for unknown columns: min_char(), max_char(), n_empty(), n_unique() and n_whitespace().
- get_skimmers(logical): Summary functions for logical/ boolean columns: mean(), which produces rates for each value, and top_counts().
- get_skimmers(complex): Summary functions for complex columns: mean().
- get_skimmers(Date): Summary functions for Date columns: min(), max(), median() and n_unique().
- get_skimmers(POSIXct): Summary functions for POSIXct columns: min(), max(), median() and n_unique().
- get_skimmers(difftime): Summary functions for difftime columns: min(), max(), median() and n_unique().
- get_skimmers(Timespan): Summary functions for Timespan columns: min(), max(), median() and n_unique().
- get_skimmers(ts): Summary functions for ts columns: min(), max(), median() and n_unique().
- get_skimmers(list): Summary functions for list columns: n_unique(), list_min_length() and list_max_length().
- get_skimmers(AsIs): Summary functions for AsIs columns: n_unique(), list_min_length() and list_max_length().
- get_skimmers(haven_labelled): Summary functions for haven_labelled columns. Finds the appropriate skimmers for the underlying data in the vector.

See Also

sfl()

Examples

```
# Defining default skimming functions for a new class, `my_class`.
# Note that the class argument is required for dynamic reassignment.
get_skimmers.my_class <- function(column) {
  sfl(
    skim_type = "my_class",
    mean,
```

```
    sd
  )
}

# Integer and double columns are both "numeric" and are treated the same
# by default. To switch this behavior in another package, add a method.
get_skimmers.integer <- function(column) {
  sfl(
    skim_type = "integer",
    p50 = ~ stats::quantile(
      .,
      probs = .50, na.rm = TRUE, names = FALSE, type = 1
    )
  )
}
x <- mtcars[c("gear", "carb")]
class(x$carb) <- "integer"
skim(x)
## Not run:
# In a package, to revert to the V1 behavior of skimming separately with the
# same functions, assign the numeric `get_skimmers`.
get_skimmers.integer <- skimr::get_skimmers.numeric

# Or, in a local session, use `skim_with` to create a different `skim`.
new_skim <- skim_with(integer = skimr::get_skimmers.numeric())

# To apply a set of skimmers from an old type to a new type
get_skimmers.new_type <- function(column) {
  modify_default_skimmers("old_type", new_skim_type = "new_type")
}

## End(Not run)
```

`knit_print` *Provide a default printing method for knitr.*

Description

Instead of standard R output, `knitr` and `RMarkdown` documents will have formatted `knitr::kable()` output on return. You can disable this by setting the chunk option `render = normal_print`.

Usage

```
## S3 method for class 'skim_df'
knit_print(x, options = NULL, ...)

## S3 method for class 'skim_list'
knit_print(x, options = NULL, ...)

## S3 method for class 'one_skim_df'
```

```
knit_print(x, options = NULL, ...)

## S3 method for class 'summary_skim_df'
knit_print(x, options = NULL, ...)
```

Arguments

x	An R object to be printed
options	Options passed into the print function.
...	Additional arguments passed to the S3 method. Currently ignored, except two optional arguments options and inline; see the references below.

Details

The summary statistics for the original data frame can be disabled by setting the knitr chunk option skimr_include_summary = FALSE. See knitr::opts_chunk for more information. You can change the number of digits shown in the printed table with the skimr_digits chunk option.

Alternatively, you can call collapse() or yank() to get the particular skim_df objects and format them however you like. One warning though. Because histograms contain unicode characters, they can have unexpected print results, as R as varying levels of unicode support. This affects Windows users most commonly. Call vignette("Using_fonts") for more details.

Value

A knit_asis object. Which is used by knitr when rendered.

Methods (by class)

- knit_print(skim_df): Default knitr print for skim_df objects.
- knit_print(skim_list): Default knitr print for a skim_list.
- knit_print(one_skim_df): Default knitr print within a partitioned skim_df.
- knit_print(summary_skim_df): Default knitr print for skim_df summaries.

See Also

knitr::kable()

---

mutate.skim_df *Mutate a skim_df*

Description

dplyr::mutate() currently drops attributes, but we need to keep them around for other skim behaviors. Otherwise the behavior is exactly the same. For more information, see https://github.com/tidyverse/dplyr/issues/3429.

Usage

```
## S3 method for class 'skim_df'
mutate(.data, ...)
```

Arguments

`.data`	A `skim_df`, which behaves like a `tbl`.
`...`	Name-value pairs of expressions, each with length 1 or the same length as the number of rows in the group, if using `dplyr::group_by()`, or in the entire input (if not using groups). The name of each argument will be the name of a new variable, and the value will be its corresponding value. Use `NULL` value in `dplyr::mutate()` to drop a variable. New variables overwrite existing variables of the same name. The arguments in `...` are automatically quoted with `rlang::quo()` and evaluated with `rlang::eval_tidy()` in the context of the data frame. They support unquoting rlang::quasiquotation and splicing. See `vignette("programming", package = "dplyr")` for an introduction to these concepts.

Value

A `skim_df` object, which also inherits the class(es) of the input data. In many ways, the object behaves like a `tibble::tibble()`.

See Also

`dplyr::mutate()` for the function's expected behavior.

---

partition *Separate a big* `skim_df` *into smaller data frames, by type.*

Description

The data frames produced by `skim()` are wide and sparse, filled with columns that are mostly `NA`. For that reason, it can be convenient to work with "by type" subsets of the original data frame. These smaller subsets have their `NA` columns removed.

Usage

```
partition(data)

bind(data)

yank(data, skim_type)
```

Arguments

`data`	A `skim_df`.
`skim_type`	A character scalar. The subtable to extract from a `skim_df`.

Details

partition() creates a list of smaller skim_df data frames. Each entry in the list is a data type from the original skim_df. The inverse of partition() is bind(), which takes the list and produces the original skim_df. While partition() keeps all of the subtables as list entries, yank() gives you a single subtable for a data type.

Value

A skim_list of skim_df's, by type.

Functions

- bind(): The inverse of a partition(). Rebuild the original skim_df.
- yank(): Extract a subtable from a skim_df with a particular type.

Examples

```
# Create a wide skimmed data frame (a skim_df)
skimmed <- skim(iris)

# Separate into a list of subtables by type
separate <- partition(skimmed)

# Put back together
identical(bind(separate), skimmed)
# > TRUE

# Alternatively, get the subtable of a particular type
yank(skimmed, "factor")
```

---

print *Print* skim *objects*

Description

skimr has custom print methods for all supported objects. Default printing methods for knitr/rmarkdown documents is also provided.

Usage

```
## S3 method for class 'skim_df'
print(
  x,
  include_summary = TRUE,
  n = Inf,
  width = Inf,
  summary_rule_width = getOption("skimr_summary_rule_width", default = 40),
  ...
```

```
)

## S3 method for class 'skim_list'
print(x, n = Inf, width = Inf, ...)

## S3 method for class 'summary_skim_df'
print(x, .summary_rule_width = 40, ...)
```

Arguments

x	Object to format or print.
include_summary	Whether a summary of the data frame should be printed
n	Number of rows to show. If NULL, the default, will print all rows if less than the print_max option. Otherwise, will print as many rows as specified by the print_min option.
width	Width of text output to generate. This defaults to NULL, which means use the width option.
summary_rule_width	Width of Data Summary cli rule, defaults to 40.
...	Passed on to tbl_format_setup().
.summary_rule_width	the width for the main rule above the summary.

Methods (by class)

- print(skim_df): Print a skimmed data frame (skim_df from skim()).
- print(skim_list): Print a skim_list, a list of skim_df objects.
- print(summary_skim_df): Print method for a summary_skim_df object.

Printing options

For better or for worse, skimr often produces more output than can fit in the standard R console. Fortunately, most modern environments like RStudio and Jupyter support more than 80 character outputs. Call options(width = 90) to get a better experience with skimr.

The print methods in skimr wrap those in the tibble package. You can control printing behavior using the same global options.

Behavior in dplyr pipelines

Printing a skim_df requires specific columns that might be dropped when using dplyr::select() or dplyr::summarize() on a skim_df. In those cases, this method falls back to tibble::print.tbl().

Options for controlling print behavior

You can control the width rule line for the printed subtables with an option: skimr_table_header_width.

See Also

`tibble::trunc_mat()` For a list of global options for customizing print formatting. `crayon::has_color()` for the variety of issues that affect tibble's color support.

`repr` *Skimr printing within Jupyter notebooks*

Description

This reproduces printed results in the console. By default Jupyter kernels render the final object in the cell. We want the version printed by `skimr` instead of the data that it contains.

Usage

```
## S3 method for class 'skim_df'
repr_text(obj, ...)

## S3 method for class 'skim_list'
repr_text(obj, ...)

## S3 method for class 'one_skim_df'
repr_text(obj, ...)
```

Arguments

`obj`	The object to print and then return the output.
`...`	ignored.

Value

None. `invisible(NULL)`.

`sfl` *Create a skimr function list*

Description

This constructor is used to create a named list of functions. It also you also pass `NULL` to identify a skimming function that you wish to remove. Only functions that return a single value, working with `dplyr::summarize()`, can be used within `sfl`.

Usage

```
sfl(..., skim_type = "")
```

Arguments

`...`	Inherited from dplyr::data_masking() for dplyr version 1 or later or dplyr::funs() for older versions of dplyr. A list of functions specified by: • Their name, `"mean"` • The function itself, `mean` • A call to the function with `.` as a dummy argument, `mean(., na.rm = TRUE)` • An anonymous function in **purrr** notation, `~mean(., na.rm = TRUE)`
`skim_type`	A character scalar. This is used to match locally-provided skimmers with defaults. See `get_skimmers()` for more detail.

Details

`sfl()` will automatically generate callables and names for a variety of inputs, including functions, formulas and strings. Nonetheless, we recommend providing names when reasonable to get better `skim()` output.

Value

A `skimr_function_list`, which contains a list of `fun_calls`, returned by `dplyr::funs()` and a list of skimming functions to drop.

See Also

`dplyr::funs()`, `skim_with()` and `get_skimmers()`.

Examples

```
# sfl's can take a variety of input formats and will generate names
# if not provided.
sfl(mad, "var", ~ length(.)^2)

# But these can generate unpredictable names in your output.
# Better to set your own names.
sfl(mad = mad, variance = "var", length_sq = ~ length(.)^2)

# sfl's can remove individual skimmers from defaults by passing NULL.
sfl(hist = NULL)

# When working interactively, you don't need to set a type.
# But you should when defining new defaults with `get_skimmers()`.
get_skimmers.my_new_class <- function(column) {
  sfl(n_missing, skim_type = "my_new_class")
}
```

skim *Skim a data frame, getting useful summary statistics*

Description

skim() is an alternative to summary(), quickly providing a broad overview of a data frame. It handles data of all types, dispatching a different set of summary functions based on the types of columns in the data frame.

Usage

```
skim(data, ..., .data_name = NULL)

skim_tee(data, ..., skim_fun = skim)

skim_without_charts(data, ..., .data_name = NULL)
```

Arguments

data	A tibble, or an object that can be coerced into a tibble.
...	Columns to select for skimming. When none are provided, the default is to skim all columns.
.data_name	The name to use for the data. Defaults to the same as data.
skim_fun	The skim function used.
skim	The skimming function to use in skim_tee().

Details

Each call produces a skim_df, which is a fundamentally a tibble with a special print method. One unusual feature of this data frame is pseudo- namespace for columns. skim() computes statistics by data type, and it stores them in the data frame as <type>.<statistic>. These types are stripped when printing the results. The "base" skimmers (n_missing and complete_rate) are the only columns that don't follow this behavior. See skim_with() for more details on customizing skim() and get_default_skimmers() for a list of default functions.

If you just want to see the printed output, call skim_tee() instead. This function returns the original data. skim_tee() uses the default skim(), but you can replace it with the skim argument.

The data frame produced by skim is wide and sparse. To avoid type coercion skimr uses a type namespace for all summary statistics. Columns for numeric summary statistics all begin numeric; for factor summary statistics begin factor; and so on.

See partition() and yank() for methods for transforming this wide data frame. The first function splits it into a list, with each entry corresponding to a data type. The latter pulls a single subtable for a particular type from the skim_df.

skim() is designed to operate in pipes and to generally play nicely with other tidyverse functions. This means that you can use tidyselect helpers within skim to select or drop specific columns for summary. You can also further work with a skim_df using dplyr functions in a pipeline.

Value

A skim_df object, which also inherits the class(es) of the input data. In many ways, the object behaves like a tibble::tibble().

Customizing skim

skim() is an intentionally simple function, with minimal arguments like summary(). Nonetheless, this package provides two broad approaches to how you can customize skim()'s behavior. You can customize the functions that are called to produce summary statistics with skim_with().

Unicode rendering

If the rendered examples show unencoded values such as <U+2587> you will need to change your locale to allow proper rendering. Please review the *Using Skimr* vignette for more information (vignette("Using_skimr", package = "skimr")).

Otherwise, we export skim_without_charts() to produce summaries without the spark graphs. These are the source of the unicode dependency.

Examples

```
skim(iris)

# Use tidyselect
skim(iris, Species)
skim(iris, starts_with("Sepal"))
skim(iris, where(is.numeric))

# Skim also works groupwise
iris %>%
  dplyr::group_by(Species) %>%
  skim()

# Which five numeric columns have the greatest mean value?
# Look in the `numeric.mean` column.
iris %>%
  skim() %>%
  dplyr::select(numeric.mean) %>%
  dplyr::top_n(5)

# Which of my columns have missing values? Use the base skimmer n_missing.
iris %>%
  skim() %>%
  dplyr::filter(n_missing > 0)

# Use skim_tee to view the skim results and
# continue using the original data.
chickwts %>%
  skim_tee() %>%
  dplyr::filter(feed == "sunflower")

# Produce a summary without spark graphs
```

```
iris %>%
  skim_without_charts()
```

---

skim-attr *Functions for accessing skim_df attributes*

Description

These functions simplify access to attributes contained within a skim_df. While all attributes are read-only, being able to extract this information is useful for different analyses. These functions should always be preferred over calling base R's attribute functions.

Usage

```
data_rows(object)

data_cols(object)

df_name(object)

dt_key(object)

group_names(object)

base_skimmers(object)

skimmers_used(object)
```

Arguments

object	A skim_df or skim_list.

Value

Data contained within the requested skimr attribute.

Functions

- data_rows(): Get the number of rows in the skimmed data frame.
- data_cols(): Get the number of columns in the skimmed data frame.
- df_name(): Get the name of the skimmed data frame. This is only available in contexts where the name can be looked up. This is often not the case within a pipeline.
- dt_key(): Get the key of the skimmed data.table. This is only available in contexts where data is of class data.table.
- group_names(): Get the names of the groups in the original data frame. Only available if the data was grouped. Otherwise, NULL.
- base_skimmers(): Get the names of the base skimming functions used.
- skimmers_used(): Get the names of the skimming functions used, separated by data type.

skim-obj *Test if an object is compatible with* skimr

Description

Objects within skimr are identified by a class, but they require additional attributes and data columns for all operations to succeed. These checks help ensure this. While they have some application externally, they are mostly used internally.

Usage

```
has_type_column(object)

has_variable_column(object)

has_skimr_attributes(object)

has_skim_type_attribute(object)

has_skimmers(object)

is_data_frame(object)

is_skim_df(object)

is_one_skim_df(object)

is_skim_list(object)

could_be_skim_df(object)

assert_is_skim_df(object)

assert_is_skim_list(object)

assert_is_one_skim_df(object)
```

Arguments

object	Any R object.

Details

Most notably, a skim_df has columns skim_type and skim_variable. And has the following special attributes

- data_rows: n rows in the original data

- `data_cols`: original number of columns
- `df_name`: name of the original data frame
- `dt_key`: name of the key if original is a data.table
- `groups`: if there were group variables
- `base_skimmers`: names of functions applied to all skim types
- `skimmers_used`: names of functions used to skim each type

The functions in these checks work like `all.equal()`. The return `TRUE` if the check passes, or otherwise notifies why the check failed. This makes them more useful when throwing errors.

Functions

- `has_type_column()`: Does the object have the `skim_type` column?
- `has_variable_column()`: Does the object have the `skim_variable` column?
- `has_skimr_attributes()`: Does the object have the appropriate `skimr` attributes?
- `has_skim_type_attribute()`: Does the object have a `skim_type` attribute? This makes it a `one_skim_df`.
- `has_skimmers()`: Does the object have skimmers?
- `is_data_frame()`: Is the object a data frame?
- `is_skim_df()`: Is the object a `skim_df`?
- `is_one_skim_df()`: Is the object a `one_skim_df`? This is similar to a `skim_df`, but does not have the `type` column. That is stored as an attribute instead.
- `is_skim_list()`: Is the object a `skim_list`?
- `could_be_skim_df()`: Is this a data frame with `skim_variable` and `skim_type` columns?
- `assert_is_skim_df()`: Stop if the object is not a `skim_df`.
- `assert_is_skim_list()`: Stop if the object is not a `skim_list`.
- `assert_is_one_skim_df()`: Stop if the object is not a `one_skim_df`.

`skim_with` *Set or add the summary functions for a particular type of data*

Description

While skim is designed around having an opinionated set of defaults, you can use this function to change the summary statistics that it returns.

Usage

```
skim_with(
  ...,
  base = sfl(n_missing = n_missing, complete_rate = complete_rate),
  append = TRUE
)
```

Arguments

...	One or more (sfl) skimmer_function_list objects, with an argument name that matches a particular data type.
base	An sfl that sets skimmers for all column types.
append	Whether the provided options should be in addition to the defaults already in skim. Default is TRUE.

Details

skim_with() is a closure: a function that returns a new function. This lets you have several skimming functions in a single R session, but it also means that you need to assign the return of skim_with() before you can use it.

You assign values within skim_with by using the sfl() helper (skimr function list). This helper behaves mostly like dplyr::funs(), but lets you also identify which skimming functions you want to remove, by setting them to NULL. Assign an sfl to each column type that you wish to modify.

Functions that summarize all data types, and always return the same type of value, can be assigned to the base argument. The default base skimmers compute the number of missing values n_missing() and the rate of values being complete, i.e. not missing, complete_rate().

When append = TRUE and local skimmers have names matching the names of entries in the default skim_function_list, the values in the default list are overwritten. Similarly, if NULL values are passed within sfl(), these default skimmers are dropped. Otherwise, if append = FALSE, only the locally-provided skimming functions are used.

Note that append only applies to the typed skimmers (i.e. non-base). See get_default_skimmer_names() for a list of defaults.

Value

A new skim() function. This is callable. See skim() for more details.

Examples

```
# Use new functions for numeric functions. If you don't provide a name,
# one will be automatically generated.
my_skim <- skim_with(numeric = sfl(median, mad), append = FALSE)
my_skim(faithful)

# If you want to remove a particular skimmer, set it to NULL
# This removes the inline histogram
my_skim <- skim_with(numeric = sfl(hist = NULL))
my_skim(faithful)

# This works with multiple skimmers. Just match names to overwrite
my_skim <- skim_with(numeric = sfl(iqr = IQR, p25 = NULL, p75 = NULL))
my_skim(faithful)

# Alternatively, set `append = FALSE` to replace the skimmers of a type.
my_skim <- skim_with(numeric = sfl(mean = mean, sd = sd), append = FALSE)
```

```
# Skimmers are unary functions. Partially apply arguments during assigment.
# For example, you might want to remove NA values.
my_skim <- skim_with(numeric = sfl(iqr = ~ IQR(., na.rm = TRUE)))

# Set multiple types of skimmers simultaneously.
my_skim <- skim_with(numeric = sfl(mean), character = sfl(length))

# Or pass the same as a list, unquoting the input.
my_skimmers <- list(numeric = sfl(mean), character = sfl(length))
my_skim <- skim_with(!!!my_skimmers)

# Use the v1 base skimmers instead.
my_skim <- skim_with(base = sfl(
  missing = n_missing,
  complete = n_complete,
  n = length
))

# Remove the base skimmers entirely
my_skim <- skim_with(base = NULL)
```

---

stats *Summary statistic functions*

Description

skimr provides extensions to a variety of functions with R's stats package to simplify creating summaries of data. All functions are vectorized over the first argument. Additional arguments should be set in the sfl() that sets the appropriate skimmers for a data type. You can use these, along with other vectorized R functions, for creating custom sets of summary functions for a given data type.

Usage

```
n_missing(x)

n_complete(x)

complete_rate(x)

n_whitespace(x)

sorted_count(x)

top_counts(x, max_char = 3, max_levels = 4)

inline_hist(x, n_bins = 8)

n_empty(x)
```

```
min_char(x)

max_char(x)

n_unique(x)

ts_start(x)

ts_end(x)

inline_linegraph(x, length.out = 16)

list_lengths_min(x)

list_lengths_median(x)

list_lengths_max(x)

list_min_length(x)

list_max_length(x)
```

Arguments

x	A vector
max_char	In top = 3, max_levels = 4
max_levels	The maximum number of levels to be displayed.
n_bins	In inline_hist, the number of histogram bars.
length.out	In inline_linegraph, the length of the character time series.

Functions

- n_missing(): Calculate the sum of NA and NULL (i.e. missing) values.
- n_complete(): Calculate the sum of not NA and NULL (i.e. missing) values.
- complete_rate(): Calculate complete values; complete values are not missing.
- n_whitespace(): Calculate the number of rows containing only whitespace values using s+ regex.
- sorted_count(): Create a contingency table and arrange its levels in descending order. In case of ties, the ordering of results is alphabetical and depends upon the locale. NA is treated as a ordinary value for sorting.
- top_counts(): Compute and collapse a contingency table into a single character scalar. Wraps sorted_count().
- inline_hist(): Generate inline histogram for numeric variables. The character length of the histogram is controlled by the formatting options for character vectors.

- `n_empty()`: Calculate the number of blank values in a character vector. A "blank" is equal to "".
- `min_char()`: Calculate the minimum number of characters within a character vector.
- `max_char()`: Calculate the maximum number of characters within a character vector.
- `n_unique()`: Calculate the number of unique elements but remove `NA`.
- `ts_start()`: Get the start for a time series without the frequency.
- `ts_end()`: Get the finish for a time series without the frequency.
- `inline_linegraph()`: Generate inline line graph for time series variables. The character length of the line graph is controlled by the formatting options for character vectors. Based on the function in the pillar package.
- `list_lengths_min()`: Get the length of the shortest list in a vector of lists.
- `list_lengths_median()`: Get the median length of the lists.
- `list_lengths_max()`: Get the maximum length of the lists.
- `list_min_length()`: Get the length of the shortest list in a vector of lists.
- `list_max_length()`: Get the length of the longest list in a vector of lists.

See Also

`get_skimmers()` for customizing the functions called by `skim()`.

---

summary.skim_df *Summary function for skim_df*

Description

This is a method of the generic function `summary()`.

Usage

```
## S3 method for class 'skim_df'
summary(object, ...)
```

Arguments

`object`	a skim dataframe.
`...`	Additional arguments affecting the summary produced. Not used.

Value

A summary of the skim data frame.

Examples

```
a <- skim(mtcars)
summary(a)
```

`to_long` *Create "long" skim output*

Description

Skim results returned as a tidy long data frame with four columns: variable, type, stat and formatted.

Usage

```
to_long(.data, ..., skim_fun = skim)

## Default S3 method:
to_long(.data, ..., skim_fun = skim)

## S3 method for class 'skim_df'
to_long(.data, ..., skim_fun = skim)
```

Arguments

`.data`	A data frame or an object that can be coerced into a data frame.
`...`	Columns to select for skimming. When none are provided, the default is to skim all columns.
`skim_fun`	The skim function used.

Value

A tibble

Methods (by class)

- `to_long(default)`: Skim a data frame and convert the results to a long data frame.
- `to_long(skim_df)`: Transform a skim_df to a long data frame.

Examples

```
to_long(iris)
to_long(skim(iris))
```

Index