

Package ‘tibble’

May 28, 2017

Encoding UTF-8

Version 1.3.3

Title Simple Data Frames

Description Provides a 'tbl_df' class (the 'tibble') that provides stricter checking and better formatting than the traditional data frame.

URL <http://tibble.tidyverse.org/>, <https://github.com/tidyverse/tibble>

BugReports <https://github.com/tidyverse/tibble/issues>

Depends R (>= 3.1.0)

Imports methods, rlang, Rcpp (>= 0.12.3), utils

Suggests covr, dplyr, knitr (>= 1.5.32), microbenchmark, nycflights13, testthat, rmarkdown, withr

LinkingTo Rcpp

LazyData yes

License MIT + file LICENSE

RoxygenNote 6.0.1

VignetteBuilder knitr

NeedsCompilation yes

Author Kirill Müller [aut, cre],
Hadley Wickham [aut],
Romain Francois [ctb],
RStudio [cph]

Maintainer Kirill Müller <krlmlr+r@mailbox.org>

Repository CRAN

Date/Publication 2017-05-28 07:21:53 UTC

R topics documented:

tibble-package	2
add_column	3

add_row	4
as_tibble	5
enframe	7
frame_matrix	8
glimpse	8
has_name	9
is.tibble	10
lst	10
rownames	11
set_tidy_names	12
tribble	14

Index	15
--------------	-----------

tibble-package	<i>tibble: Simple Data Frames</i>
----------------	-----------------------------------

Description

Provides a 'tbl_df' class (the 'tibble') that provides stricter checking and better formatting than the traditional data frame.

Details

The S3 class `tbl_df` wraps a local data frame. The main advantage to using a `tbl_df` over a regular data frame is the printing: `tbl` objects only print a few rows and all the columns that fit on one screen, describing the rest of it as text.

Methods

`tbl_df` implements four important base methods:

print By default only prints the first 10 rows (at most 20), and the columns that fit on screen; see `print.tbl()`

[Never simplifies (drops), so always returns `data.frame`

[[, \$ Calls `.subset2()` directly, so is considerably faster. Returns NULL if column does not exist, \$ warns.

Important functions

`tibble()` and `tribble()` for construction, `as_tibble()` for coercion, and `print.tbl()` and `glimpse()` for display.

Package options

Display options for `tbl_df`, used by `trunc_mat()` and (indirectly) by `print.tbl()`.

`tibble.print_max` Row number threshold: Maximum number of rows printed. Set to `Inf` to always print all rows. Default: 20.

`tibble.print_min` Number of rows printed if row number threshold is exceeded. Default: 10.

`tibble.width` Output width. Default: `NULL` (use width option).

`tibble.max_extra_cols` Number of extra columns printed in reduced form. Default: 100.

Author(s)

Maintainer: Kirill Müller <krlmlr+r@mailbox.org>

Authors:

- Hadley Wickham <hadley@rstudio.com>

Other contributors:

- Romain Francois <romain@r-enthusiasts.com> [contributor]
- RStudio [copyright holder]

See Also

Useful links:

- <http://tibble.tidyverse.org/>
- <https://github.com/tidyverse/tibble>
- Report bugs at <https://github.com/tidyverse/tibble/issues>

add_column

Add columns to a data frame

Description

This is a convenient way to add one or more columns to an existing data frame.

Usage

```
add_column(.data, ..., .before = NULL, .after = NULL)
```

Arguments

`.data` Data frame to append to.

`...` Name-value pairs, all values must have one element for each row in the data frame, or be of length 1

`.before`, `.after`

One-based column index or column name where to add the new columns, default: after last column

See Also

Other addition: [add_row](#)

Examples

```
# add_row -----
df <- tibble(x = 1:3, y = 3:1)

add_column(df, z = -1:1, w = 0)

# You can't overwrite existing columns
## Not run:
add_column(df, x = 4:6)

## End(Not run)
# You can't create new observations
## Not run:
add_column(df, z = 1:5)

## End(Not run)
```

 add_row

Add rows to a data frame

Description

This is a convenient way to add one or more rows of data to an existing data frame. See [tribble\(\)](#) for an easy way to create an complete data frame row-by-row.

Usage

```
add_row(.data, ..., .before = NULL, .after = NULL)
```

Arguments

.data	Data frame to append to.
...	Name-value pairs. If you don't supply the name of a variable, it'll be given the value NA.
.before, .after	One-based row index where to add the new rows, default: after last row

See Also

Other addition: [add_column](#)

Examples

```
# add_row -----
df <- tibble(x = 1:3, y = 3:1)

add_row(df, x = 4, y = 0)

# You can specify where to add the new rows
add_row(df, x = 4, y = 0, .before = 2)

# You can supply vectors, to add multiple rows (this isn't
# recommended because it's a bit hard to read)
add_row(df, x = 4:5, y = 0:-1)

# Absent variables get missing values
add_row(df, x = 4)

# You can't create new variables
## Not run:
add_row(df, z = 10)

## End(Not run)
```

as_tibble

Coerce lists and matrices to data frames.

Description

`as.data.frame()` is effectively a thin wrapper around `data.frame`, and hence is rather slow (because it calls `data.frame()` on each element before `cbinding` together). `as_tibble` is a new S3 generic with more efficient methods for matrices and data frames.

Usage

```
as_tibble(x, ...)
```

S3 method for class 'tbl_df'

```
as_tibble(x, ...)
```

S3 method for class 'data.frame'

```
as_tibble(x, validate = TRUE, ...)
```

S3 method for class 'list'

```
as_tibble(x, validate = TRUE, ...)
```

S3 method for class 'matrix'

```
as_tibble(x, ...)
```

S3 method for class 'table'

```
as_tibble(x, n = "n", ...)

## S3 method for class 'NULL'
as_tibble(x, ...)

## Default S3 method:
as_tibble(x, ...)
```

Arguments

x	A list. Each element of the list must have the same length.
...	Other arguments passed on to individual methods.
validate	When TRUE, verifies that the input is a valid data frame (i.e. all columns are named, and are 1d vectors or lists). You may want to suppress this when you know that you already have a valid data frame and you want to save some time.
n	Name for count column, default: "n".

Details

This is an S3 generic. `tibble` includes methods for data frames (adds `tbl_df` classes), tibbles (returns unchanged input), lists, matrices, and tables. Other types are first coerced via `as.data.frame()` with `stringsAsFactors = FALSE`.

`as_data_frame` and `as_tibble` are aliases.

Examples

```
l <- list(x = 1:500, y = runif(500), z = 500:1)
df <- as_tibble(l)

m <- matrix(rnorm(50), ncol = 5)
colnames(m) <- c("a", "b", "c", "d", "e")
df <- as_tibble(m)

# as_tibble is considerably simpler than as.data.frame
# making it more suitable for use when you have things that are
# lists
## Not run:
if (requireNamespace("microbenchmark", quiet = TRUE)) {
  l2 <- replicate(26, sample(letters), simplify = FALSE)
  names(l2) <- letters
  microbenchmark::microbenchmark(
    as_tibble(l2, validate = FALSE),
    as_tibble(l2),
    as.data.frame(l2)
  )
}

if (requireNamespace("microbenchmark", quiet = TRUE)) {
  m <- matrix(runif(26 * 100), ncol = 26)
  colnames(m) <- letters
```

```
microbenchmark::microbenchmark(  
  as_tibble(m),  
  as.data.frame(m)  
)  
}  
  
## End(Not run)
```

enframe

Converting atomic vectors to data frames, and vice versa

Description

enframe() converts named atomic vectors or lists to two-column data frames. For unnamed vectors, the natural sequence is used as name column.

deframe() converts two-column data frames to a named vector or list, using the first column as name and the second column as value.

Usage

```
enframe(x, name = "name", value = "value")
```

```
deframe(x)
```

Arguments

x	An atomic vector (for enframe()) or a data frame (for deframe())
name, value	Names of the columns that store the names and values

Value

A [tibble](#)

Examples

```
enframe(1:3)  
enframe(c(a = 5, b = 7))
```

frame_matrix	<i>Row-wise matrix creation</i>
--------------	---------------------------------

Description

Create matrices laying out the data in rows, similar to `matrix(..., byrow = TRUE)`, with a nicer-to-read syntax. This is useful for small matrices, e.g. covariance matrices, where readability is important. The syntax is inspired by `tribble()`.

Usage

```
frame_matrix(...)
```

Arguments

... Arguments specifying the structure of a `frame_matrix`. Column names should be formulas, and may only appear before the data.

Value

A [matrix](#).

Examples

```
frame_matrix(  
  ~col1, ~col2,  
  1,     3,  
  5,     2  
)
```

glimpse	<i>Get a glimpse of your data.</i>
---------	------------------------------------

Description

This is like a transposed version of `print`: columns run down the page, and data runs across. This makes it possible to see every column in a data frame. It's a little like `str()` applied to a data frame but it tries to show you as much data as possible. (And it always shows the underlying data, even when applied to a remote data source.)

Usage

```
glimpse(x, width = NULL, ...)
```


Arguments

x	An object to glimpse at.
width	Width of output: defaults to the setting of the option <code>tibble.width</code> (if finite) or the width of the console.
...	Other arguments passed onto individual methods.

Value

x original x is (invisibly) returned, allowing `glimpse()` to be used within a data pipe line.

S3 methods

`glimpse` is an S3 generic with a customised method for `tbls` and `data.frames`, and a default method that calls `str()`.

Examples

```
glimpse(mtcars)

if (!requireNamespace("nycflights13", quietly = TRUE))
  stop("Please install the nycflights13 package to run the rest of this example")

glimpse(nycflights13::flights)
```

has_name	<i>Does an object have an element with this name?</i>
----------	---

Description

This function returns a logical value that indicates if a data frame or another named object contains an element with a specific name.

Usage

```
has_name(x, name)
```

Arguments

x	A data frame or another named object
name	Element name(s) to check

Details

Unnamed objects are treated as if all names are empty strings. NA input gives FALSE as output.

Value

A logical vector of the same length as name

<code>is.tibble</code>	<i>Test if the object is a tibble.</i>
------------------------	--

Description

Test if the object is a tibble.

Usage

```
is.tibble(x)
```

```
is_tibble(x)
```

Arguments

`x` An object

Value

TRUE if the object inherits from the `tbl_df` class.

<code>lst</code>	<i>Build a list</i>
------------------	---------------------

Description

`lst()` is similar to `list()`, but like `tibble()`, it evaluates its arguments lazily and in order, and automatically adds names.

`tibble()` is a trimmed down version of `data.frame()` that:

- Never coerces inputs (i.e. strings stay as strings!).
- Never adds `row.names`.
- Never munges column names.
- Only recycles length 1 inputs.
- Evaluates its arguments lazily and in order.
- Adds `tbl_df` class to output.
- Automatically adds column names.

Usage

```
lst(...)
```

```
lst_(xs)
```

```
tibble(...)
```

```
tibble_(xs)
```

Arguments

- ... A set of name-value pairs. Arguments are evaluated sequentially, so you can refer to previously created variables.
- xs A list of unevaluated expressions created with `~`, `quote()`, or (deprecated) `lazyeval::lazy()`.

See Also

[as_tibble\(\)](#) to turn an existing list into a data frame.

Examples

```
lst(n = 5, x = runif(n))

# You can splice-unquote a list of quotes and formulas
lst(!!! list(n = rlang::quo(2 + 3), y = quote(runif(n))))

a <- 1:5
tibble(a, b = a * 2)
tibble(a, b = a * 2, c = 1)
tibble(x = runif(10), y = x * 2)

lst(n = 5, x = runif(n))

# tibble never coerces its inputs
str(tibble(letters))
str(tibble(x = list(diag(1), diag(2))))

# or munges column names
tibble(`a + b` = 1:5)

# You can splice-unquote a list of quotes and formulas
tibble(!!! list(x = rlang::quo(1:10), y = quote(x * 2)))

# data frames can only contain 1d atomic vectors and lists
# and can not contain POSIXlt
## Not run:
tibble(x = tibble(1, 2, 3))
tibble(y = strptime("2000/01/01", "%x"))

## End(Not run)
```

Description

While a tibble can have row names (e.g., when converting from a regular data frame), they are removed when subsetting with the `[]` operator. A warning will be raised when attempting to assign non-NULL row names to a tibble. Generally, it is best to avoid row names, because they are basically

a character column with different semantics to every other column. These functions allow you to detect if a data frame has row names (`has_rownames()`), remove them (`remove_rownames()`), or convert them back-and-forth between an explicit column (`rownames_to_column()` and `column_to_rownames()`). Also included is `rowid_to_column()` which adds a column at the start of the dataframe of ascending sequential row ids starting at 1. Note that this will remove any existing row names.

Usage

```
has_rownames(df)

remove_rownames(df)

rownames_to_column(df, var = "rowname")

rowid_to_column(df, var = "rowid")

column_to_rownames(df, var = "rowname")
```

Arguments

<code>df</code>	A data frame
<code>var</code>	Name of column to use for rownames.

Details

In the printed output, the presence of row names is indicated by a star just above the row numbers.

Examples

```
has_rownames(mtcars)
has_rownames(iris)
has_rownames(remove_rownames(mtcars))

head(rownames_to_column(mtcars))

mtcars_tbl <- as_tibble(rownames_to_column(mtcars))
mtcars_tbl
column_to_rownames(as.data.frame(mtcars_tbl))
```

set_tidy_names

Repair object names.

Description

`tidy_names()` ensures its input has non-missing and unique names (duplicated names get a suffix of the format `..#` where `#` is the position in the vector). Valid names are left unchanged, with the exception that existing suffixes are reorganized.

tidy_names() is the workhorse behind set_tidy_names(), it treats the argument as a string to be used to name a data frame or a vector.

repair_names() is an older version with different renaming heuristics, kept for backward compatibility. New code should prefer tidy_names().

Usage

```
set_tidy_names(x, syntactic = FALSE, quiet = FALSE)
```

```
tidy_names(name, syntactic = FALSE, quiet = FALSE)
```

```
repair_names(x, prefix = "V", sep = "")
```

Arguments

x	A named vector.
syntactic	Should all names be made syntactically valid via make.names() ?
quiet	If TRUE suppresses output from this function
name	A character vector representing names.
prefix	A string, the prefix to use for new column names.
sep	A string inserted between the column name and de-duplicating number.

Value

x with valid names.

Examples

```
# Works for lists and vectors, too:
set_tidy_names(3:5)
set_tidy_names(list(3, 4, 5))

# Clean data frames are left unchanged:
set_tidy_names(mtcars)

# By default, all rename operations are printed to the console:
tbl <- as_tibble(structure(list(3, 4, 5), class = "data.frame"),
                 validate = FALSE)
set_tidy_names(tbl)

# Optionally, names can be made syntactic:
tidy_names("a b", syntactic = TRUE)
```

`tribble`*Row-wise tibble creation*

Description

Create [tibles](#) using an easier to read row-by-row layout. This is useful for small tables of data where readability is important. Please see [tibble-package](#) for a general introduction.

Usage

```
tribble(...)
```

Arguments

... Arguments specifying the structure of a tibble. Variable names should be formulas, and may only appear before the data.

Details

`frame_data()` is an older name for `tribble()`. It will eventually be phased out.

Value

A [tibble](#).

Examples

```
tribble(
  ~colA, ~colB,
  "a", 1,
  "b", 2,
  "c", 3
)

# tribble will create a list column if the value in any cell is
# not a scalar
tribble(
  ~x, ~y,
  "a", 1:3,
  "b", 4:6
)
```

Index

`.subset2()`, 2

`add_column`, 3, 4
`add_row`, 4, 4
`as.data.frame()`, 5
`as.tibble(as_tibble)`, 5
`as_data_frame(as_tibble)`, 5
`as_tibble`, 5
`as_tibble()`, 2, 11

`cbind`, 5
`column_to_rownames(rownames)`, 11

`data.frame()`, 5, 10
`data_frame(lst)`, 10
`data_frame_(lst)`, 10
`deframe(enframe)`, 7

`enframe`, 7

`frame_data(tribble)`, 14
`frame_matrix`, 8

`glimpse`, 8
`glimpse()`, 2

`has_name`, 9
`has_rownames(rownames)`, 11

`is.tibble`, 10
`is_tibble(is.tibble)`, 10

`lazyeval::lazy()`, 11
`list()`, 10
`lst`, 10
`lst_(lst)`, 10

`make.names()`, 13
`matrix`, 8

`print.tbl()`, 2, 3

`quote()`, 11

`remove_rownames(rownames)`, 11
`repair_names(set_tidy_names)`, 12
`rowid_to_column(rownames)`, 11
`rownames`, 11
`rownames_to_column(rownames)`, 11

`set_tidy_names`, 12
`str()`, 8, 9

`tibble`, 7, 14
`tibble(lst)`, 10
`tibble()`, 2
`tibble-package`, 2, 14
`tibble_(lst)`, 10
`tidy_names(set_tidy_names)`, 12
`tribble`, 14
`tribble()`, 2, 4, 8
`trunc_mat()`, 3