Package: data.trame (via r-universe)

May 26, 2026

Type Package

Version 0.9.0

Title 'SciViews::R' - A Better Data Frame

Description The 'data.trame' object is an hybrid between 'data.table', 'tibble' and 'data.frame'. It enhances the 'data.frame' with the speed of 'data.table' and the nice features of 'tibble' (petty printing, stricter rules...).

Maintainer Philippe Grosjean <phgrosjean@sciviews.org>

Depends R (>= 4.4.0)

Imports checkmate (>= 2.3.0), collapse (>= 2.1.0), data.table (>= 1.16.0), pillar (>= 1.10.0), rlang (>= 1.1.0), tibble (>= 3.2.1), utils (>= 4.4.0)

Suggests covr (>= 3.5.0), knitr (>= 1.42), rmarkdown (>= 2.21), spelling (>= 2.2.1), testthat (>= 3.0.0)

License MIT + file LICENSE

URL https://github.com/SciViews/data.trame, https://www.sciviews.org/data.trame/, https://sciviews.r-universe.dev/data.trame

BugReports https://github.com/SciViews/data.trame/issues

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

VignetteBuilder knitr

Encoding UTF-8

Language en-US

ByteCompile yes

Config/testthat/edition 3

Repository https://sciviews.r-universe.dev

Date/Publication 2025-08-29 06:12:57 UTC

RemoteUrl https://github.com/SciViews/data.trame

RemoteRef HEAD

RemoteSha ea8b68629faeb3c5ae89791eca1e65c4cc75ed02

Contents

`data.trame-package` *A better 'data.frame' for 'SciViews::R'*

Description

The 'data.trame' object is an hybrid between 'data.table', tibble' and 'data.frame'. It enhances the 'data.frame' with the speed of 'data.table' and the nice features of 'tibble' (petty printing, stricter rules...).

Important functions

- `data.trame()` to construct a data.trame object,
- `as.data.trame()` to coerce into a data.trame object.
- `is.data.trame()` to test for data.trame objects.
- Methods for data.frame objects.

Author(s)

Maintainer: Philippe Grosjean <phgrosjean@sciviews.org> (ORCID)

See Also

Useful links:

- https://github.com/SciViews/data.trame
- https://www.sciviews.org/data.trame/
- https://sciviews.r-universe.dev/data.trame
- Report bugs at https://github.com/SciViews/data.trame/issues

`data.trame` *Build, coerce and test for 'data.trame' objects*

Description

Build, coerce and test for 'data.trame' objects

Usage

```
data.trame(
  ...,
  .key = NULL,
  .rows = NULL,
  .name_repair = c("check_unique", "unique", "universal", "minimal")
)

as.data.trame(
  x,
  .key = NULL,
  .rows = NULL,
  .rownames = NA,
  .name_repair = c("check_unique", "unique", "universal", "minimal"),
  ...
)

## Default S3 method:
as.data.trame(
  x,
  .key = NULL,
  .rows = NULL,
  .rownames = NA,
  .name_repair = c("check_unique", "unique", "universal", "minimal"),
  ...
)

## S3 method for class 'list'
as.data.trame(
  x,
  .key = NULL,
  .rows = NULL,
  .rownames = NA,
  .name_repair = c("check_unique", "unique", "universal", "minimal"),
  ...
)

## S3 method for class 'data.frame'
as.data.trame(
```

```
  x,
  .key = NULL,
  .rows = NULL,
  .rownames = NA,
  .name_repair = c("check_unique", "unique", "universal", "minimal"),
  ...
)

## S3 method for class 'data.trame'
as.data.trame(
  x,
  .key = NULL,
  .rows = NULL,
  .rownames = NA,
  .name_repair = c("check_unique", "unique", "universal", "minimal"),
  ...
)

## S3 method for class 'data.table'
as.data.trame(
  x,
  .key = NULL,
  .rows = NULL,
  .rownames = NA,
  .name_repair = c("check_unique", "unique", "universal", "minimal"),
  ...
)

## S3 method for class 'data.trame'
as.data.table(x, keep.rownames = FALSE, ...)

## S3 method for class 'data.trame'
as.data.frame(x, row.names = NULL, optional = TRUE, ..., keep.attr = FALSE)

## S3 method for class 'data.trame'
as_tibble(
  x,
  ...,
  .rows = NULL,
  .name_repair = c("check_unique", "unique", "universal", "minimal"),
  rownames = NULL,
  keep.attr = FALSE
)

is.data.trame(x)
```

Arguments

...	A set of name-value pairs that constitute the 'data.trame'.

.key	A character vector with the name of the columns to use as key (sorting the data)
.rows	The number of rows in the final 'data.trame'. Useful to create a 0-column object, or for additional check.
.name_repair	Treatment of problematic column names: could be "check_unique" (default, no name repair but check they are unique), "unique" (make sure names are unique), "universal" (make sure names are unique and syntactically correct), "minimal" (no check or name repair except for existence), or a function for custom name repair, e.g., .name.repair = make.names for base R style.
x	An object.
.rownames	The name of the column that holds the row names of the original object, if any. If NA (default), row names are left intact. If NULL row names are removed.
keep.rownames	For compatibility with the generic, but not used here
row.names	A character vector of row names, or NULL (default).
optional	If TRUE, the row names are not checked for uniqueness. If FALSE, the row names are corrected with make.names().
keep.attr	If TRUE, the attributes of the data frame are kept.
rownames	NULL remove row names, NA (default) leaves them, or a single string to create a column with that name.

Value

A 'data.trame' object, which is indeed a 'data.trame'/'data.frame' object, thus subclassing 'data.frame'. is.data.trame() returns TRUE if the object is a 'data.trame', and FALSE otherwise.

Examples

```
dtrm <- data.trame(
  a = 1:3,
  b = letters[1:3],
  c = factor(LETTERS[1:3]), .key = c('a', 'b'), .rows = 3
)
is.data.trame(dtrm)
```

---

dcast *Fast dcast for data.trame*

Description

dcast() transforms a data.trame from long to wide format, a little bit like tidyr::pivot_wider(). See data.table::dcast() for explanations.

Usage

```
dcast(
  data,
  formula,
  fun.aggregate = NULL,
  ...,
  margins = NULL,
  subset = NULL,
  fill = NULL,
  value.var = guess(data)
)

## S3 method for class 'data.trame'
dcast(
  data,
  formula,
  fun.aggregate = NULL,
  sep = "_",
  ...,
  margins = NULL,
  subset = NULL,
  fill = NULL,
  drop = TRUE,
  value.var = guess(data),
  verbose = getOption("datatable.verbose")
)
```

Arguments

`data`	A data.trame object.
`formula`	A formula LHS ~ RHS, see `data.table::dcast()` for details.
`fun.aggregate`	Te function to aggregate multiple data before dcasting.
`...`	Arguments passed to the aggregating function.
`margins`	Note implemented yet.
`subset`	Should the dcasting be done on a subset of the data?
`fill`	Value with which to fill missing cells.
`value.var`	The name of the column to use as value variable. If not provided it is "guessed" the `guess()` internal function is used to build a good default name.
`sep`	Character vector of length 1, used to separate parts of the variable names (_ by default).
`drop`	`FALSE` should dcast include all missing combinations?
`verbose`	Not used yet.

Value

A keyed data.trame is returned with the dcasted data.

See Also

data.table::dcast(), melt()

Examples

```
# Adapted from first example of ?dcast.data.table
ChickWeight = as.data.trame(ChickWeight)
ChickWeight <- set_names(ChickWeight, tolower(names(ChickWeight)))
dtrm <- melt(ChickWeight, id.vars = 2:4)
dcast(dtrm, time ~ variable, fun.aggregate = mean)
```

---

let_data.table_to_data.trame

Switch back and forth quickly between data.trame and data.table by reference

---

Description

These functions just change the class of the object by reference (data.trame and data.table objects are, internally, identical except for their class). These functions are intended only for programmers! There is no check that the object is correct before the change.

Usage

```
let_data.table_to_data.trame(x)

let_data.trame_to_data.table(x)
```

Arguments

x	A valid data.trame or data.table object

Value

A data.table or data.trame object.

Examples

```
dtrm <- data.trame(a = 1:3, b = letters[1:3], c = factor(LETTERS[1:3]))
class(dtrm)
let_data.trame_to_data.table(dtrm)
class(dtrm)
let_data.table_to_data.trame(dtrm)
class(dtrm)

# Whenever you need to use a data.table method on a data.trame in a function,
# you can do something like this:
test_fun <- function(x, i, expr) {
```

```
  force(i) # Make sure i is evaluated before changing the class of x
  # value will be evaluated with x being a data.table here!
  let_data.trame_to_data.table(x)
  on.exit(let_data.table_to_data.trame(x))
  print(class(x)) # Internally, it is now a data.table
  expr
}
test_fun(dtrm, 1:2, class(dtrm))
class(dtrm) # Back to data.trame
```

`melt` *Fast melt for data.trame*

Description

`melt()` is reshaping wide-to-long, notb unlike tidyr::pivot_longer()‘ for data.trame objects. See `data.table::melt()` for explanations.

Usage

```
melt(data, ..., na.rm = FALSE, value.name = "value")

## S3 method for class 'data.trame'
melt(
  data,
  id.vars,
  measure.vars,
  variable.name = "variable",
  value.name = "value",
  ...,
  na.rm = FALSE,
  variable.factor = TRUE,
  value.factor = FALSE,
  verbose = getOption("datatable.verbose")
)
```

Arguments

`data`	A data.trame object.
`...`	Arguments passed to other methods.
`na.rm`	Should `NA` values be removed? Default is `FALSE`.
`value.name`	Name for the molten data values column(s).
`id.vars`	Vector of `id` variables.
`measure.vars`	Measure variables for melting. Can be missing.
`variable.name`	Name (default `variable`) of output column containing the information about melted columns.

variable.factor
: If TRUE, the variable column is converted to a factor, otherwise, it is a character column.

value.factor
: If TRUE, the value column is converted to a factor, else, it is left unchanged.

verbose
: TRUE turns on status and information messages.

Value

An unkeyed data.trame containing the molten data.

See Also

data.table::melt(), dcast()

Examples

```
# Adapted from first example of ?melt.data.table
set.seed(45)
library(data.trame)
dtrm <- data.trame(
  i_1 = c(1:5, NA),
  n_1 = c(NA, 6, 7, 8, 9, 10),
  f_1 = factor(sample(c(letters[1:3], NA), 6L, TRUE)),
  f_2 = ordered(c("z", "a", "x", "c", "x", "x")),
  c_1 = sample(c(letters[1:3], NA), 6L, TRUE),
  c_2 = sample(c(LETTERS[1:2], NA), 6L, TRUE),
  d_1 = as.Date(c(1:3,NA,4:5), origin = "2013-09-01"),
  d_2 = as.Date(6:1, origin = "2012-01-01")
)
# add a couple of list cols
dtrm$l_1 <- dtrm[, ~list(c = list(rep(i_1, sample(5, 1L)))), by = ~i_1]$c
dtrm$l_2 <- dtrm[, ~list(c = list(rep(c_1, sample(5, 1L)))), by = ~i_1]$c

# id.vars, measure.vars as character/integer/numeric vectors
melt(dtrm, id.vars = 1:2, measure.vars = "f_1")
```

---

methods *Various methods implemented for data.trame objects.*

Description

These methods handle data.trame objects correctly, so that they remain internally consistent with data.tables.

Usage

```
## S3 method for class 'data.trame'
head(x, n = 6L, ...)

## S3 method for class 'data.trame'
tail(x, n = 6L, ...)

## S3 replacement method for class 'data.trame'
names(x) <- value

set_names()

let_names(x, value)

## S3 replacement method for class 'data.trame'
row.names(x) <- value

set_row_names(x, value)

let_row_names(x, value)

## S3 method for class 'data.trame'
edit(name, ...)

## S3 method for class 'data.trame'
cbind(
  x,
  ...,
  keep.rownames = FALSE,
  check.names = FALSE,
  key = NULL,
  stringsAsFactors = FALSE
)

## S3 method for class 'data.trame'
rbind(
  x,
  ...,
  use.names = TRUE,
  fill = FALSE,
  idcol = NULL,
  ignore.attr = FALSE
)
```

Arguments

x	A data.trame object.
n	The number of rows to keep.

...	Further parameters (not used yet).
value	The value passed to the method.
name	The name of the data.trame to edit.
keep.rownames	If TRUE, the row names are kept as a column
check.names	If TRUE, the names of the columns are checked
key	The key to set on the resulting data.trame. If NULL, no key is set.
stringsAsFactors	If TRUE, character columns are converted to factors.
use.names	If TRUE, the names of the columns are matched. If FALSE, match is done by position. if "check", warn if names do not match.
fill	If TRUE, fill missing columns with NA. By default FALSE.
idcol	Create a column with ids showing where the data come from. With TRUE, the column is named id. With idcol = "col_name", it has that name.
ignore.attr	If TRUE, ignore attributes when binding.

Value

head() and tail() return truncated data.trame objects (n first or last rows). names(dtrm) <- value and set_names() both set names (colnames). let_names() change names by reference. row.names(dtrm) <- value and set_row_names() both set the row names of a data.trame. However, only the second one keeps the selfref pointer integrity. let_row_names() is a faster version, but it changes the row names by reference. With all let_xxx() functions, you need to take extra care to avoid unexpected side effects, see examples. cbind() combines data.trames by columns, rbind() combines them by rows.

Examples

```
dtrm <- data.trame(
  a = 1:10,
  b = letters[1:10],
  c = factor(LETTERS[1:10]), .key = c('a', 'b')
)
head(dtrm)
tail(dtrm, n = 3L)
cbind(dtrm, dtrm)
rbind(dtrm, dtrm)
dtrm2 <- set_row_names(dtrm, paste("row", letters[1:10]))
dtrm2
dtrm # Not changed
# Take care with let_xxx() functions: it propagates changes to other
# data.trames if you did not used copy()!
dtrm2 <- dtrm
dtrm3 <- data.table::copy(dtrm)
let_row_names(dtrm2, paste("row", letters[11:20]))
dtrm2 # OK
dtrm # Also changed!
dtrm3 # Not changed, because created using copy()
```

`print.data.trame` *Printing data.trames*

Description

A data.trame prints almost like a tibble.

Usage

```
## S3 method for class 'data.trame'
print(
  x,
  width = NULL,
  ...,
  n = NULL,
  max_extra_cols = NULL,
  max_footer_lines = NULL
)

## S3 method for class 'data.trame'
format(
  x,
  width = NULL,
  ...,
  n = NULL,
  max_extra_cols = NULL,
  max_footer_lines = NULL
)

## S3 method for class 'datatrame'
obj_sum(x)

## S3 method for class 'data.trame'
obj_sum(x)

## S3 method for class 'datatrame'
tbl_sum(x, ...)

## S3 method for class 'data.trame'
tbl_sum(x, ...)

## S3 method for class 'data.trame'
tbl_nrow(x, ...)

## S3 method for class 'data.trame'
str(object, ..., indent.str = " ", nest.lev = 0)
```

Arguments

x	A data.trame object.
width	The width of the text output. If NULL, the default, getOption("width") is used.
...	Additional arguments passed to format().
n	The number of rows to print. If NULL, a default number is used.
max_extra_cols	The maximum number of extra columns to print abbreviated. If NULL by default, a reasonable default value is used.
max_footer_lines	Maximum number of lines in the footer. If NULL, a reasonable default value is used.
object	A data.trame.
indent.str	The string used for indentation.
nest.lev	The current nesting level, used for recursive printing.

Examples

```
dtrm <- data.trame(
  a = -1:3,
  b = letters[1:5],
  c = factor(LETTERS[1:5]),
  d = c(TRUE, FALSE, TRUE, NA, TRUE), .key = c('a', 'b'), .rows = 5
)
dtrm
str(dtrm)
```

---

subsetting *Subsetting data.trames*

Description

Subsetting data.trames uses a syntax similar to tibble, or formulas for i, j, and possibly by or keyby to use the data.table syntax instead.

Usage

```
## S3 method for class 'data.trame'
x[i, j, by, keyby, with = TRUE, drop = FALSE, ...]

set_(x, i, j, value, byref = FALSE)

let_(x, i = NULL, j = seq_along(x), value)
```

Arguments

x	A data.trame object.
i	Selection of rows by indices, negative indices, logical or a formula
j	Selection of columns by indices, negative indices, logical, names or a formula (both i and j must be formulas simultaneously). If := is used in the formula to create one or more new variables by reference, the expression must be placed between {} to avoid operators precedence issues, or better: := could be just replaced by ~.
by	Grouping columns (must be a formula and j must be also provided as a formula)
keyby	Either TRUE/FALSE if by is provided, or a formula (and j must also be provided as a formula)
with	Logical, whether to evaluate j in the data.trame if TRUE or in the calling environment if FALSE (default is TRUE). with = FALSE is similar to tibble subsetting and it is forced when i or j are not formulas.
drop	Coerce to a vector if the returned data.trame only has one column
...	Further arguments passed to the underlying data.table subsetting
value	The value to insert as subassignment in a data.trame object.
byref	Logical, whether to use by reference or not (FALSE by default).

Value

A data.trame object, or a vector if drop = TRUE and the result has only one column.

Examples

```
dtrm <- data.trame(
  a = 1:3,
  b = letters[1:3],
  c = factor(LETTERS[1:3])
)
# Subsetting rows, the tibble-way
dtrm[1:2, ]
dtrm[-1, ]
dtrm[c(TRUE, FALSE, TRUE), ]
# On the contrary to data.table, providing only one arg, means subsetting
# columns (like for data.frame or tibble)
dtrm[c(TRUE, FALSE, TRUE)]
dtrm[dtrm$a > 1, ] # Must fully qualify the column name
# Subsetting the data.table way, with formulas: no fully qualification needed
dtrm[~a > 1, ]

# Subsetting the columns, the tibble way
dtrm[, 1:2]
dtrm[, -1]
dtrm[, c(TRUE, FALSE, TRUE)]
dtrm[, c("a", "b")]
# You must set drop = TRUE explicitly to return a vector
dtrm[, 2] # Still a data.trame, like tibble, but unlike the data.frame method
```

```
dtrm[, 2, drop = TRUE] # Now a vector
# The selection is referentially transparent, i.e., you can do:
sel <- c("c", "b")
dtrm[, sel]
# Subsetting the columns, the data.table way, with formulas
dtrm[~1:2, ~.(b)]
dtrm[~1:2, ~b] # If not enclosed in .(), returns a vector instead
# Precautions are needed here because it is NOT referentially transparent:
dtrm[, ~..sel] # In data.table language, this is how you access `sel`

# Extended data.table syntax using i, j, by, or keyby with formulas
# Warning: due to precedence of operators, you must use braces here!
dtrm[, ~{d := paste0(b, c)}] # Changed in place (by reference!)
# Another form that does not need braces, but is less readable:
dtrm[, ~`:=`(e, paste0(b, a))]
# or equivalently:
dtrm[, ~let(e = paste0(b, a))]
# In this case, it is much better to just replace `:=` by `~`, but internally
# it uses set(). It is faster, but much more limited and cannot use by or
# or keyby:
dtrm[, f ~ paste0(c, a)]
# One can also use standard evaluation in that case using with = FALSE
dtrm[, f ~ paste0(dtrm$c, dtrm$a), with = FALSE]
#
# Take care when you provide only one argument:
# If it is a formula, the data.table syntax is used (select rows)
# otherwise, the data.frame syntax applies, and columns are selected!
dtrm[1:2] # All rows and 2 first columns
dtrm[~1:2] # All columns and 2 first rows!

# For $, on the contrary to data.frame/data.table, but like tibble,
# no partial match is allowed (returns NULL with a warning)
dtrm$count <- dtrm$c
names(dtrm)
dtrm$count #OK
#dtrm$co # Not OK, no partial match allowed
```

Index