

Package ‘dplyr’

August 29, 2016

Title Data Table Back-End for 'dplyr'

Version 0.0.1

Description This implements the data table back-end for 'dplyr' so that you can seamlessly use data table and 'dplyr' together.

License GPL (>= 2)

Imports dplyr (>= 0.5.0), data.table, lazyeval

Suggests Lahman, nycflights13, testthat, covr

LazyData true

RoxygenNote 5.0.1

URL <https://github.com/hadley/dplyr>

BugReports <https://github.com/hadley/dplyr/issues>

NeedsCompilation no

Author Hadley Wickham [aut, cre],
RStudio [cph]

Maintainer Hadley Wickham <hadley@rstudio.com>

Repository CRAN

Date/Publication 2016-06-27 06:59:59

R topics documented:

grouped_dt	2
join.tbl_dt	2
src_dt	4
tbl_dt	4
Index	6

grouped_dt	<i>A grouped data table.</i>
------------	------------------------------

Description

The easiest way to create a grouped data table is to call the `group_by` method on a data table or `tbl`: this will take care of capturing the unevaluated expressions for you.

Usage

```
grouped_dt(data, vars, copy = TRUE)
```

```
is.grouped_dt(x)
```

Arguments

<code>data</code>	a <code>tbl</code> or data frame.
<code>vars</code>	a list of quoted variables.
<code>copy</code>	If <code>TRUE</code> , will make copy of input.
<code>x</code>	an object to check

Examples

```
library(dplyr, warn.conflicts = FALSE)
if (require("nycflights13")) {
  flights_dt <- tbl_dt(flights)
  group_size(group_by(flights_dt, year, month, day))
  group_size(group_by(flights_dt, dest))

  monthly <- group_by(flights_dt, month)
  summarise(monthly, n = n(), delay = mean(arr_delay))
}
```

join.tbl_dt	<i>Join data table tbls.</i>
-------------	------------------------------

Description

See [join](#) for a description of the general purpose of the functions.

Usage

```
## S3 method for class 'data.table'
inner_join(x, y, by = NULL, copy = FALSE, ...)

## S3 method for class 'data.table'
left_join(x, y, by = NULL, copy = FALSE, ...)

## S3 method for class 'data.table'
right_join(x, y, by = NULL, copy = FALSE, ...)

## S3 method for class 'data.table'
semi_join(x, y, by = NULL, copy = FALSE, ...)

## S3 method for class 'data.table'
anti_join(x, y, by = NULL, copy = FALSE, ...)

## S3 method for class 'data.table'
full_join(x, y, by = NULL, copy = FALSE, ...)
```

Arguments

x, y	tbls to join
by	a character vector of variables to join by. If NULL, the default, join will do a natural join, using all variables with common names across the two tables. A message lists the variables so that you can check they're right (to suppress the message, simply explicitly list the variables that you want to join). To join by different variables on x and y use a named vector. For example, <code>by = c("a" = "b")</code> will match x.a to y.b.
copy	If x and y are not from the same data source, and copy is TRUE, then y will be copied into the same src as x. This allows you to join tables across srcs, but it is a potentially expensive operation so you must opt into it.
...	Included for compatibility with generic; otherwise ignored.

Examples

```
library(dplyr, warn.conflicts = FALSE)

if (require("Lahman")) {
  batting_dt <- tbl_dt(Batting)
  person_dt <- tbl_dt(Master)

  # Inner join: match batting and person data
  inner_join(batting_dt, person_dt)

  # Left join: keep batting data even if person missing
  left_join(batting_dt, person_dt)

  # Semi-join: find batting data for top 4 teams, 2010:2012
```

```

grid <- expand.grid(
  teamID = c("WAS", "ATL", "PHI", "NYA"),
  yearID = 2010:2012)
top4 <- semi_join(batting_dt, grid, copy = TRUE)

# Anti-join: find batting data with out player data
anti_join(batting_dt, person_dt)
}

```

src_dt	<i>A local data table source.</i>
--------	-----------------------------------

Description

This is mainly useful for testing, since makes it possible to refer to local and remote tables using exactly the same syntax.

Usage

```
src_dt(pkg = NULL, env = NULL)
```

Arguments

pkg, env	Either the name of a package or an environment object in which to look for objects.
----------	---

tbl_dt	<i>Create a data table tbl.</i>
--------	---------------------------------

Description

A data table tbl wraps a local data table.

Usage

```
tbl_dt(data, copy = TRUE)
```

Arguments

data	a data table
copy	If the input is a data.table, copy it?

Examples

```

ds <- tbl_dt(mtcars)
ds
data.table::as.data.table(ds)

library(dplyr, warn.conflicts = FALSE)
if (require("nycflights13")) {
  flights2 <- tbl_dt(flights)
  flights2 %>% filter(month == 1, day == 1, dest == "DFW")
  flights2 %>% select(year:day)
  flights2 %>% rename(Year = year)
  flights2 %>%
    summarise(
      delay = mean(arr_delay, na.rm = TRUE),
      n = length(arr_delay)
    )
  flights2 %>%
    mutate(gained = arr_delay - dep_delay) %>%
    select(ends_with("delay"), gained)
  flights2 %>%
    arrange(dest, desc(arr_delay))

  by_dest <- group_by(flights2, dest)

  filter(by_dest, arr_delay == max(arr_delay, na.rm = TRUE))
  summarise(by_dest, arr = mean(arr_delay, na.rm = TRUE))

  # Normalise arrival and departure delays by airport
  by_dest %>%
    mutate(arr_z = scale(arr_delay), dep_z = scale(dep_delay)) %>%
    select(starts_with("arr"), starts_with("dep"))

  arrange(by_dest, desc(arr_delay))
  select(by_dest, -(day:tailnum))
  rename(by_dest, Year = year)

  # All manip functions preserve grouping structure, except for summarise
  # which removes a grouping level
  by_day <- group_by(flights2, year, month, day)
  by_month <- summarise(by_day, delayed = sum(arr_delay > 0, na.rm = TRUE))
  by_month
  summarise(by_month, delayed = sum(delayed))

  # You can also manually ungroup:
  ungroup(by_day)
}

```

Index

`.datatable.aware (tbl_dt)`, 4
`anti_join.data.table (join.tbl_dt)`, 2
`full_join.data.table (join.tbl_dt)`, 2
`grouped_dt`, 2
`inner_join.data.table (join.tbl_dt)`, 2
`is.grouped_dt (grouped_dt)`, 2
`join`, 2
`join.tbl_dt`, 2
`left_join.data.table (join.tbl_dt)`, 2
`right_join.data.table (join.tbl_dt)`, 2
`semi_join.data.table (join.tbl_dt)`, 2
`src_dt`, 4
`tbl_dt`, 4