Package ‘dtplyr’
January 23, 2020

**Title**  Data Table Back-End for 'dplyr'

**Version**  1.0.1

**Description**  Provides a data.table backend for 'dplyr'. The goal of ‘dtplyr’ is to allow you to write 'dplyr' code that is automatically translated to the equivalent, but usually much faster, data.table code.

**License**  GPL-3

**URL**  https://github.com/tidyverse/dtplyr

**BugReports**  https://github.com/tidyverse/dtplyr/issues

**Depends**  R (>= 3.2)

**Imports**  crayon, data.table (>= 1.12.4), dplyr (>= 0.8.1), rlang, tibble, tidyselect

**Suggests**  bench, covr, knitr, rmarkdown, testthat (>= 2.1.0)

**VignetteBuilder**  knitr

**Encoding**  UTF-8

**LazyData**  true

**RoxygenNote**  7.0.2

**NeedsCompilation**  no

**Author**  Lionel Henry [aut], Hadley Wickham [cre], RStudio [cph]

**Maintainer**  Hadley Wickham <hadley@rstudio.com>

**Repository**  CRAN

**Date/Publication**  2020-01-23 06:00:03 UTC

---

**R topics documented:**

- dtplyr-package .................................................................................. 2
- collect.dtplyr_step ............................................................................ 2
- group_modify.dtplyr_step .................................................................... 3
- lazy_dt ................................................................................................. 4
- single_table ......................................................................................... 5
**Description**

Provides a data.table backend for 'dplyr'. The goal of 'dplyr' is to allow you to write 'dplyr' code that is automatically translated to the equivalent, but usually much faster, data.table code.

**Author(s)**

Maintainer: Hadley Wickham <hadley@rstudio.com>

Authors:

- Lionel Henry <lionel@rstudio.com>

Other contributors:

- RStudio [copyright holder]

**See Also**

Useful links:

- [https://github.com/tidyverse/dplyr](https://github.com/tidyverse/dplyr)
- Report bugs at [https://github.com/tidyverse/dplyr/issues](https://github.com/tidyverse/dplyr/issues)

**collect.dtplyr_step**

*Force computation of a lazy data.table*

**Description**

- `collect()` returns a tibble, grouped if needed
- `compute()` returns a new `lazy_dt`
- `as.data.table()` returns a data.table
- `as.data.frame()` returns a data frame
- `as_tibble()` returns a tibble
Usage

```r
## S3 method for class 'dtplyr_step'
collect(x, ...)

## S3 method for class 'dt plyr_step'
compute(x, ...)

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

## S3 method for class 'dtplyr_step'
as.data.frame(x, ...)

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

Arguments

- `x` : A `lazy_dt`
- `...` : Arguments used by other methods.
- `keep.rownames` : Ignored as `dplyr` never preserves rownames.

Description

`group_modify()` applies `.f` to each group, returning a modified `lazy_dt()`. This function is a little less flexible than the data.frame method due to the constraints of the code generation that `dplyr` uses.

Usage

```r
## S3 method for class 'dtplyr_step'
group_modify(.tbl, .f, ..., keep = FALSE)
```

Arguments

- `.tbl` : A `lazy_dt`
- `.f` : The name of a two argument function. The first argument is passed `.SD`, the data.table representing the current group; the second argument is passed `.BY`, a list giving the current values of the grouping variables. The function should return a list or data.table.
- `...` : Additional arguments passed to `.f`
- `keep` : Not supported for `lazy_dt`. 

---

**group_modify.dtplyr_step**

Modify a lazy_dt in place
Examples

```r
library(dplyr)

mtcars %>%
lazy_dt() %>%
group_by(cyl) %>%
group_modify(head, n = 2L)
```

### lazy_dt

Create a "lazy" data.table for use with dplyr verbs

**Description**

A lazy data.table lazy captures the intent of dplyr verbs, only actually performing computation when requested (with `collect()`, `pull()`, `as.data.frame()`, `data.table::as.data.table()`, or `tibble::as_tibble()`). This allows dtplyr to convert dplyr verbs into as few data.table expressions as possible, which leads to a high performance translation.

See vignette("translation") for the details of the translation.

**Usage**

```r
lazy_dt(x, name = NULL, immutable = TRUE, key_by = NULL)
```

**Arguments**

- **x**: A data table (or something can can be coerced to a data table).
- **name**: Optionally, supply a name to be used in generated expressions. For expert use only.
- **immutable**: If TRUE, x is treated as immutable and will never be modified by any code generated by dtplyr. Alternatively, you can set `immutable = FALSE` to allow dtplyr to modify the input object.
- **key_by**: Set keys for data frame, using `select()` semantics (e.g. `key_by = c(key1,key2)`). This uses `data.table::setkey()` to sort the table and build an index. This will considerably improve performance for subsets, summaries, and joins that use the keys.

See vignette("datatable-keys-fast-subset") for more details.

**Examples**

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

mtcars2 <- lazy_dt(mtcars)
mtcars2

mtcars %>% select(mpg:cyl)
mtcars %>% select(x = mpg, y = cyl)
mtcars %>% filter(cyl == 4) %>% select(mpg)
```
mtcars2 %>% select(mpg, cyl) %>% filter(cyl == 4)
mtcars2 %>% mutate(cyl2 = cyl * 2, cyl4 = cyl2 * 2)
mtcars2 %>% transmute(cyl2 = cyl * 2, vs2 = vs * 2)
mtcars2 %>% filter(cyl == 8) %>% mutate(cyl2 = cyl * 2)

by_cyl <- mtcars2 %>% group_by(cyl)
by_cyl %>% summarise(mpg = mean(mpg))
by_cyl %>% mutate(mpg = mean(mpg))
by_cyl %>% filter(mpg < mean(mpg)) %>% summarise(hp = mean(hp))

## S3 method for class 'dtplyr_step'
group_by(.data, ..., add = FALSE, arrange = TRUE)

### Arguments

- `.data` A data.table
- `...` In `group_by()`, variables or computations to group by. In `ungroup()`, variables to remove from the grouping.
- `add` When `FALSE`, the default, `group_by()` will override existing groups. To add to the existing groups, use `.add = TRUE`. This argument was previously called `add`, but that prevented creating a new grouping variable called `add`, and conflicts with our naming conventions.
- `arrange` If `TRUE`, will automatically arrange the output of subsequent grouped operations by group. If `FALSE`, output order will be left unchanged. In the generated data.table code this switches between using the `keyby` (TRUE) and by (FALSE) arguments.

### Description

This documents differences between standard dplyr verbs and their data.table instantiation.
Index

as.data.frame(), 4
as.data.frame.dtplyr_step
collect.dtplyr_step, 2
as.data.table.dtplyr_step
collect.dtplyr_step, 2
as_tibble.dtplyr_step
collect.dtplyr_step, 2

collect(), 4
collect.dtplyr_step, 2
compute.dtplyr_step
collect.dtplyr_step, 2

data.table::as.data.table(), 4
data.table::setkey(), 4
dplyr (dplyr-package), 2
dplyr-package, 2

group_by.dtplyr_step (single_table), 5
group_modify.dtplyr_step, 3
grouped_dt (lazy_dt), 4

lazy_dt, 2, 3, 4

lazy_dt(), 3

pull(), 4

select(), 4

single_table, 5

tbl_dt (lazy_dt), 4
tibble::as_tibble(), 4