Package 'mlr3db'

October 29, 2019

Title Data Base Backend for 'mlr3'

Version 0.1.3

Description Extends the 'mlr3' package with a backend to transparently work with data bases. Internally relies on the abstraction of package 'dbplyr' to interact with one of the many supported data base management systems (DBMS).

License LGPL-3

URL https:///mlr3db.mlr-org.com, https://github.com/mlr-org/mlr3db

BugReports https://github.com/mlr-org/mlr3db/issues

Depends R (>= 3.1.0)

Imports checkmate, data.table, digest, dplyr, mlr3 (>= 0.1.4), R6

Suggests DBI, dbplyr, lgr, RSQLite, testthat, tibble

Encoding UTF-8

RoxygenNote 6.1.1

NeedsCompilation no

Author Michel Lang [cre, aut] (<https://orcid.org/0000-0001-9754-0393>)

Maintainer Michel Lang <michellang@gmail.com>

Repository CRAN

Date/Publication 2019-10-29 17:10:02 UTC

R topics documented:

mlr3db-package 2
as_sqlite_backend 2
DataBackendDplyr 3

Index **5**

mlr3db-package *mlr3db: Data Base Backend for 'mlr3'*

Description

Extends the 'mlr3' package with a backend to transparently work with data bases. Internally relies on the abstraction of package 'dbplyr' to interact with one of the many supported data base management systems (DBMS).

Author(s)

Maintainer: Michel Lang <michellang@gmail.com> (0000-0001-9754-0393)

See Also

Useful links:

- https:///mlr3db.mlr-org.com
- https://github.com/mlr-org/mlr3db
- Report bugs at https://github.com/mlr-org/mlr3db/issues

as_sqlite_backend *Convert to use a SQLite Backend*

Description

Converts to a DataBackendDplyr using a **RSQLite** data base, depending on the input type:

- `data.frame`: Converts to a DataBackendDplyr.
- `[mlr3::DataBackend]`: Creates a new DataBackendDplyr using the data of the provided mlr3::DataBackend.
- `[mlr3::Task]`: Replaces the DataBackend in slot `$task` with a new backend. Only active columns and rows are considered.

Usage

```
as_sqlite_backend(data, path = NULL, ...)
```

Arguments

`data`	(`data.frame()` | mlr3::DataBackend | mlr3::Task) See description.
`path`	(`NULL` | `character(1)`) Path for the SQLite data base. Defaults to a file in the temporary directory of the R session, see `tempfile()`.
`...`	(any) Additional arguments, currently ignored.

Value

DataBackendDplyr.

`DataBackendDplyr` *DataBackend for dplyr/dbplyr*

Description

A mlr3::DataBackend using `dplyr::tbl()` from packages **dplyr/dbplyr**. This includes `tibbles`. Allows to let a mlr3::Task interface an out-of-memory data base.

Format

R6::R6Class object inheriting from mlr3::DataBackend.

Construction

```
DataBackendDplyr$new(data, primary_key = NULL, strings_as_factors = TRUE)
```

- `data` :: `dplyr::tbl()`
  The data object.
- `primary_key` :: `character(1)`
  Name of the primary key column.
- strings_as_factors :: `logical(1)` || `character()`
  Either a character vector of column names to convert to factors, or a single logical flag: if `FALSE`, no column will be converted, if `TRUE` all string columns (except the primary key). The backend is queried for distinct values of the respective columns and their levels are stored in `$levels`.

Alternatively, use `mlr3::as_data_backend()` on a `dplyr::tbl()` which will construct a DataBackend for you.

Fields

All fields from mlr3::DataBackend, and additionally:

- `levels` :: named `list()`
  List of factor levels, named with column names. The columns get automatically converted to factors in `$data()` and `head()`.

Methods

All methods from mlr3::DataBackend.

Examples

```
# Backend using a in-memory tibble
data = tibble::as_tibble(iris)
data$Sepal.Length[1:30] = NA
data$row_id = 1:150
b = DataBackendDplyr$new(data, primary_key = "row_id")

# Object supports all accessors of DataBackend
print(b)
b$nrow
b$ncol
b$colnames
b$data(rows = 100:101, cols = "Species")
b$distinct(b$rownames, "Species")

# Classification task using this backend
task = mlr3::TaskClassif$new(id = "iris_tibble", backend = b, target = "Species")
print(task)
task$head()

# Create a temporary SQLite data base
con = DBI::dbConnect(RSQLite::SQLite(), ":memory:")
dplyr::copy_to(con, data)
tbl = dplyr::tbl(con, "data")

# Define a backend on a subset of the data base
tbl = dplyr::select_at(tbl, setdiff(colnames(tbl), "Sepal.Width")) # do not use column "Sepal.Width"
tbl = dplyr::filter(tbl, row_id %in% 1:120) # Use only first 120 rows
b = DataBackendDplyr$new(tbl, primary_key = "row_id")
print(b)

# Query disinct values
b$distinct(b$rownames, "Species")

# Query number of missing values
b$missings(b$rownames, b$colnames)

# Note that SQLite does not support factors, column Species has been converted to character
lapply(b$head(), class)

# Cleanup
rm(tbl)
DBI::dbDisconnect(con)
```

Index

∗Topic **datasets**
DataBackendDplyr, 3

as_sqlite_backend, 2

DataBackend, *2, 3*
DataBackendDplyr, *2, 3*, 3
dplyr::tbl(), *3*

mlr3::as_data_backend(), *3*
mlr3::DataBackend, *2, 3*
mlr3::Task, *2, 3*
mlr3db *(*mlr3db-package*)*, 2
mlr3db-package, 2

R6::R6Class, *3*

tempfile(), *2*
tibbles, *3*