Package: certedb (via r-universe)

September 12, 2024

Title A Certe R Package for Connecting to Databases

Version 1.12.1

Description A Certe R Package for connecting to internal Certe cBases and local DuckDB databases. This package is part of the 'certedata' universe.

URL https://certe-medical-epidemiology.github.io/certedb, https://github.com/certe-medical-epidemiology/certedb

Depends R (>= 4.1.0)

Imports certestyle, certetoolbox, AMR (>= 2.0.0), cli (>= 3.4.0), DBI (>= 1.0.0), dbplyr (>= 2.1.0), dplyr (>= 1.0.0), DT (>= 0.3.0), duckdb (>= 0.8.0), knitr (>= 1.40), odbc (>= 1.3.0), pillar (>= 1.8.0), shiny (>= 1.9.0), rlang (>= 1.1.0), tidyr (>= 1.1.0), yaml (>= 2.2.1)

Suggests callr, clipr (>= 0.8.0), RMariaDB, rstudioapi, testthat (>= 2.0.0), utils

License GPL-2

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/testthat/edition 2

Repository https://certe-medical-epidemiology.r-universe.dev

RemoteUrl https://github.com/certe-medical-epidemiology/certedb

RemoteRef HEAD

RemoteSha 999086d22d807e65f7e6664769bc2174651f1197

Contents

---

db_connect *Connect to a Certe Database*

Description

Connect to an internal (yet remote) Certe database server using `DBI::dbConnect()`, or any other database.

Usage

```
db_connect(driver, ..., print = TRUE)

db_close(conn, ..., print = TRUE)
```

Arguments

`driver`	database driver to use, such as `odbc::odbc()` and `duckdb::duckdb()`
`...`	arguments passed on to `DBI::dbDisconnect()`
`print`	a logical to indicate whether info about the connection should be printed
`conn`	connection to close, such as the output of `db_connect()`

Examples

```
# create a local duckdb database
db <- db_connect(duckdb::duckdb(), "~/my_duck.db")
db

db |> db_write_table("my_iris_table", values = iris)
db |> db_list_tables()
db |> db_has_table("my_iris_table")

if (require(dplyr, warn.conflicts = FALSE)) {
  db |>
    tbl("my_iris_table") |>
    filter(Species == "setosa", Sepal.Width > 3) |>
    collect() |>
    as_tibble()
}

db |> db_drop_table("my_iris_table")
db |> db_list_tables()

db |> db_close()

# remove the database
unlink("~/my_duck.db")
```

get_diver_data *Download Data from a Local or Remote Database*

Description

Thes functions can be used to download local or remote database data, e.g. Spectre data from DiveLine on a Diver server (from Dimensional Insight). The `get_diver_data()` function sets up an ODBC connection (using `db_connect()`), which requires their quite limited DI-ODBC driver.

Usage

```
get_diver_data(
  date_range = this_year(),
  where = NULL,
  review_qry = interactive(),
  antibiogram_type = "sir",
  distinct = TRUE,
  auto_transform = TRUE,
  snake_case = TRUE,
  preset = read_secret("db.preset_default"),
  date_column = NULL,
  diver_cbase = NULL,
  diver_project = read_secret("db.diver_project"),
  diver_dsn = if (diver_testserver == FALSE) read_secret("db.diver_dsn") else
    read_secret("db.diver_dsn_test"),
  diver_testserver = FALSE,
  diver_tablename = "data",
  info = interactive(),
  limit = Inf,
  in_background = FALSE,
  ...
)

certedb_query(query, where = NULL, auto_transform = TRUE, info = interactive())

get_duckdb_data(
  date_range = this_year(),
  where = NULL,
  preset = NULL,
  review_qry = interactive(),
  duckdb_path = read_secret("db.duckdb_path"),
  duckdb_table = read_secret("db.duckdb_table"),
  auto_transform = TRUE,
  info = interactive()
)

certedb_getmmb(
```

```
  dates = NULL,
  where = NULL,
  select_preset = "mmb",
  preset = "mmb",
  select = NULL,
  add_cols = NULL,
  info = interactive(),
  first_isolates = FALSE,
  eucast_rules = "all",
  keep_synonyms = getOption("AMR_keep_synonyms", FALSE),
  mic = FALSE,
  disk = FALSE,
  zipcodes = FALSE,
  ziplength = 4,
  tat_hours = FALSE,
  only_real_patients = TRUE,
  only_conducted_tests = TRUE,
  only_validated = FALSE,
  only_show_query = FALSE,
  review_where = interactive(),
  auto_transform = TRUE,
  query = NULL,
  ...
)

certedb_getmmb_tat(
  dates = NULL,
  where = NULL,
  add_cols = NULL,
  info = interactive(),
  only_real_patients = TRUE,
  only_conducted_tests = TRUE,
  only_validated = TRUE,
  only_show_query = FALSE,
  ...
)

get_glims10_data(
  date_range = this_year(),
  where = NULL,
  review_qry = interactive(),
  antibiogram_type = "sir",
  distinct = TRUE,
  auto_transform = TRUE,
  diver_tablename = "glims10",
  info = interactive(),
  limit = Inf,
  ...
```

)

Arguments

date_range	date range, can be length 1 or 2 (or more to use the min/max) to filter on the column specified in the YAML file, see presets. Defaults to this_year(). Use NULL to set no date filter. Can also be years, or functions such as last_month(). Date-time ojects will be converted to dates, so using times as input is useless. It is supported to filter on a date-time column though.
where	arguments to filter data on, will be passed on to filter(). **Do not use** && **or** || **but only** & **or** | **in filtering.**
review_qry	a logical to indicate whether the query must be reviewed first, defaults to TRUE in interactive mode and FALSE otherwise. This will always be FALSE in Quarto / R Markdown, since the output of knitr::pandoc_to() must be NULL.
antibiogram_type	antibiotic transformation mode. Leave blank to strip antibiotic results from the data, "sir" to keep SIR values, "mic" to keep MIC values or "disk" to keep disk diffusion values. Values will be cleaned with as.sir(), as.mic() or as.disk().
distinct	logical to apply distinct() to the resulting data set
auto_transform	logical to apply auto_transform() to the resulting data set
snake_case	logical to convert column names to snake case, **only** when auto_transform = TRUE
preset	a preset to choose from presets(). Will be ignored if diver_cbase is set, even if it is set to NULL.
date_column	column name of data set to query. Normally this should be set in a preset, but this argument can be used to override that.
diver_cbase, diver_project, diver_dsn, diver_testserver	properties to set in db_connect(). The diver_cbase argument will be based on preset, but can also be set to blank NULL to manually select a cBase in a popup window.
diver_tablename	name of the database table to download data from. This is hard-coded by DI and should normally never be changed.
info	settings for old certedb_getmmb() function
limit	maximum number of rows to return.
in_background	run data collection in the background using callr::r_bg(). Use ...$get_result() to retrieve results, or ...$is_active() to check whether the background process still runs.
...	not used anymore, previously settings for old certetools::certedb_getmmb() function
query	a data.frame to view the query of, or a character string to run as query in certedb_getmmb() (which will ignore all other arguments, except for where, auto_transform and info).

duckdb_path	path to the local DuckDB database
duckdb_table	name of the DuckDB database to retrieve data from
dates	date range, can be length 1 or 2 (or more to use the min/max) to filter on the column Ontvangstdatum. Defaults to this_year(). Use NULL to set no date filter. Can also be years, or functions such as last_month().
select_preset	settings for old certedb_getmmb() function
select	settings for old certedb_getmmb() function
add_cols	settings for old certedb_getmmb() function
first_isolates	settings for old certedb_getmmb() function
eucast_rules	settings for old certedb_getmmb() function
keep_synonyms	a logical to indicate if old, previously valid taxonomic names must be preserved and not be corrected to currently accepted names. The default is FALSE, which will return a note if old taxonomic names were processed. The default can be set with the package option AMR_keep_synonyms, i.e. options(AMR_keep_synonyms = TRUE) or options(AMR_keep_synonyms = FALSE).
mic	settings for old certedb_getmmb() function
disk	settings for old certedb_getmmb() function
zipcodes	settings for old certedb_getmmb() function
ziplength	settings for old certedb_getmmb() function
tat_hours	settings for old certedb_getmmb() function
only_real_patients	settings for old certedb_getmmb() function
only_conducted_tests	settings for old certedb_getmmb() function
only_validated	settings for old certedb_getmmb() function
only_show_query	settings for old certedb_getmmb() function
review_where	a logical to indicate whether the query must be reviewed first, defaults to TRUE in interactive mode and FALSE otherwise. This will always be FALSE in Quarto / R Markdown, since the output of knitr::pandoc_to() must be NULL.

Details

These functions return a 'certedb tibble' from Diver or the certemmb MySQL database, which prints information in the tibble header about the used source and current user.

Use certedb_query() to retrieve the original query that was used to download the data.

Using certedb_query("your qry here") is identical to using certedb_getmmb(query = "your qry here").

Examples

```
## Not run:

# peek-preview of a cBase:
get_diver_data(diver_cbase = "models/MedEpi/GLIMS10_Resistenties.cbase",
               date_range = NULL, limit = 10, review_qry = FALSE, auto_transform = FALSE)

# these two work identical:
get_diver_data(date_range = 2024, where = BepalingCode == "PXNCOV")
get_diver_data(2024, BepalingCode == "PXNCOV")

# use di$ to pull a list with column names while you type
get_diver_data(2024, di$BepalingCode == "PXNCOV")

# for the `where`, use `&` or `|`:
get_diver_data(last_month(),
               di$BepalingCode == "PXNCOV" & di$Zorglijn == "2e lijn")
get_diver_data(c(2020:2024),
               where = di$BepalingCode == "PXNCOV" | di$Zorglijn == "2e lijn")


# use %like%, %unlike%, %like_case% or %unlike_case% for regular expressions
get_diver_data(2024, where = di$MateriaalNaam %like% "Bloed")
get_diver_data(2024, where = di$MateriaalNaam %unlike% "Bloed")
get_diver_data(2024, where = di$MateriaalNaam %like_case% "bloed")
get_diver_data(2024, where = di$MateriaalNaam %unlike_case% "Bloed")

get_diver_data(2024,
               where = di$BepalingNaam %like% "Noro" &
                         di$PatientLeeftijd >= 75)

# R objects will be converted
materialen <- c("A", "B", "C")
get_diver_data(2024, where = di$MateriaalNaam %in% materialen)
leeftijden <- 65:85
get_diver_data(2024, where = di$PatientLeeftijd %in% leeftijden)


# USING DIVER INTEGRATOR LANGUAGE --------------------------------------

# See the website for an overview of allowed functions:
# https://www.dimins.com/online-help/workbench_help/Content/ODBC/di-odbc-sql-reference.html

# Use Diver Integrator functions within EVAL():
get_diver_data(2024, where = EVAL('regexp(value("MateriaalCode"),"^B")'))

get_diver_data(
  2024,
  where = EVAL('rolling(12, value("OntvangstDatum"), date("2024/11/27"))')
)
```

```
## End(Not run)


# USING DUCKDB DATABASE ----------------------------------------------

# create a local duckdb database and write a table
db <- db_connect(duckdb::duckdb(), "~/my_duck.db")
db |> db_write_table("iris", values = iris)
db |> db_close()

df <- get_duckdb_data(date_range = NULL,
                      where = Species == "setosa" & Sepal.Width > 3,
                      # not needed in production environment:
                      duckdb_path = "~/my_duck.db",
                      duckdb_table = "iris",
                      review_qry = FALSE,
                      info = TRUE)
df
certedb_query(df)

# remove the database
unlink("~/my_duck.db")
```

`presets` *Available Presets for* `get_diver_data()`

Description

Work with presets for `get_diver_data()`. This automates selecting, filtering, and joining cBases.

Usage

```
presets()

get_preset(preset)
```

Arguments

`preset`	name of the preset

Details

The function `presets()` returns a data.frame with available presets, as defined in the secrets YAML file under `db.presets`.

The function `get_preset()` will return all the details of a preset as a list.

Required YAML Format

This YAML information should be put into the YAML file that is read using `read_secret()`. Afterwards, the name of the preset can be used as `get_diver_data(preset = "...")`.

The most basic YAML form:

```
name_new_preset:
  cbase: "location/to/name.cbase"
  date_col: "ColumnNameDate"
```

The most extensive YAML form:

```
name_new_preset:
  cbase: "location/to/name.cbase"
  date_col: "ColumnNameDate"
 filter: ColumnName1 %in% c("Filter1", "Filter2") & ColumnName2 %in% c("Filter3", "Filter4")
  select: ColumnName1, ColumnName2, ColumnNameReceiptDate, new_name = OldName
  join:
    cbase: "location/to/another.cbase"
    by: ColumnName1, ColumnName2
    type: "left"
   select: ColumnName1, ColumnName2, ColumnName3, everything(), !starts_with("abc")
    wide_names_from: ColumnName3
  join2:
    cbase: "location/to/yet_another.cbase"
    by: ColumnName1, ColumnName2
    type: "left"
   select: ColumnName1, ColumnName2, ColumnName3, everything(), !where(is.numeric)
    wide_names_from: ColumnName3
    wide_name_trans: gsub("_", "..", .x)
```

For all presets, `cbase` and `date_col` are required.

Input for `select` will be passed on to `select()`, meaning that column names can be used, but also `tidyselect` functions such as `everything()`. Input for `filter` will be passed on to `filter()`.

Joins:

For joins, you must set at least `cbase`, `by`, and `type` ("left", "right", "inner", etc., see here).

An unlimited number of joins can be used, but all so-called 'keys' must be unique and start with `join`, e.g. `joinA` / `joinB` or `join` / `join2`.

If `wide_names_from` is set, the dataset is first transformed to long format using the columns specified in `by`, and then reshaped to wide format with values in `wide_names_from`.

Use `wide_name_trans` to transform the values in `wide_names_from` before the reshaping to wide format is applied. Use `.x` for the column values. As this will be applied before the data is transformed to a wide format, it allows to refine the values in `wide_names_from` using e.g., `case_when()`.

shiny_explore *Search cBase Interactively*

Description

Use this Shiny app to search a cBase. There is also an RStudio add-in.

Usage

```
shiny_explore(
  preset = read_secret("db.preset_default_shiny"),
  diver_cbase = NULL,
  diver_project = read_secret("db.diver_project"),
  diver_dsn = if (diver_testserver == FALSE) read_secret("db.diver_dsn") else
    read_secret("db.diver_dsn_test"),
  diver_testserver = FALSE,
  diver_tablename = "data"
)
```

Arguments

preset	a preset to choose from presets(). Will be ignored if diver_cbase is set, even if it is set to NULL.
diver_cbase, diver_project, diver_dsn, diver_testserver	properties to set in db_connect(). The diver_cbase argument will be based on preset, but can also be set to blank NULL to manually select a cBase in a popup window.
diver_tablename	name of the database table to download data from. This is hard-coded by DI and should normally never be changed.

Index