Title  Utilities for 'DIZ' R Package Development
Version  0.1.2
Date  2023-01-23
Description  Utility functions used for the R package development infrastructure inside the data integration centers (DIZ) to standardize and facilitate repetitive tasks such as setting up a database connection or issuing notification messages and to avoid redundancy.
License  GPL-3
URL  https://github.com/miracum/misc-dizutils
BugReports  https://github.com/miracum/misc-dizutils/issues
Depends  R (>= 3.1.0)
Imports  data.table, DBI (>= 1.1.0), DIZtools, Hmisc, psych, RJDBC, RJSONIO, RPostgres, xml2
Suggests  lintr, testthat
Encoding  UTF-8
Language  en-US
SystemRequirements  libpq >= 9.0: libpq-dev (deb) or postgresql-devel (rpm)
RoxygenNote  7.2.3
NeedsCompilation  no
Author  Jonathan M. Mang [aut, cre] (<https://orcid.org/0000-0003-0518-4710>), Lorenz A. Kapsner [aut] (<https://orcid.org/0000-0003-1866-860X>), MIRACUM - Medical Informatics in Research and Care in University Medicine [fnd], Universitätsklinikum Erlangen, Germany [cph]
Maintainer  Jonathan M. Mang <jonathan.mang@uk-erlangen.de>
Repository  CRAN
Date/Publication  2023-01-23 10:10:11 UTC
R topics documented:

- check_if_table_exists .................................................. 2
- close_connection ......................................................... 3
- combine_stats ......................................................... 3
- db_connection .......................................................... 4
- get_config_env ......................................................... 5
- get_db_systems ......................................................... 7
- get_default_config_elements ................................. 7
- query_database .......................................................... 8
- xml_2_json ............................................................... 9

Index

check_if_table_exists  Check if a database table exists.

Description

Check if a database table exists.

Usage

check_if_table_exists(db_con, table_name)

Arguments

- db_con A DBI database connection. See 'db_connection()' for details.
- table_name (String) The name of the table or view to be checked.

Value

True, if the table exists, false otherwise.

Examples

```r
## Not run:
res <- DIZutils::check_if_table_exists(
  db_con = DIZutils::db_connection(...),
  table_name = "my_table"
)
## End(Not run)
```
close_connection  close_connection helper function

Description

Internal function to close the database connection. The function is just a wrapper around ‘RPostgres::dbDisconnect’

Usage

close_connection(conn)

Arguments

conn  A DBI database connection.

Value

The function is just a wrapper around RPostgres::dbDisconnect / DBI::dbDisconnect and does not return any value.

Examples

## Not run:
db_con <- DIZutils::db_connection(
  db_name = "i2b2",
  db_type = "postgres"
)

DIZutils::close_connection(
  conn = db_con
)

## End(Not run)

combine_stats  Combine aggregated statistics.

Description

This function provides the functionality to combine multiple statistics to a single statistical overview. This is e.g. useful if you are only allowed to export statistical characteristics from a site but not the data itself. So in this case you have e.g. mean, median and N from each site but want to say something about the mean, median and N over all sites like you had the data of all sites in one big pool and would do the statistics there.
Usage

combine_stats(summaries, demo = FALSE)

Arguments

- summaries: (data.table) Data table containing all stats you want to combine as rows. This data.table must contain the columns ‘Min’, ‘Q10’, ‘Q25’, ‘Median’, ‘Mean’, ‘SD’, ‘Q75’, ‘Q90’, ‘Max’, ‘N’. Each row in this data table represents a site as of the example described above.
- demo: (boolean, default = FALSE) Do you want to see how the function works? Then call ‘combine_stats(summaries = NULL, demo = TRUE)’.

Value

A one-row data.table containing the calculated, aggregates statistics of the input.

---

**db_connection**

**db_connection helper function**

Description

Internal function to test and get the database connection of the target data system.

Usage

```
db_connection(
  system_name = NULL,
  db_type,
  headless = TRUE,
  from_env = TRUE,
  settings = NULL,
  timeout = 30,
  logfile_dir = NULL,
  lib_path = NULL
)
```

Arguments

- system_name: (Default = NULL) A character. Name of the database system. Used to find the correct settings from the env. If you don’t want to load the settings from the environment, use the ‘settings’ parameter. Otherwise this function will search for all settings beginning with ‘system_name’ in the environment. If ‘system_name = "i2b2"’ settings like ‘I2B2_HOST’ or ‘I2B2_PORT’ (notice the uppercase) will be loaded from the environment. You can load such an env file e.g. by using ‘DIZtools::setenv_file(path_to_file)’.
- db_type: A character. Type of the database system. Currently implemented systems are: ‘postgres’, ‘oracle’.
get_config_env

headless
A boolean (default: ‘FALSE’). Indicating, if the function is run only in the console (‘headless = TRUE’) or on a GUI frontend (‘headless = FALSE’).

from_env
A boolean (default: ‘TRUE’). Should database connection be read from the environment or from a settings file. All necessary parameters must be uppercase and have the prefix of the db_name. E.g.: ‘I2B2_HOST’ or ‘I2B2_PORT’. See the ‘settings’ parameter for all necessary variables.

settings
A list. Required if ‘from_env == FALSE’. A list containing settings for the database connection. Required fields are ‘host’, ‘db_name’, ‘port’, ‘user’ and ‘password’. Additionally for Oracle DB’s: ‘sid’ (instead of ‘db_name’). If ‘settings’ is set, ‘from_env’ will be set to ‘FALSE’ automatically.

timeout
A timeout in sec. for the db-connection establishment. Values below 2 seconds are not recommended. Default is 30 seconds.

logfile_dir
(Optional, String, default: "tempdir()") The absolute path to folder where the logfile will be stored.

lib_path
A character string. The path to the ojdbc*.jar file. If you run one of the R-containers from the UK-Erlangen DIZ, there might be a lib for oracle here: ‘lib_path = "/opt/libs/ojdbc8.jar"’

Value
If successful, the result will be the established connection. Otherwise the result will be null.

See Also

dbConnect, RPostgres

Examples

## Not run:
db_con <- DIZutils::db_connection(
  db_name = "i2b2",
  db_type = "postgres"
)
## End(Not run)
Usage

get_config_env(
    system_name,
    logfile_dir = tempdir(),
    headless = TRUE,
    ignore_presets = FALSE,
    uppercase_system = TRUE
)

Arguments

system_name  The name of the system (This is also the prefix used to get the environment variables with ‘SYSTEM_KEY’, e.g. ‘I2B2_DBNAME’). This function also works if there are multiple instances like ‘I2B2_1_DBNAME’ and ‘I2B2_2_DBNAME’. Then the result will contain nested lists for each occurrence.

logfile_dir  (Optional, String, default: “tempdir()”) The absolute path to folder where the logfile will be stored.

headless  A boolean (default: ‘FALSE’). Indicating, if the function is run only in the console (‘headless = TRUE’) or on a GUI frontend (‘headless = FALSE’).

ignore_presets  (boolean, default = FALSE) Only return something if all elements from the presets are found? These are currently ‘host’, ‘port’, ‘user’, ‘password’, ‘sid’, ‘path’. If you have another suffix after ‘system_name’ in your config file, you won’t see it here. To see everything with prefix ‘system_name’ simply set ‘ignore_presets = TRUE’. To obtain a list of current default elements, run DIZutils:::get_default_config_elements().


Value

If successful it returns the config, null otherwise.

Examples

get_config_env(
    system_name = "i2b2",
    logfile_dir = tempdir(),
    headless = FALSE
)
get_db_systems

Quickly get all currently implemented database systems

Description
Function to quickly get the currently implemented database systems

Usage
get_db_systems()

Value
The currently implemented database systems as string array. ‘E.g. c(“postgres”, “oracle”)’

Examples
get_db_systems()
# Result: c(“postgres”, “oracle”)

get_default_config_elements

Get default element names (suffixes) for env reading

Description
Internal function. Get default element names (suffixes) for env reading

Usage
get_default_config_elements()

Value
Vector with default elements
query_database

query_database helper function

Description

Internal function to query the database. The function sends a sql statement to the database and returns a data.table.

Usage

query_database(
  db_con,
  sql_statement,
  no_result = FALSE,
  close_connection = FALSE
)

Arguments

- db_con: A DBI database connection.
- sql_statement: A character string containing a valid SQL statement. Caution: Everything after the first ';' will be cut off.
- no_result: (boolean, default: FALSE) Is the sql meant to return nothing? E.g. if you just insert or update a table. Then supply ‘TRUE’ here. If you supply ‘FALSE’ here, the function expects to receive a result table and tries to convert it to a data.table.
- close_connection: (boolean, default = FALSE). If TRUE, the connection will be closed after the query was sent and the result received.

Value

Returns the result of the db-query. If ‘no_result’ is ‘TRUE’, the return value will be ‘TRUE’ if the query was successfully sent. Otherwise (if ‘no_result’ is ‘FALSE’ which is the default), the result will be the result of the sql query as data.table.

Examples

```r
## Not run:
db_con <- DIZutils::db_connection(
  db_name = "i2b2",
  db_type = "postgres"
)

query_database(
  db_con = db_con,
  sql_statement = "SELECT * FROM table_name;"
)
```
xml_2_json

query_database(
    db_con = db_con,
    sql_statement = "INSERT INTO table_name DEFAULT VALUES;",
    no_result = TRUE
)

## End(Not run)

---

**xml_2_json**

*Quickly transform a xml object into a json object.*

**Description**

See title.

**Usage**

xml_2_json(xml)

**Arguments**

- **xml**: An xml object.

**Value**

The json-representation of the xml object.
Index

check_if_table_exists, 2
close_connection, 3
combine_stats, 3
db_connection, 4
dbConnect, 5
get_config_env, 5
get_db_systems, 7
get_default_config_elements, 7
query_database, 8
RPostgres, 5
xml_2_json, 9