CheatSheet - Connecting to Databases from R

Topic	Syntax	Description	Example
RJDBC			
library()	`library(packageName)`	library() function loads the R package	`library(RJDBC)` `library(RODBC)`

Initialize Database connection parameters RJDBC	`dsn_driver = "driver_name"`
	`dsn_database = "database_name"`
	`dsn_hostname = "your hostname"`
	`dsn_port = "port"`
	`dsn_protocol = "TCPIP"`
	`dsn_uid = "your uid"`
	`dsn_pwd = "your password"`
		Initializes and set the connection parameters which are required to connect to the database	`dsn_driver = "com.ibm.db2.jcc.DB2Driver"`
			`dsn_database = "BLUDB"`
			`dsn_hostname = "dashdb-txn-sbox-yp-lon02-01.services.eu-gb.bluemix.net"`
			`dsn_port = "50000"`
			`dsn_protocol = "TCPIP"`
			`dsn_uid = "qwp24135"`
			`dsn_pwd = "xhhq+4bq4bsx018k"`

RJDBC Connection	`JDBC (driverClass = "", classPath = "", identifier.quote = NA)`	`JDBC()` function creates a new DBI driver that can be used to start JDBC connections. **identifier.quote** is the character to use for automatically generated SQL statements or defaults to NA if the back-end doesn't support quoted identifiers.	`jcc <- JDBC(driverClass="com.ibm.db2.jcc.DB2Driver", classPath = "/home/jupyterlab/.rlang/db2jcc-db2jcc4.jar");`
Connect to the RJDBC database	`dbConnect(driver, host, user="username", password="pwd",...)`	`dbConnect()` Connect to a DBMS going through the appropriate authorization procedure. Some implementations may allow you to have multiple connections open, so you may invoke this function repeatedly assigning its output to different objects. The authorization mechanism depends onthe drivers used. '...' indicated the other parameters passed to the method.	`conn <- dbConnect(jcc, "jdbc:db2://hostname:32733/bludb", user="qdt91544", password="b1fghNO6d23ZgN38", security="ssl")`
dbSendQuery()	`dbSendQuery(conn, statement, ...)`	`dbSendQuery` function just submits and synchronously executes the SQL query to the database engine. It does not extract any records. **...** indicates other named parameters passed on to methods.	`query = "SELECT * FROM SYSIBM.SYSSCHEMATA";` `rs = dbSendQuery(conn, query);`
fetch()	`fetch(res, n, ...)`	`fetch()` function records from a previously executed query	`df = fetch(rs, -1);`
dbDisconnect()	`dbDisconnect(conn, ...)`	`dbDisconnect()` closes the connection, discards all pending work, and frees resources (e.g., memory, sockets). **...** indicates other named parameters passed on to methods.	`dbDisconnect(conn)`

RODBC			

Initialize Database connection parameters RJDBC	`dsn_driver = "driver_name"`
	`dsn_database = "database_name"`
	`dsn_hostname = "your hostname"`
	`dsn_port = "port"`
	`dsn_protocol = "TCPIP"`
	`dsn_uid = "your uid"`
	`dsn_pwd = "your password"`
		Initializes and set the connection parameters which are required to connect to the database	`dsn_driver <- "{IBM DB2 ODBC Driver}"`
			`dsn_database <- "BLUDB" # e.g. "BLUDB"`
			`dsn_hostname <- "dashdb-txn-sbox-yp-lon02-01.services. eu-gb.bluemix.net" # e.g.: "dashdb-txn-sbox-yp-dal09-11.services.dal.bluemix.net"`
			`dsn_port <- "50000" # e.g. "50000"`
			`dsn_protocol <- "TCPIP" # i.e. "TCPIP"`
			`dsn_uid <- "qwp24135" # e.g. "zjh17769"`
			`dsn_pwd <- "xhhq+4bq4bsx018k"`

Connect to the RODBC database	`paste (…, sep = " ", collapse = NULL)`	`paste()` function concatenate vectors after converting to character.	`conn_path <- paste("DRIVER=", dsn_driver, ";DATABASE=", dsn_database, ";HOSTNAME=", dsn_hostname, ";PORT=", dsn_port, ";PROTOCOL=", dsn_protocol, ";UID=", dsn_uid, ";PWD=", dsn_pwd, sep="")`
odbcDriverConnect()	`odbcDriverConnect(conn_path)`	`odbcDriverConnect()`function takes the connection parameters as a string containing name value pairs. It returns a connection object.	`conn <- odbcDriverConnect(conn_path)`
attributes()	`attributes(x) where x: object whose attributes to be accessed.`	`attribute()` function is used to get all the attributes of data. This function is also used to set new attributes to data.	`attributes(conn)`
sqlColumns()	`sqlColumns(channel, sqtable, errors = FALSE, as.is = TRUE, special = FALSE, catalog = NULL, schema = NULL, literal = FALSE)`	`sqlColumns()` function enquire about the column structure of tables on an ODBC database connection.	`col.detail <- sqlColumns(conn, "SYSSCHEMATA")`
odbcCloseAll()	`odbcCloseAll()`	`odbcCloseAll()` function closes connections to ODBC databases.	`odbcCloseAll()`

Author(s)

D.M Naidu s

Other Contributor(s)

Malika Singla

Changelog

Date	Version	Changed by	Change Description
2020-08-31	1.0	Malika Singla	Initial Version