

Package ‘dbProject’

June 2, 2026

Title Database Connection Management and Utilities for 'dbverse'

Version 0.1.1

Description Provides an R6-based project container for managing 'DuckDB' connections, reconnecting lazy database tables after session restarts, and storing metadata for database-backed objects used by packages in the 'dbverse'. The package supplies S4 base classes and generics for database-backed data, helpers for validating 'DuckDB' connections and table names, utilities for creating persistent database views, and methods for writing and restoring pinned lazy tables through the 'pins' package. These tools help package authors and analysts keep database paths, cached connections, and table references synchronized across interactive sessions and project directories.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

URL <https://github.com/dbverse-org/dbproject-r>,
<https://dbverse-org.github.io/dbproject-r/>

BugReports <https://github.com/dbverse-org/dbproject-r/issues>

Depends R (>= 4.1.0)

Imports dplyr, duckdb (>= 1.4.0), DBI, pins, dbplyr, methods, glue,
cli, R6, connections, yaml

Suggests knitr, rmarkdown, withr, testthat (>= 3.0.0), dbMatrix

VignetteBuilder knitr

Config/testthat/edition 3

Collate 'classes.R' 'table-reconnection.R' 'connection-registry.R'
'generics.R' 'connection-methods.R' 'dbList.R' 'dbMatrix-pin.R'
'dbProject-R6.R' 'dbSpatial-pin.R' 'extract.R' 'imports.R'
'input-validation.R' 'path-utils.R' 'to_view.R'
'unique_table_name.R'

NeedsCompilation no

Author Edward C. Ruiz [aut, cre] (ORCID:
<https://orcid.org/0000-0002-9174-5387>),
 Jiayi George Chen [aut],
 Ruben Dries [aut]

Maintainer Edward C. Ruiz <ecr7407@gmail.com>

Repository CRAN

Date/Publication 2026-06-02 16:40:02 UTC

Contents

connection_pin_read	2
dbData-class	3
dbData-connection-accessors	3
dbData-extract	4
dbList	5
dbList,DBIConnection-method	5
dbLoad	6
dbProject	6
dbReconnect	10
read_pin_conn	11
read_pin_conn.conn_matrix_table	11
read_pin_conn.conn_spatial_table	12
to_view	12
to_view,ANY-method	13
write_pin_conn	13
write_pin_conn.dbMatrix	14
write_pin_conn.dbSpatial	14
write_pin_conn.default	15
Index	16

connection_pin_read *Read a pinned dbMatrix object from a board*

Description

Read a pinned dbMatrix object from a board

Usage

```
connection_pin_read(board, name, version = NULL)
```

Arguments

board	A pins board object
name	The name of the pin
version	The version of the pin to get (optional)

Value

A dbMatrix object (dbSparseMatrix or dbDenseMatrix)

 dbData-class

Virtual Base Class for Database-Backed Data Objects

Description

dbData is a virtual base class for database-backed objects in the dbverse ecosystem. This class provides a common interface for objects that are backed by database tables. It is extended by concrete classes like dbMatrix and dbSpatial that provide specific implementations.

Slots

value tbl_duckdb_connection that represents the data in the database

name name of table within database that contains the data

Extensibility

When creating a new type of database-backed object, extend this class to ensure compatibility with the dbverse ecosystem.

 dbData-connection-accessors

Connection accessor methods for dbData objects

Description

These methods provide a unified interface for accessing and setting database connections across all dbverse packages. They handle different connection storage patterns across various subclasses.

Get or set the database connection for a dbData object.

Usage

```
conn(x)
```

```
conn(x) <- value
```

```
## S4 method for signature 'dbData'
conn(x)
```

```
## S4 replacement method for signature 'dbData'
conn(x) <- value
```

Arguments

x	A dbData object
value	A DBIConnection object

Details

The connection accessor methods support multiple implementation patterns:

1. Nested connection: Some classes (like dbMatrix) store connections in value\$src\$con
2. dbplyr connections: Objects with tbl_duckdb_connection values can use dbplyr::remote_con()

The implementation automatically tries these access patterns in order of specificity, falling back to more generic approaches if needed.

Auto-reconnects if the current connection is invalid.

Value

For getter: The database connection. For setter: The updated object

For getter: DBIConnection. For setter: Updated object.

See Also

[dbData](#)

dbData-extract	<i>Extract and replace methods for dbData objects</i>
----------------	-------------------------------------------------------

Description

Provides extraction (`[]`) and replacement (`[]=`) methods for dbData objects. The extractor attempts to auto-reconnect via `dbReconnect()` before returning the underlying database-backed value.

Arguments

x	A dbData object.
...	Additional arguments passed to the extractor.
value	Replacement value for <code>[]=</code> .

Value

For `[]`: the underlying value stored in the object. For `[]=`: the updated dbData.

dbList	<i>List remote tables, temporary tables, and views</i>
--------	--------------------------------------------------------

Description

A generic function to list tables, temporary tables, and views in a database connection. This provides an enhanced view over `DBI::dbListTables` with categorization.

Usage

```
dbList(conn, ...)
```

Arguments

conn	A DBI database connection
...	Additional arguments passed to methods

Value

Method implementations may return different formats, typically a categorized list

See Also

[DBI::dbListTables\(\)](#)

dbList,DBIConnection-method
<i>List remote tables, temporary tables, and views</i>

Description

Pretty prints tables, temporary tables, and views in the database.

Usage

```
## S4 method for signature 'DBIConnection'  
dbList(conn)
```

Arguments

conn	A DBIConnection object, as returned by DBI::dbConnect() .
------	---------------------------------------------------------------------------

Details

Similar to `DBI::dbListTables`, but categorizes tables into three categories:

- Tables
- Temporary Tables (these will be removed when the connection is closed)
- Views (these may be removed when the connection is closed)

dbLoad	<i>Load a dbverse object from the database</i>
--------	------------------------------------------------

Description

Generic function to reconstruct dbverse objects (`dbMatrix`, `dbSpatial`, etc.) from tables stored in a database connection.

Usage

```
dbLoad(conn, name, class, ...)
```

Arguments

conn	A DBI database connection
name	Character name of the table/view in the database
class	Character class name of the object to load (e.g., "dbMatrix", "dbSpatial")
...	Additional arguments passed to specific methods

Value

A dbverse object of the specified class

dbProject	<i>dbProject: database connection and table management</i>
-----------	------------------------------------------------------------

Description

R6 class for managing DuckDB connections and pinning lazy database tables. Wraps a `pins::board_folder()` with a cached connection object stored as "cachedConnection" for automatic reconnection.

Methods

Public methods:

- `dbProject$new()`
- `dbProject$disconnect()`
- `dbProject$reconnect()`
- `dbProject$set_dbdir()`
- `dbProject$get_conn()`
- `dbProject$get_board()`
- `dbProject$pin_write()`
- `dbProject$pin_delete()`
- `dbProject$pin_read()`
- `dbProject$restore()`
- `dbProject$dbRemoveTable()`
- `dbProject$dbRemoveView()`
- `dbProject$print()`
- `dbProject$is_connected()`
- `dbProject$clone()`

Method `new()`: Create a new `dbProject` object.

Usage:

```
dbProject$new(path, ...)
```

Arguments:

`path` A character string specifying the folder path for pins.

`...` Additional arguments passed to `connections::connection_open()`

Method `disconnect()`: Close the current connection, if any.

Usage:

```
dbProject$disconnect()
```

Method `reconnect()`: Reconnect to the database in the project.

Usage:

```
dbProject$reconnect()
```

Method `set_dbdir()`: Update the project's cached DuckDB database path.

Usage:

```
dbProject$set_dbdir(dbdir)
```

Arguments:

`dbdir` Path to the DuckDB database file.

Details: Overwrites the "cachedConnection" pin with a connection that uses the provided `dbdir`. Uses forced evaluation to avoid pins restoring a connection that references an out-of-scope variable.

Method `get_conn()`: Retrieve the DBI connection from the project, reconnecting if necessary.

Usage:

```
dbProject$get_conn()
```

Returns: A DBIConnection object for direct database operations.

Method `get_board()`: Return the `pins::board_folder()` object used by the project.

Usage:

```
dbProject$get_board()
```

Returns: The `pins::board_folder()` object.

Method `pin_write()`: Write a lazy database tbl to the project.

Usage:

```
dbProject$pin_write(x, name)
```

Arguments:

x A `dplyr::tbl` object to be written to the board.

name A character string specifying the name of the pin.

Returns: The materialized object (for `dbMatrix` or `dbSpatial` objects) pointing to permanent table, or invisibly returns the `dbProject` object for other types.

Method `pin_delete()`: Delete a pin from the project.

Usage:

```
dbProject$pin_delete(name, ...)
```

Arguments:

name A character string specifying the name of the pin to delete.

... Additional arguments passed to `pins::pin_delete()`.

Returns: Invisibly returns the `dbProject` object for method chaining.

Method `pin_read()`: Read a pinned object from the project's board.

Usage:

```
dbProject$pin_read(name)
```

Arguments:

name A character string specifying the name of the pin.

Returns: The object stored in the specified pin.

Method `restore()`: Restore all pins from the board manifest.

Usage:

```
dbProject$restore()
```

Details: Reads the board manifest and restores the cached connection and all pinned objects. The connection is restored internally, while other pinned objects are returned as a named list for the user to assign.

Returns: A named list of restored pinned objects (excluding the connection).

Examples:

```
project_path <- tempfile("dbproject-")
proj <- dbProject$new(path = project_path)
proj$pin_write(data.frame(id = 1:3), "example")
restored <- proj$restore()
names(restored)
proj$disconnect()
unlink(project_path, recursive = TRUE)
```

Method `dbRemoveTable()`: Remove a table from the connected database

Usage:

```
dbProject$dbRemoveTable(name)
```

Arguments:

name A character string specifying the name of the table to remove

Returns: Invisibly returns the dbProject object for method chaining

Method `dbRemoveView()`: Remove a view from the connected database

Usage:

```
dbProject$dbRemoveView(name)
```

Arguments:

name A character string specifying the name of the table to remove

Returns: Invisibly returns the dbProject object for method chaining

Method `print()`: Print summary information, including connection status, path, and board details.

Usage:

```
dbProject$print(...)
```

Arguments:

... Unused arguments (for consistency with generic print method)

Method `is_connected()`: Check if there is an active database connection.

Usage:

```
dbProject$is_connected()
```

Returns: A logical value indicating whether the project has an active connection.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
dbProject$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

[conn\(\)](#) S4 generic for dbData objects

Examples

```
## -----  
## Method `dbProject$restore`  
## -----  
  
project_path <- tempfile("dbproject-")  
proj <- dbProject$new(path = project_path)  
proj$pin_write(data.frame(id = 1:3), "example")  
restored <- proj$restore()  
names(restored)  
proj$disconnect()  
unlink(project_path, recursive = TRUE)
```

dbReconnect

Reconnect database connection for dbData objects

Description

Reconnects invalid database connections for dbData objects, updating the object with the new valid connection.

Usage

```
dbReconnect(x)  
  
## S4 method for signature 'dbData'  
dbReconnect(x)  
  
## S4 method for signature 'DBIConnection'  
dbReconnect(x)  
  
## S4 method for signature 'connConnection'  
dbReconnect(x)
```

Arguments

x A dbData object

Value

The dbData object with an updated connection

read_pin_conn	<i>Read Pin Connection Generic</i>
---------------	------------------------------------

Description

Generic function for reading connection objects from a pins board

Usage

```
read_pin_conn(x)
```

Arguments

x	A pinned connection object
---	----------------------------

read_pin_conn.conn_matrix_table	<i>Read a pinned dbMatrix from pins board</i>
---------------------------------	-----------------------------------------------

Description

S3 method for reading dbMatrix objects from a pins board. Reconstructs the full dbMatrix object.

Usage

```
## S3 method for class 'conn_matrix_table'  
read_pin_conn(x)
```

Arguments

x	A pinned conn_matrix_table object
---	-----------------------------------

Value

A dbMatrix object (dbSparseMatrix or dbDenseMatrix)

```
read_pin_conn.conn_spatial_table
```

Read a pinned dbSpatial object from a pins board

Description

Read a pinned dbSpatial object from a pins board

Usage

```
## S3 method for class 'conn_spatial_table'
read_pin_conn(x)
```

Arguments

x A pinned conn_spatial_table object

Value

A dbSpatial object

```
to_view
```

Convert lazy table to named view

Description

Convert lazy table to named view

Usage

```
to_view(x, name, temporary = TRUE, overwrite = TRUE, ...)
```

Arguments

x A lazy table object (tbl_duckdb_connection)

name Character name to assign view within database

temporary Logical, if TRUE (default), the view will be deleted after the session ends

overwrite Logical, if TRUE (default), the view will overwrite an existing view

... Additional arguments passed to DBI::dbExecute

to_view,ANY-method	<i>Convert lazy table to named VIEW</i>
--------------------	-----------------------------------------

Description

Convert lazy table to named VIEW

Usage

```
## S4 method for signature 'ANY'
to_view(x, name, temporary = TRUE, overwrite = TRUE, ...)
```

Arguments

x	tbl_sql Required. tbl_sql object to convert to a VIEW.
name	character Required. Name to assign VIEW within database. Auto-generated if none provided with prefix "tmp_view_"
temporary	logical If TRUE (default), the VIEW will not be saved in the database
overwrite	logical If TRUE (default), the VIEW will overwrite an existing VIEW of the same name
...	Additional arguments passed to DBI::dbExecute()

write_pin_conn	<i>Write Pin Connection Generic</i>
----------------	-------------------------------------

Description

Generic function for writing connection objects to a pins board

Usage

```
write_pin_conn(x, board, ...)
```

Arguments

x	Object to write
board	A pins board object
...	Additional arguments passed to methods

```
write_pin_conn.dbMatrix
```

Write a dbMatrix object to a pins board

Description

S3 method for writing dbMatrix objects to a pins board while maintaining connection state and metadata consistent with the connections package.

Usage

```
## S3 method for class 'dbMatrix'
write_pin_conn(x, board, name, ...)
```

Arguments

x	A dbMatrix object (dbSparseMatrix or dbDenseMatrix)
board	A pins <code>pins::board_folder</code> object
name	Name for the pin (required)
...	Additional arguments passed to <code>pins::pin_write()</code>

Value

Invisibly returns the input object

```
write_pin_conn.dbSpatial
```

Write a dbSpatial object to a pins board

Description

Write a dbSpatial object to a pins board

Usage

```
## S3 method for class 'dbSpatial'
write_pin_conn(x, board, name, ...)
```

Arguments

x	A dbSpatial object
board	A pins board object
name	Name for the pin
...	Additional arguments passed to <code>pins::pin_write</code>

Value

Invisibly returns the input object

`write_pin_conn.default`

Default method for write_pin_conn

Description

Fallback method for non-database objects. Delegates directly to `pins::pin_write()` for regular R objects (matrices, data.frames, vectors, etc.)

Usage

```
## Default S3 method:  
write_pin_conn(x, board, ...)
```

Arguments

<code>x</code>	Any R object
<code>board</code>	A pins board object
<code>...</code>	Additional arguments passed to <code>pins::pin_write()</code>

Value

Invisibly returns the input object

Index

* **dbData**

- dbList, DBIConnection-method, [5](#)
- to_view, ANY-method, [13](#)
- [(dbData-extract), [4](#)
- [, dbData-method (dbData-extract), [4](#)
- [<- , dbData-method (dbData-extract), [4](#)

- conn (dbData-connection-accessors), [3](#)
- conn(), [9](#)
- conn, dbData-method
 - (dbData-connection-accessors), [3](#)
- conn<- (dbData-connection-accessors), [3](#)
- conn<- , dbData-method
 - (dbData-connection-accessors), [3](#)
- connection_pin_read, [2](#)
- connections::connection_open(), [7](#)

- dbData, [4](#)
- dbData-class, [3](#)
- dbData-connection-accessors, [3](#)
- dbData-extract, [4](#)
- DBI::dbConnect(), [5](#)
- DBI::dbExecute(), [13](#)
- DBI::dbListTables(), [5](#)
- dbList, [5](#)
- dbList, DBIConnection-method, [5](#)
- dbLoad, [6](#)
- dbProject, [6](#)
- dbReconnect, [10](#)
- dbReconnect(), [4](#)
- dbReconnect, connConnection-method
 - (dbReconnect), [10](#)
- dbReconnect, dbData-method
 - (dbReconnect), [10](#)
- dbReconnect, DBIConnection-method
 - (dbReconnect), [10](#)
- dplyr::tbl, [8](#)

- pins::board_folder, [14](#)
- pins::board_folder(), [6, 8](#)
- pins::pin_delete(), [8](#)
- pins::pin_write(), [14](#)

- read_pin_conn, [11](#)
- read_pin_conn.conn_matrix_table, [11](#)
- read_pin_conn.conn_spatial_table, [12](#)

- to_view, [12](#)
- to_view, ANY-method, [13](#)

- write_pin_conn, [13](#)
- write_pin_conn.dbMatrix, [14](#)
- write_pin_conn.dbSpatial, [14](#)
- write_pin_conn.default, [15](#)