Package: rpgconn (via r-universe)

October 19, 2026

Type Package

Title User-Friendly PostgreSQL Connection Management

Version 0.4.1.9000

Maintainer Bobby Fatemi <bfatemi07@gmail.com>

Description Provides a user-friendly interface for managing PostgreSQL database connection settings. Supports standard PostgreSQL URI connection strings (postgresql://user:pass@host:port/db) with query parameters, IPv6 hosts, and URL-encoded credentials. The package supplies helper functions to create, edit and load connection and option configuration files stored in a user-specific directory using the 'RPostgres' back end. These helpers make it easy to construct a reproducible connection string from a configuration file, either by reading user-defined YAML files or by parsing an environment variable. Validation-first architecture provides clear, actionable error messages instead of cryptic connection failures.

Encoding UTF-8

License MIT + file LICENSE

Language en-US

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

URL https://github.com/r-data-science/rpgconn, https://r-data-science.github.io/rpgconn/

BugReports https://github.com/r-data-science/rpgconn/issues

Imports DBI, fs, RPostgres, stringr, usethis, yaml

Suggests covr, data.table, devtools, knitr, pak, pkgdown, pool, quarto, rcmdcheck, roxygen2, rstudioapi, shiny, spelling, testthat (>= 3.0.0), urlchecker, withr

Config/testthat/edition 3

VignetteBuilder quarto

Config/pak/sysreqs cmake git make libgit2-dev libicu-dev libuv1-dev libssl-dev libpq-dev libx11-dev

Repository https://r-data-science.r-universe.dev

Date/Publication 2026-02-21 08:19:24 UTC

RemoteUrl https://github.com/r-data-science/rpgconn

RemoteRef HEAD

RemoteSha 0231c3562b2756c930ae9c5ae929a43fe7d981e8

Contents

pgdbconn 2
use_config 5

Index **7**

---

pgdbconn *PostgreSQL Database Connection Management*

Description

Simplified interface for connecting to PostgreSQL databases using either environment variables, URI connection strings, or YAML configuration files.

Usage

```
dbc(
  cfg = NULL,
  db = NULL,
  args_only = FALSE,
  cfg_path = NULL,
  opt_path = NULL,
  path = NULL
)

dbd(cn)

init_yamls()

dir_rpg()

edit_config()

edit_options()
```

Arguments

cfg	A connection configuration name used when loading connection arguments from internally stored YAML files. If ‘NULL‘ (the default), connection arguments are read from the environment variable ‘RPG_CONN_STRING‘.
db	Database name. If ‘NULL‘, the ‘dbname‘ value from the connection string or configuration is used. Required when using ‘cfg‘ parameter.
args_only	Logical. If ‘TRUE‘, return only the connection arguments without establishing a connection. If ‘FALSE‘ (default), establish and return the connection.
cfg_path	Optional path to override default database config file location. Useful for testing or project-specific configurations.
opt_path	Optional path to override default database options file location.
path	Deprecated alias for ‘cfg_path‘. Retained for backward compatibility.
cn	A database connection object to disconnect.

Details

Why rpgconn?

Managing PostgreSQL connections across environments (local, staging, production) is error-prone and insecure. rpgconn eliminates this friction by:

- **Portable**: Uses standard PostgreSQL URI format that works across languages and tools - **Secure**: Stores configurations locally in user directories, never in version control - **Flexible**: Supports URI format, keyword/value format, and YAML configs - **Fail-fast**: Validates connection strings upfront with clear error messages - **Zero friction**: Single function call replaces repetitive ‘DBI::dbConnect()‘ boilerplate

Connection Methods

Method 1: Environment Variable (Recommended for Cloud)

Set ‘RPG_CONN_STRING‘ to a PostgreSQL URI:

“‘r Sys.setenv(RPG_CONN_STRING = "postgresql://user:pass@host:5432/db") cn <- dbc() “‘

Supports query parameters for SSL and other options:

“‘r Sys.setenv(RPG_CONN_STRING = "postgresql://user:pass@host/db?sslmode=require") “‘

Method 2: YAML Configuration (Recommended for Teams)

Initialize and edit config files:

“‘r init_yamls() edit_config() # Opens ~/.config/rpgconn/config.yml “‘

Then connect using named configurations:

“‘r cn <- dbc(cfg = "local", db = "mydb") “‘

Connection String Formats

The package supports multiple PostgreSQL connection string formats:

- **URI**: ‘postgresql://user:pass@host:5432/db?sslmode=require‘ - **Keyword/value (semicolon)**: ‘user=...;password=...;host=...;port=...;dbname=...‘ - **Keyword/value (whitespace)**: ‘host=... user=... dbname=... port=...‘

See ‘vignette("advanced-workflow")‘ for comprehensive format documentation.

Value

- ‘dbc()‘: A database connection object (class ‘PqConnection‘) when ‘args_only = FALSE‘, or a named list of connection arguments when ‘args_only = TRUE‘. - ‘dbd()‘: ‘NULL‘, invisibly. Called for side effect of closing connection. - ‘init_yamls()‘: Path to rpgconn config directory, invisibly. - ‘dir_rpg()‘: Character string path to rpgconn config directory. - ‘edit_config()‘: Path to config file, invisibly. - ‘edit_options()‘: Path to options file, invisibly.

Functions

- `dbc()`: Connect to a database or return the connection arguments
- `dbd()`: Disconnect from a database
- `init_yamls()`: Initialize connection files
- `dir_rpg()`: get the path to the rpg settings directory
- `edit_config()`: edit the internally configured connection parameters
- `edit_options()`: edit the internally configured connection options

See Also

- [RPostgres::Postgres()] for the underlying PostgreSQL driver - [DBI::dbConnect()] for lower-level connection details - [use_config()] for adopting external configuration files

Other configuration: `use_config()`

Examples

```
## Not run:
# Method 1: Using connection string (cloud databases)
Sys.setenv(RPG_CONN_STRING = "postgresql://user:pass@localhost:5432/mydb")
cn <- dbc()
DBI::dbGetQuery(cn, "SELECT version()")
dbd(cn)

# Method 2: Using YAML config (team/local setups)
init_yamls()
edit_config()  # Edit ~/.config/rpgconn/config.yml
cn <- dbc(cfg = "local", db = "mydb")
dbd(cn)

# Get connection args without connecting (useful for debugging)
args <- dbc(args_only = TRUE)
str(args)

# Use custom config file (testing, project-specific)
tmp_cfg <- tempfile(fileext = ".yml")
yaml::write_yaml(list(config = list(test = list(host = "localhost", port = 5432))), tmp_cfg)
args <- dbc(cfg = "test", db = "testdb", args_only = TRUE, cfg_path = tmp_cfg)
str(args)

## End(Not run)
```

use_config *Adopt External Configuration File*

Description

Replace rpgconn's active configuration with an external YAML file. This function solves the problem of managing multiple config files across environments without manual file copying.

Usage

```
use_config(path, overwrite = FALSE)
```

Arguments

path	Character string. Path to YAML config file to adopt. Can be absolute or relative to current working directory.
overwrite	Logical. If 'TRUE', overwrite existing config. If 'FALSE' (default), error if config already exists.

Details

Why use_config()?

This function addresses several common pain points:

Problem 1: Team Collaboration Teams often maintain a shared config in their project repository. Instead of manually copying 'team-config.yml' to '~/.config/rpgconn/config.yml', just call 'use_config("team-config.yml")'.

Problem 2: Environment-Specific Configs Different environments (dev/staging/prod) need different configs. Store each as a separate file and switch between them:

"'r use_config("config-dev.yml") # Use dev database use_config("config-prod.yml", overwrite = TRUE) # Switch to prod "'

Problem 3: Testing with Fixtures Tests need isolated, reproducible configurations:

"'r use_config("config-test.yml") "'

How It Works

1. Reads and validates the YAML file at 'path' 2. Copies it to '~/.config/rpgconn/config.yml' (or platform equivalent) 3. All subsequent 'dbc(cfg = "name")' calls use the new config

The function ensures the directory structure exists before copying, making it safe to use even on fresh installations.

Config File Format

Config files should follow this structure:

"'yaml config: local: host: localhost port: 5432 prod: host: prod.example.com port: 5432 user: app_user password: secret "'

See 'vignette("quickstart")' for complete examples.

Value

Invisibly returns the path to the active configuration file after replacement. In non-interactive sessions, the path is returned silently without opening an editor.

See Also

- [edit_config()] to edit the active configuration file - [init_yamls()] to initialize default configuration files - [dbc()] to connect using the active configuration

Other configuration: pgdbconn

Examples

```
## Not run:
# Adopt project-level config
use_config("project_config.yml")

# Force overwrite existing config
use_config("new_config.yml", overwrite = TRUE)

# Use environment-specific config
env <- Sys.getenv("APP_ENV", "dev")
use_config(paste0("config-", env, ".yml"))

# Testing with fixture
withr::with_tempfile("temp_config", {
  yaml::write_yaml(
    list(config = list(test = list(host = "localhost", port = 5432))),
    temp_config
  )
  use_config(temp_config, overwrite = TRUE)
  args <- dbc(cfg = "test", db = "testdb", args_only = TRUE)
  str(args)
})

## End(Not run)
```

Index

∗ **config-management**
  pgdbconn, 2
  use_config, 5
∗ **configuration**
  pgdbconn, 2
  use_config, 5
∗ **connection**
  pgdbconn, 2
∗ **database-config**
  pgdbconn, 2
∗ **database-connection**
  pgdbconn, 2
∗ **database**
  pgdbconn, 2
  use_config, 5
∗ **postgresql**
  pgdbconn, 2

dbc *(*pgdbconn*)*, 2
dbd *(*pgdbconn*)*, 2
dir_rpg *(*pgdbconn*)*, 2

edit_config *(*pgdbconn*)*, 2
edit_options *(*pgdbconn*)*, 2

init_yamls *(*pgdbconn*)*, 2

pgdbconn, 2, *6*

use_config, *4*, 5