1. Connection Setup

Cob Staines

2025-04-01

Table of contents

This tutorial is available as a .qmd on Github.

Motivation

- Connect to the RIBBiTR (or another remote) database with ease and repeatability
- Manage login credentials for ease and security, so they won't be lost or shared with your code

R

Here is a quick tutorial to (re)orient ourselves to connecting to the RIBBiTR database useing R/RStudio. If you aren't yet familiar with R/Rstudio, check out this quick Getting Started tutorial by POSIT.

Store & access your database connection parameters

Access your local .Renviron file

Your .Renviron file a local file where you can save and reference your login credentials for easy use within R and RStudio, without risking losing them or potentially sharing them on accident when you share your code. A simple way to access your .Renviron file is with the function `usethis::edit_r_environ()`

```r
install.packages("usethis")

# open your local .Reniron file
usethis::edit_r_environ()
```

Save connections parameters

Copy the following database connection parameters to your .Renviron file, substituting your login credentials (user & password).

```
# RIBBiTR DB credentials
ribbitr.dbname = "ribbitr"
ribbitr.host = "ribbitr.c6p56tuocn5n.us-west-1.rds.amazonaws.com"
ribbitr.port = "5432"
ribbitr.user = "[YOUR-USERNAME-HERE]"
ribbitr.password = "[YOUR-PASSWORD-HERE]"
```

Save and close .Renviron, and restart RStudio.

Establish database connection

Create a new R project (or .qmd, .Rmd, .R etc.) file where you can follow the tutorial and establish the database connection.

Load packages

Rtools

Rtools is required to build R packages locally, as part of this tutorial. Check to see if Rtools is installed by running:

```r
Sys.which("make")
```

```
           make
"/usr/bin/make"
```

If this returns a path to the make function (e.g. `/usr/bin/make`), Rtools is installed and you can proceed to the next step. If you return an empty string `""` you will need to download Rtools first.

Librarian

"librarian" is a package and library management package in R which makes it easier to install, load, update and unload packages to meet dynamic environment needs. There are other ways to download, load, and maintain packages in R (e.g. `install.packages()` and `library()`, but we recommend librarian for its simplicity and portability.

```r
# install and load "librarian" R package
install.packages("librarian")
```

librarian downloads and loads packages using the `librarian::shelf` function. Below are the minimal recommended packages to establish a connection to the RIBBiTR database.

```r
# update your ribbitrrr package to the latest version
librarian::shelf(RIBBiTR-BII/ribbitrrr, update_all = TRUE)
```

```

  These packages will be installed:

  'ribbitrrr'

  It may take some time.
```

```
Downloading GitHub repo RIBBiTR-BII/ribbitrrr@HEAD
```

```
-- R CMD build -----------------------------------------------------------------
* checking for file ‘/tmp/RtmpDkcw0g/remotes1dc8f1e873410/RIBBiTR-BII-ribbitrrr-ec3d34b/DESC
* preparing ‘ribbitrrr’:
* checking DESCRIPTION meta-information ... OK
* checking for LF line-endings in source and make files and shell scripts
* checking for empty or unneeded directories
* building ‘ribbitrrr_0.0.2.0.tar.gz’
```

```
# minimal packages for establishing RIBBiTR DB connection
librarian::shelf(tidyverse, dbplyr, RPostgres, DBI, RIBBiTR-BII/ribbitrrr)
```

Connect

Now, using the `ribbitrrr:hopToDB()` function, let's establish a connection!

```
# establish database connection
dbcon <- hopToDB("ribbitr")
```

```
Connecting to 'ribbitr'... Success!
```

`hopToDB()` returns a database connection object (`dbcon`). Keep track of this, you will call it to explore and pull data later.

Begin using your connection!

Try out your connection by loading table metadata from the database

```
mdt <- tbl(dbcon, Id("public", "all_tables")) %>%
  collect()
head(mdt)
```

```
# A tibble: 6 x 4
  table_schema      table_name       column_count table_description
  <chr>             <chr>                 <int64> <chr>
1 microclimate_data metadata_columns           25 <NA>
2 microclimate_data metadata_tables             4 <NA>
3 microclimate_data logger                      4 <NA>
4 microclimate_data sensor                      5 <NA>
5 microclimate_data ts_dew_point                3 <NA>
6 microclimate_data ts_illuminance              3 <NA>
```

Disconnect

It is good practice to close your database connection, to let the server know it can stop listening for you (otherwise it will continue to use server resources in anticipation of your). You can think of this as saying goodbye at the end of a phone call.

```
# disconnect from database
dbDisconnect(dbcon)
```

Also try

- For those managing multiple database connections, the `hopToDB()` function allows you to store and fetch various sets of login credentials with a single keyword. Just substitute "ribbitr" in the .Renviron example above with your own keywords to juggle multiple logins.
- Your login credentials can also be accessed explicitly anytime using `Sys.getenv("ribbitr.dbname")`, etc. In most cases the `hopToDB()` function is all you will need, however.

Python

Here is a quick tutorial to (re)orient ourselves to connecting to the RIBBiTR database useing Python. If you aren't yet familiar with Python, check out this quick Getting Started tutorial by DATAQUEST.

Store & access your database connection parameters

Create a dbconfig file

We recommend you create a local database config (`dbconfig.py`) file where you can save and reference your login credentials for easy use in python, without risking losing them or potentially sharing them on accident when you share your code.

Create a file nammed `dbconfig.py` in your project working directory (or another preferred location, see "Also try" below). Copy the following to `dbconfig.py`:

```
# dbconfig.py

ribbitr = {
  "database":"ribbitr",
  "host":"ribbitr.c6p56tuocn5n.us-west-1.rds.amazonaws.com",
  "port":"5432",
  "user":"[YOUR-USERNAME-HERE]",
  "password":"[YOUR-PASSWORD-HERE]",
}
```

Save `dbconfig.py`.

Be sure to add `dbconfig.py` to your local `.gitignore` file if you are using git/github, so you don't accidentally publish you login credentials!

Establish database connection

Create a new .py (or .qmd, .ipynb, etc.) file where you can follow the tutorial and establish the database connection.

Import packages

This method requires installing the ibis.postgres package to your working environment, in addition to pandas. We also import the `dbconfig.py` file to access your login credentials.

```
import ibis
import pandas as pd
import dbconfig  # import connection credentials
```

Connect

Now, using the `ibis.postgres.connect()` function, let's establish a connection!

```
# establish database connection
dbcon = ibis.postgres.connect(**dbconfig.ribbitr)
```

`ibis.postgres.connect()` returns a database connection object (`dbcon`). Keep track of this, you will call it to explore and pull data later.

Begin using your connection

Try out your connection by loading table metadata from the database

```
mdt = dbcon.table(database = "public", name = "all_tables").to_pandas()
mdt.head()
```

```
        table_schema        table_name  column_count table_description
0  microclimate_data  metadata_columns            25              None
1  microclimate_data   metadata_tables             4              None
2  microclimate_data            logger             4              None
3  microclimate_data            sensor             5              None
4  microclimate_data      ts_dew_point             3              None
```

Disconnect

It is good practice to close your database connection, to let the server know it can stop listening for you (otherwise it will continue to use server resources in anticipation of your). You can think of this as saying goodbye at the end of a phone call.

```
# close the connection
dbcon.disconnect()
```

Also try

- For those managing multiple database connections, this method allows you to store and fetch various sets of login credentials with a single keyword. Just substitute "ribbitr" in the `dbconfig.py` file with your own keywords and call them as needed!
- If you will be connecting to the database from different python projects, you may want to save your `dbconfig.py` file to a more general location. In this case, include the following lines in each of your project files:

```
import sys
sys.path.append("/path/to/dbconfig/dir/")
import dbconfig
```

DBeaver

Here is a quick tutorial to (re)orient ourselves to connecting to the RIBBiTR database useing DBeaver. DBeaver is a free, universal database tool which provides a great visual platform and robust backend for interacting with databases. Download DBeaver and install to get started with the tutorial.

Set up and test your connection.

1) Open DBeaver.
2) Right-click in the Database Navigator" panel and select `Create -> Connection`

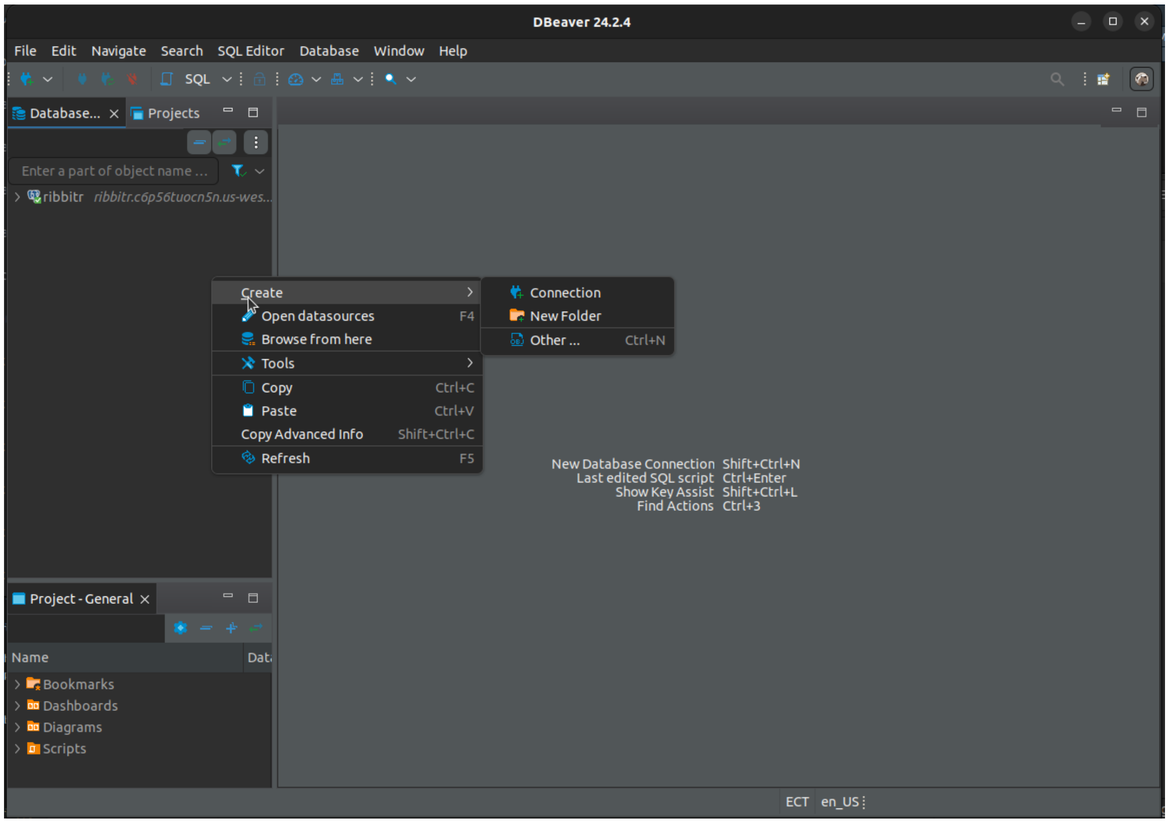
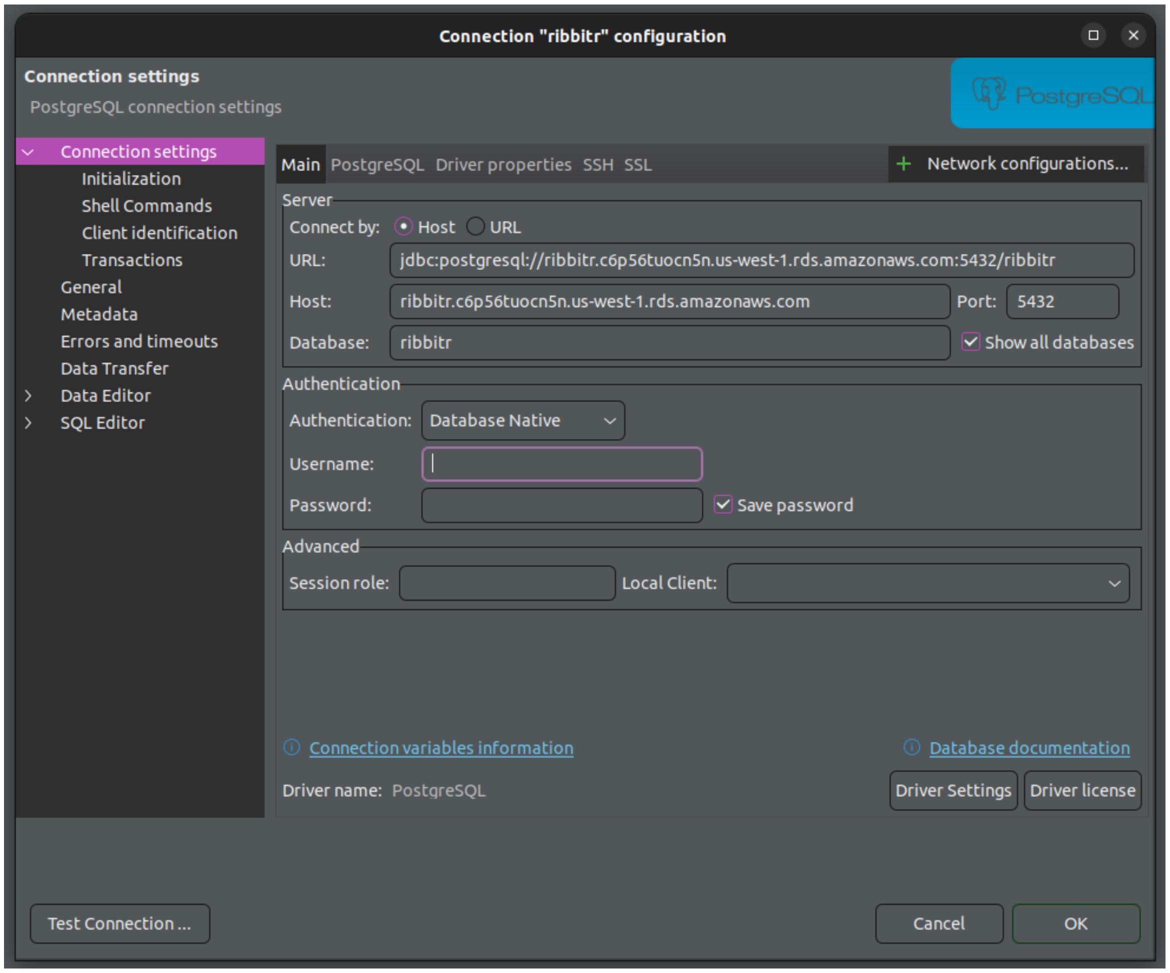

DBeaver 24.2.4
File Edit Navigate Search SQL Editor Database Window Help
SQL
Database... Projects
Enter a part of object name ...
ribbitr ribbitr.c6p56tuocn5n.us-wes...
Create
Open datasources F4
Browse from here
Tools
Copy Ctrl+C
Paste Ctrl+V
Copy Advanced Info Shift+Ctrl+C
Refresh F5
Connection
New Folder
Other ... Ctrl+N
New Database Connection Shift+Ctrl+N
Last edited SQL script Ctrl+Enter
Show Key Assist Shift+Ctrl+L
Find Actions Ctrl+3
Project - General
Name
Dat
Bookmarks
Dashboards
Diagrams
Scripts
ECT en_US

Figure 1: Right-click in the Database Navigator" panel and select `Create -> Connection`

3) Enter your database credentials as shown above, substituting your user and password.


Connection "ribbitr" configuration

Connection settings
PostgreSQL connection settings

Server
Connect by: [✓ Host] URL
URL: jdbc:postgresql://ribbitr.c6p56tuocn5n.us-west-1.rds.amazonaws.com:5432/ribbitr
Host: ribbitr.c6p56tuocn5n.us-west-1.rds.amazonaws.com
Port: 5432
Database: ribbitr
[✓] Show all databases

Authentication
Authentication: Database Native
Username:
Password:
[✓] Save password

Advanced
Session role:
Local Client:

Driver name: PostgreSQL

Figure 2: Enter your database credentials as shown above, substituting your user and password.

4) Click `Test Connection` in the bottom left to make sure your credentials work. Then click `OK`.

That's it, you are ready to start using your connection!

Disconnect

When you are done, click the red `Disconnect` button in top left corner to terminate your connection.

It is good practice to close your database connection, to let the server know it can stop listening for you (otherwise it will continue to use server resources in anticipation of your). You can think of this as saying goodbye at the end of a phone call.

2. Data Discovery ->