RIBBiTR Database Connection Setup

Cob Staines

2024-11-01

Table of contents

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 Database Connection

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()

```
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

"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.

```
# install and load "librarian"
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.

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

Connect

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

```
# establish database connection
dbcon = hopToDB("ribbitr")
```

```
Connecting to database... 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 bay_area     amphib_dissect               41 <NA>
2 bay_area     amphib_parasite              11 <NA>
```

```
3 bay_area     water_quality_info          27 <NA>
4 bay_area     site                        25 <NA>
5 bay_area     wetland_info                25 <NA>
6 bay_area     bd_results                  25 <NA>
```

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 example above with your own keyword!
- Your login credentials can also be accessed explicitly anytime using Sys.getenv("ribbitr.dbname"), etc. In most cases the HopToDB() function is all you need, however.

Python Database Connection

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     bay_area      amphib_dissect            41              None
1     bay_area     amphib_parasite            11              None
2     bay_area  water_quality_info            27              None
3     bay_area                site            25              None
4     bay_area        wetland_info            25              None
```

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
```