7. Database Refresher

Cob Staines

2025-04-07

Table of contents

This tutorial is available as a .qmd on Github.

Motivation

- Refresh ourselves on how to connect, browse, join, pull, and organize data
- Demonstrate linking data from different datasets

Resources

To navigate and query the data we need to test our hypothesis, will use the following resources:

- **RIBBiTR Database** - a digital collection of field and lab data from across the RIBBiTR project, hosted on the web and accessible (with a login) via a database client (e.g. DBeaver) or database backend (e.g. dbplyr for R, ibis for Python). This database includes:
  - Data tables - information about field and lab observations
  - Metadata tables - information about how these data are collected, stored, and represented
- **RIBBiTR Data Access Center** - A Github pages website (this one!) for RIBBiTR data access resources, including:
  - Tutorials
  - Schema Diagrams
    - survey_data
    - microclimate_data
  - Database Changelog
- **ribbitrrr (R package)** - An R package with a handful of scripts for facilitated interaction with the RIBBiTR database.

Scenario

Suppose we want to explore a hypothesis that temperature is related to Bd load. Let's explore and collect data from the RIBBiTR database to use in testing our hypothesis.

R Setup

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

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

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

```
# load table metadata
mdt <- tbl(dbcon, Id("public", "all_tables")) %>%
  filter(table_schema == "survey_data") %>%
  collect()

# load column metadata
mdc <- tbl(dbcon, Id("survey_data", "metadata_columns")) %>%
  filter(table_schema == "survey_data") %>%
  collect()
```

Data discovery

The `ribbitrrr:hopToDB()` function used to establish our database connection now supports browsing in the RStudio connections panel, to preview data.

The metadata we downloaded also provides important definitions and context to guide us through these data. Check out `mdt` and `mdc` to get an idea.

Taking a look at the database schema for survey_data, we might determine that our observation tables of interest are: "bd_qpcr_results", "capture", and "environmental". Let's create "pointers" to these tables, plus the related supporting tables:

Table pointers

Supported with the R package `dbplyr`, you can think of these table pointers as addresses, an object which points R to where the actual data table is stored on the database, but which you can interact with similar to a data table (with some nuances…)

```
# pointers for all tables of interest
# format var_name = tbl(dbcon, Id(schema_name, table_name))
db_bdqpcr = tbl(dbcon, Id("survey_data", "bd_qpcr_results"))
db_sample = tbl(dbcon, Id("survey_data", "sample"))
db_capture = tbl(dbcon, Id("survey_data", "capture"))
db_env = tbl(dbcon, Id("survey_data", "environmental"))
db_survey = tbl(dbcon, Id("survey_data", "survey"))
db_visit = tbl(dbcon, Id("survey_data", "visit"))
db_site = tbl(dbcon, Id("survey_data", "site"))
db_region = tbl(dbcon, Id("survey_data", "region"))
db_country = tbl(dbcon, Id("survey_data", "country"))

# we will also use the following lookup tables
db_lab = tbl(dbcon, Id("survey_data", "lab"))
db_taxa = tbl(dbcon, Id("survey_data", "taxonomy"))
```

Starting from Captures

Let's begin with our central observation table "capture", with corresponding table pointer `db_capture`. We can always `collect()` or `pull()` the data from the database when we are ready, but there is a lot we can do with the table pointer prior to collecting data to explore them, whittling down our query, to avoid pulling unnecessarily large datasets every time we run a code block:

```r
# to collect data, use...
data_capture = db_capture %>%
  collect()
#... but this is a big query. Can we be smarter about how we do this?

# see what columns come from specified tables using the pointer, just like a data table
colnames(db_capture)
```

```
 [1] "taxon_capture"          "time_of_capture"        "capture_transect_m"
 [4] "microhabitat_type"      "body_temp_c"            "substrate_temp_c"
 [7] "svl_mm"                 "body_mass_g"            "life_stage"
[10] "sex"                    "capture_animal_state"   "comments_capture"
[13] "photo"                  "photo_id"               "microhabitat_detailed"
[16] "body_and_bag_mass_g"    "bag_mass_g"             "marked"
[19] "capture_utme"           "capture_utmn"           "capture_type"
[22] "observer_capture"       "bag_id"                 "processor"
[25] "cmr_id"                 "microhabitat_notes"     "tail_length_mm"
[28] "bucket"                 "inside_outside_serdp"   "temp_gun"
[31] "clearcut"               "number_of_mites"        "flir"
[34] "tad_stage"              "capture_id"             "survey_id"
[37] "microhabitat_wet"       "capture_utm_zone"       "capture_latitude"
[40] "capture_longitude"
```

```r
# we can count the rows of this table
db_capture %>%
  count()
```

```
# Source:   SQL [?? x 1]
# Database: postgres  [cob_reads@ribbitr.c6p56tuocn5n.us-west-1.rds.amazonaws.com:5432/ribbi
        n
  <int64>
1   80280
```

```r
# left join supporting tables
db_capture_country = db_capture %>%
  left_join(db_survey, by = "survey_id") %>%
  left_join(db_visit, by = "visit_id") %>%
  left_join(db_site, by = "site_id") %>%
  left_join(db_region, by = "region_id") %>%
  left_join(db_country, by = "country_id")

# see what columns are available in the joined table pointer
colnames(db_capture_country)
```

```
 [1] "taxon_capture"          "time_of_capture"        "capture_transect_m"
 [4] "microhabitat_type"      "body_temp_c"            "substrate_temp_c"
 [7] "svl_mm"                 "body_mass_g"            "life_stage"
[10] "sex"                    "capture_animal_state"   "comments_capture"
[13] "photo"                  "photo_id"               "microhabitat_detailed"
[16] "body_and_bag_mass_g"    "bag_mass_g"             "marked"
[19] "capture_utme"           "capture_utmn"           "capture_type"
[22] "observer_capture"       "bag_id"                 "processor"
[25] "cmr_id"                 "microhabitat_notes"     "tail_length_mm"
[28] "bucket"                 "inside_outside_serdp"   "temp_gun"
[31] "clearcut"               "number_of_mites"        "flir"
[34] "tad_stage"              "capture_id"             "survey_id"
[37] "microhabitat_wet"       "capture_utm_zone"       "capture_latitude"
[40] "capture_longitude"      "start_time"             "end_time"
[43] "detection_type"         "duration_minutes"       "observers_survey"
[46] "comments_survey"        "description"            "survey_quality"
[49] "transect"               "number_observers"       "visit_id"
[52] "start_timestamp_utc"    "end_timestamp_utc"      "date"
[55] "time_of_day"            "campaign"               "visit_status"
[58] "comments_visit"         "site_id"                "visit_lab"
[61] "project"                "site"                   "site_utm_zone"
[64] "site_utme"              "site_utmn"              "area_sqr_m"
[67] "site_code"              "site_elevation_m"       "depth_m"
[70] "topo"                   "wilderness"             "site_comments"
[73] "region_id"              "site_name_alt"          "site_latitude"
[76] "site_longitude"         "geographic_area"        "geographic_area_type"
[79] "region"                 "country_id"             "time_zone"
[82] "country"                "iso_country_code"
```

Combining Bd qPCR results with Captures

```
# Two ways we can build on our query above:

# 1) copy the full code, and add to it...
db_bd_country = db_bdqpcr %>%
  left_join(db_sample, by = "sample_id") %>%
  left_join(db_capture, by = "capture_id") %>%  # left_join bd results and capture
  left_join(db_survey, by = "survey_id") %>%
  left_join(db_visit, by = "visit_id") %>%
  left_join(db_site, by = "site_id") %>%
  left_join(db_region, by = "region_id") %>%
  left_join(db_country, by = "country_id") %>%
  filter(!is.na(capture_id)) # filter to drop any results with no capture
```

```
# ... or 2) build on the `db_capture_country` pointer we defined above
db_bd_country = db_bdqpcr %>%
  left_join(db_sample, by = "sample_id") %>%
  inner_join(db_capture_country, by = "capture_id")  # inner join to include only rows which

# see what columns our `db_bd_country` table pointer has
colnames(db_bd_country)
```

```
 [1] "result_id"                    "sample_id"
 [3] "sample_name_bd"               "detected"
 [5] "replicate"                    "replicate_count"
 [7] "replicate_detected"           "average_ct"
 [9] "average_target_quant"         "total_qpcr_volume_uL"
[11] "qpcr_dilution_factor"         "volume_template_dna_uL"
[13] "extract_volume_uL"            "target_quant_per_swab"
[15] "average_its1_copies_per_swab" "swab_type"
[17] "standard_target_type"         "standard"
[19] "master_mix"                   "extraction_plate_name"
[21] "extraction_date"              "extraction_kit"
[23] "extraction_lab"               "qpcr_plate_name"
[25] "qpcr_well"                    "qpcr_plate_run"
[27] "qpcr_date"                    "qpcr_machine"
[29] "qpcr_lab"                     "comments_qpcr"
[31] "sample_name"                  "sample_type"
[33] "capture_id"                   "sample_name_conflict"
[35] "taxon_capture"                "time_of_capture"
[37] "capture_transect_m"           "microhabitat_type"
[39] "body_temp_c"                  "substrate_temp_c"
[41] "svl_mm"                       "body_mass_g"
[43] "life_stage"                   "sex"
[45] "capture_animal_state"         "comments_capture"
[47] "photo"                        "photo_id"
[49] "microhabitat_detailed"        "body_and_bag_mass_g"
[51] "bag_mass_g"                   "marked"
[53] "capture_utme"                 "capture_utmn"
[55] "capture_type"                 "observer_capture"
[57] "bag_id"                       "processor"
[59] "cmr_id"                       "microhabitat_notes"
[61] "tail_length_mm"               "bucket"
[63] "inside_outside_serdp"         "temp_gun"
[65] "clearcut"                     "number_of_mites"
[67] "flir"                         "tad_stage"
[69] "survey_id"                    "microhabitat_wet"
[71] "capture_utm_zone"             "capture_latitude"
```

```
 [73] "capture_longitude"           "start_time"
 [75] "end_time"                    "detection_type"
 [77] "duration_minutes"            "observers_survey"
 [79] "comments_survey"             "description"
 [81] "survey_quality"              "transect"
 [83] "number_observers"            "visit_id"
 [85] "start_timestamp_utc"         "end_timestamp_utc"
 [87] "date"                        "time_of_day"
 [89] "campaign"                    "visit_status"
 [91] "comments_visit"              "site_id"
 [93] "visit_lab"                   "project"
 [95] "site"                        "site_utm_zone"
 [97] "site_utme"                   "site_utmn"
 [99] "area_sqr_m"                  "site_code"
[101] "site_elevation_m"            "depth_m"
[103] "topo"                        "wilderness"
[105] "site_comments"               "region_id"
[107] "site_name_alt"               "site_latitude"
[109] "site_longitude"              "geographic_area"
[111] "geographic_area_type"        "region"
[113] "country_id"                  "time_zone"
[115] "country"                     "iso_country_code"
```

Join with environmental observations

We want to include environmental observations like water and air temperature, found in the `environmental` table. To align these with the capture data, let's aggregate and join at the visit level:

```
# view environmental table columns
colnames(db_env)
```

```
 [1] "environmental_id"             "survey_id"
 [3] "wind_speed_m_s"               "air_temp_c"
 [5] "water_temp_c"                 "p_h"
 [7] "tds_ppm"                      "wind"
 [9] "sky"                          "air_time"
[11] "water_time"                   "sample_loccation_desctiption"
[13] "dissolved_o2_percent"         "salinity_ppt"
[15] "cloud_cover_percent"          "precip"
[17] "soil_moisture_m3_m3"          "wind_speed_scale"
[19] "precipitation_during_visit"   "precipitation_last_48_h"
[21] "temperature_last_48_h"        "weather_condition_notes"
[23] "relative_humidity_percent"    "wind_speed_min_m_s"
```

```
[25] "wind_speed_max_m_s"           "air_temp_c_drop"
[27] "densiometer_d1_num_covered"   "d1_n"
[29] "d1_s"                         "d1_e"
[31] "d1_w"                         "d1_percent_cover"
[33] "densiometer_d2_num_covered"   "d2_n"
[35] "d2_s"                         "d2_e"
[37] "d2_w"                         "d2_percent_cover"
[39] "depth_of_water_from_d2_cm"    "vegetation_cover_percent"
[41] "vegetation_notes"             "secchi_depth_cm"
[43] "conductivity_us_cm"           "fish"
[45] "comments_environmental"       "environmental_utmn"
[47] "environmental_utme"           "environmental_utm_zone"
[49] "environmental_elevation_m"    "environmental_latitude"
[51] "environmental_longitude"      "air_pressure_mbar"
```

```r
# join with survey and visit tables,
db_env_visit = db_env %>%
  left_join(db_survey, by = "survey_id") %>%
  left_join(db_visit, by = "visit_id") %>%
  group_by(visit_id) %>%  # aggregate to a single value per visit
  summarise(env_n = n(),
            air_temp_c_mean = mean(air_temp_c, na.rm = TRUE),
            water_temp_c_mean = mean(water_temp_c, na.rm = TRUE))

# left_join db_bd_country with db_env_visit, select desired columns
db_bd_env = db_bd_country %>%
  left_join(db_env_visit, by = "visit_id") %>%
  select(sample_name_bd,
         detected,
         average_target_quant,
         taxon_capture,
         svl_mm,
         body_mass_g,
         sex,
         life_stage,
         substrate_temp_c,
         air_temp_c_mean,
         water_temp_c_mean,
         env_n,
         microhabitat_type,
         time_of_capture,
         date,
         site,
         region,
         country,
```

```
          visit_lab,
          extraction_lab,
          qpcr_lab)
```

Exploring and filtering

```
# count the number of rows in our data table so far
db_bd_env %>%
  count()
```

```
# Source:   SQL [?? x 1]
# Database: postgres  [cob_reads@ribbitr.c6p56tuocn5n.us-west-1.rds.amazonaws.com:5432/ribbi
        n
  <int64>
1   62750
```

```
# group by region & country, and count each combination
summary_bd_region = db_bd_env %>%
  group_by(life_stage) %>%
  count() %>%
  collect()

# filter to adult life stage (dropping all others)...
db_adult = db_bd_env %>%
  filter(life_stage == "adult",
         !is.na(substrate_temp_c))

# ... and count again
db_adult %>%
  count()
```

```
# Source:   SQL [?? x 1]
# Database: postgres  [cob_reads@ribbitr.c6p56tuocn5n.us-west-1.rds.amazonaws.com:5432/ribbi
        n
  <int64>
1    8711
```

```
# other ways we might want to group and summarize this data to explore it:
# group by region & country, and count each combination
summary_bd_region = db_adult %>%
  group_by(country, region) %>%
```

```
  count() %>%
  collect() %>%
  arrange(country, region)

# define new variable `year`, group by year, count by year
summary_bd_year = db_adult %>%
  mutate(year = year(date)) %>%
  group_by(year) %>%
  count() %>%
  collect() %>%
  arrange(year)
```

Data pulling

Collect these data!

Now that we are happy with these data, or at least want to take a closer look at them, lets collect them from the database.

```
# take a look at our SQL "shopping list":
db_adult_query = sql_render(db_adult)

# take our shopping list to the store, return with the data we want
data_adult = db_adult %>%
  collect()

# peek in the bag to see what you got :)
head(data_adult)
```

```
# A tibble: 6 x 21
  sample_name_bd  detected average_target_quant taxon_capture svl_mm body_mass_g
  <chr>           <lgl>                  <dbl> <chr>          <dbl>       <dbl>
1 181015-36-TN05~ FALSE                      0 ambystoma_op~   49           4.8
2 170412-17-VT02~ FALSE                      0 rana_sylvati~   48          14
3 180204-08-LA02~ FALSE                      0 acris_blanch~   21.4         0.7
4 180709-33-TN04~ FALSE                      0 hyla_chrysos~   40           5.1
5 180403-40-PA12~ FALSE                      0 rana_sylvati~   NA          12.8
6 180514-07-TN04~ FALSE                      0 lithobates_s~   54.7        10
# i 15 more variables: sex <chr>, life_stage <chr>, substrate_temp_c <dbl>,
#   air_temp_c_mean <dbl>, water_temp_c_mean <dbl>, env_n <int64>,
#   microhabitat_type <chr>, time_of_capture <time>, date <date>, site <chr>,
#   region <chr>, country <chr>, visit_lab <chr>, extraction_lab <chr>,
#   qpcr_lab <chr>
```

Additional exploration: Taxonomy

We can continue using our table pointer "shopping list" to pull in other data. For example, suppose we want some information on the taxa in our dataset:

```
# count distinct taxa
taxa_relevant = db_adult %>%
  group_by(taxon_capture) %>%
  summarise(taxa_count = n()) %>%
  left_join(db_taxa, by = c("taxon_capture" = "taxon_id")) %>%
  collect() %>%
  arrange(desc(taxa_count))

# filter to taxa with more that 50 counts
taxa_gt50 = taxa_relevant %>%
  filter(taxa_count >= 50)

# filter our data to those taxa in our list
db_adult_gt50 = db_adult %>%
  filter(taxon_capture %in% taxa_gt50$taxon_capture)

# collect these newly filtered data
data_adult_gt50 = db_adult_gt50 %>%
  collect()
```

Get query metadata

Want to refine the column metadata to include columns for your recently queried data only?

```
# use the ribbitrrr::get_query_metadata() function
metadata_adult_gt50 = ribbitrrr::get_query_metadata(dbcon, db_adult_gt50)
```

Additional exploration: Collaborators!

```
# create a list of unique labs associated with your data
labs_associated = unique(c(data_adult_gt50$visit_lab,
              data_adult_gt50$extraction_lab,
              data_adult_gt50$qpcr_lab))

# look at columns in the labs table for context
colnames(db_lab)
```

```
[1] "lab_id"           "lab_name"         "lab_data_contact"
```

```
# filter to
collaborators = db_lab %>%
  filter(lab_id %in% labs_associated) %>%
  collect()
```

Disconnect

When we are done, we can disconnect from the database to reduce server load. You can also disconnect using the connections panel.

```
dbDisconnect(dbcon)
```

<- 6. Microclimate Workflow |