Connecting to the FIM SQL Database: Connecting to a restored SQL backup from R

Best Option: Connect to SQL but then use R to query and analyze

This allows you to set up multiple connections, for each database

If you have datasets in same database but different schema, the "in_schema" function becomes very helpful when pulling data

This option allows you to filter/process data remotely before "collecting" it to your computer – if you can do your large query steps on the server, I recommend it before "collecting" data to your local machine. Just keep in mind that in order to query data on the server, the R code has to be translated (in the background) to SQL language (this is what the dbplyr package is for), so not all functions are able to be translated.

Therefore, sometimes it is beneficial to "collect" more to R and then process using R language.

The following example shows how to connect to the database and then use R to query and analyze. It is reproduced in the R script "SQL_from_R_SampleScript.R" for easier integration into your R workflow.

```
# Load packages
library(tidyverse)
library(dbplyr)
library(odbc)
library(lubridate)

# Connect to database
Corp <- dbConnect(odbc::odbc(),
                  driver = "SQL Server",
                  server = "", # Enter your local server path here
                  database = "FIMCorpInshore",
                  trusted_connection = TRUE)

# Pull physical data
# The tbl function and in_schema function can be used to select a table
# from the database
physmast <- tbl(Corp, in_schema("hsdb","tbl_corp_physical_master")) %>%
  # The collect function pulls the dataset from the database into your r
  # environment as a tibble
  collect()

# Pull biological data
Bio <- tbl(Corp, in_schema("hsdb","tbl_corp_biology_number")) %>%
# You can use tidyverse dplyr functions to wrangle data prior to pulling it
# into your environment (from the dbplyr package)
```

```
select(Reference,
       Species_record_id,
       Splittype,
       Splitlevel,
       Cells,
       NODCCODE,
       Number,
       FHC) %>%
filter(FHC != "D") %>%
mutate(Count = case_when(!is.na(as.numeric(Splittype)) ~
                         Number*(as.numeric(Splittype)^as.numeric(Splitlevel)),
                         TRUE ~ as.numeric(Number))) %>%
select(Reference, Species_record_id, NODCCODE, Count) %>%
group_by(Reference, NODCCODE) %>%
mutate(N = sum(Count)) %>%
# You can also join data from another table in the database
left_join(tbl(Corp,
              in_schema("hsdb","tbl_corp_ref_species_list")),
              by = "NODCCODE") %>%
select(Reference, Species_record_id, NODCCODE, Scientificname, N) %>%
arrange(Reference, NODCCODE) %>%
collect()
```