CalCOFI.io Docs

2024-09-10

Table of contents

1 Process

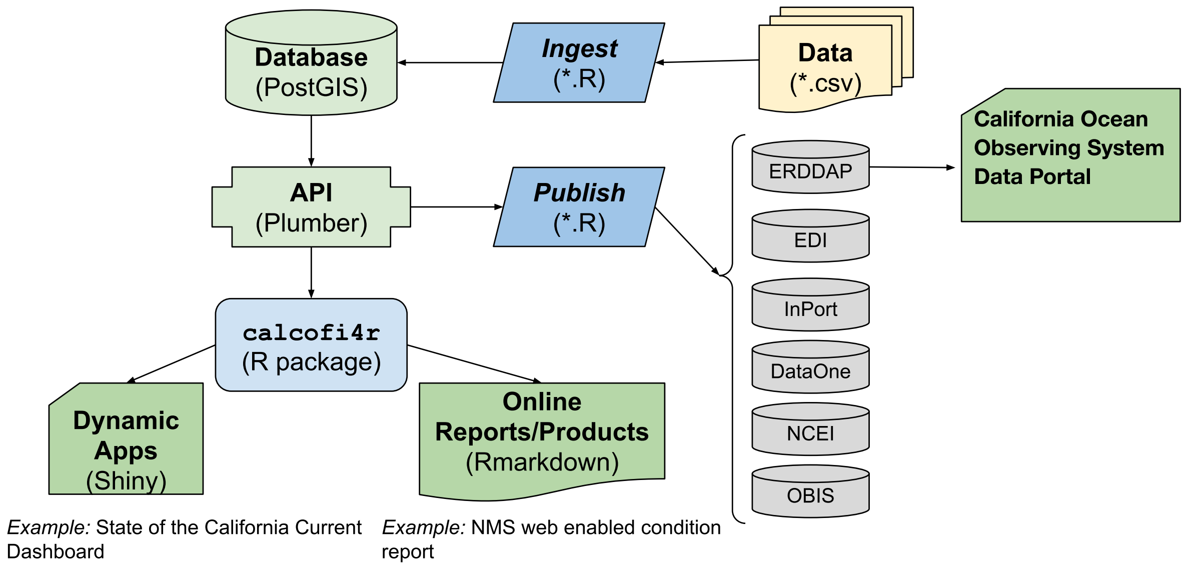

Figure 1.1: CalCOFI data workflow.

The original raw **data**, most often in tabular format [e.g., comma-separated value (*.csv)], gets **ingest**ed into the **database** by R scripts that use functions and lookup data tables in the R package `calcofi4r` where functions are organized into *Read*, *Analyze* and *Visualize* concepts. The application programming interface (**API**) provides a program-language-agnostic public interface for rendering subsets of data and custom visualizations given a set of documented input parameters for feeding interactive applications (**Apps**) using Shiny (or any other web application framework) and **reports** using Rmarkdown (or any other report templating framework). Finally, R scripts will **publish** metadata (as Ecological Metadata Language) and data packages (e.g., in Darwin format) for discovery on a variety of data ***portals*** oriented around slicing the tabular or gridded data (ERDDAP), biogeographic analysis (OBIS), long-term archive (DataOne, NCEI) or metadata discovery (InPort). The **database** will be spatially enabled by PostGIS for summarizing any and all data by ***Areas of Interest*** (AoIs), whether pre-defined (e.g., sanctuaries, MPAs, counties, etc.) or arbitrary new areas. (Figure 1.1)

- ERDDAP: great for gridded or tabular data, but does not aggregate on the server or clip to a specific area of interest

2 Reports

2.1 Sanctuaries

- Channel Islands WebCR
  web-enabled Condition Report
  - Forage Fish
    example of using calcofi4r functions that pull from the API
- UCSB Student Capstone

3 Applications

- CalCOFI Oceanography
  oceanographic summarization by arbitrary area of interest and sampling period
- UCSB Student Capstone

4 API

The raw interface to the Application Programming Interface (API) is available at:

- api.calcofi.io

Here we will provide more guidance on how to use the API functions with documented input arguments, output results and examples of use.

4.1 `/variables`: get list of variables for timeseries

Get list of variables for use in `/timeseries`

4.2 `/species_groups`: get species groups for larvae

Not yet working. Get list of species groups for use with variables `larvae_counts.count` in `/timeseries`

4.3 `/timeseries`: get time series data

4.4 `/cruises`: get list of cruises

Get list of cruises with summary stats as CSV table for time (`date_beg`)

4.5 `/raster`: get raster map of variable

Get raster of variable

4.6 `/cruise_lines`: get station lines from cruises

Get station lines from cruises (with more than one cast)

4.7 `/cruise_line_profile`

Get profile at depths for given variable of casts along line of stations

5 Database

5.1 Database naming conventions

- Learn SQL: Naming Conventions
- Best Practices for Database Naming Conventions - Drygast.NET

5.1.1 Name tables

- Table names are plural and use all lower case.

5.1.2 Name columns

- To name columns, use **snake-case** (i.e., lower-case with underscores) so as to prevent the need to quote SQL statements. (TIP: Use `janitor::clean_names()` to convert a table.)
- Unique **identifiers** are suffixed with:
  - `*_id` for unique integer keys;
  - `*_key` for unique string keys;
  - `*_seq` for auto-incrementing sequence integer keys.
- Suffix with **units** where applicable (e.g., `*_m` for meters, `*_km` for kilometers, `degc` for degrees Celsius). See units vignette.
- Set geometry column to **`geom`** (used by PostGIS spatial extension). If the table has multiple geometry columns, use `geom` for the default geometry column and `geom_{type}` for additional geometry columns (e.g., `geom_point`, `geom_line`, `geom_polygon`).

5.2 Describe tables and columns

- Use the `COMMENT` clause to add descriptions to tables and columns, either through the GUI pgadmin.calcofi.io (by right-clicking on the table or column and selecting `Properties`) or with SQL. For example:

```
COMMENT ON TABLE public.aoi_fed_sanctuaries IS 'areas of interest (`aoi`) polygons for f
```

- Note the use of **markdown** for including links and formatting (e.g., bold, code, italics), such that the above SQL will render like so:

  > areas of interest (`aoi`) polygons for federal **National Marine Sanctuaries**; loaded by *workflow* load_sanctuaries

- It is especially helpful to link to any ***workflows*** that are responsible for the ingesting or updating of the input data.

- These descriptions can be viewed in the CalCOFI **API** api.calcofi.io as CSV tables (see code in calcofi/api: `plumber.R`):

  – api.calcofi.io/`db_tables`
    fields:

    * `schema`: (only "public" so far)
    * `table_type`: "table", "view", or "materialized view" (none yet)
    * `table`: name of table
    * `table_description`: description of table (possibly in markdown)

  – api.calcofi.io/`db_columns`
    fields:

    * `schema`: (only "public" so far)
    * `table_type`: "table", "view", or "materialized view" (none yet)
    * `table`: name of table
    * `column`: name of column
    * `column_type`: data type of column
    * `column_description`: description of column (possibly in markdown)

- Fetch and display these descriptions into an interactive table with `calcofi4r::cc_db_catalog()`.

5.3 Relational Database Structure

5.4 Spatial Tips

- Use `ST_Subdivide()` when running spatial joins on large polygons.

6 References

6.1 R packages

- API: plumber (Schloerke and Allen 2024)
- docs: Quarto (Allaire and Dervieux 2024)
- apps: Shiny (Chang et al. 2024)

Allaire, JJ, and Christophe Dervieux. 2024. *Quarto: R Interface to Quarto Markdown Publishing System*. https://github.com/quarto-dev/quarto-r.

Chang, Winston, Joe Cheng, JJ Allaire, Carson Sievert, Barret Schloerke, Yihui Xie, Jeff Allen, Jonathan McPherson, Alan Dipert, and Barbara Borges. 2024. *Shiny: Web Application Framework for r*. https://shiny.posit.co/.

Schloerke, Barret, and Jeff Allen. 2024. *Plumber: An API Generator for r*. https://www.rplumber.io.