CalCOFI.io Docs

2024-09-09

Table of contents

1 Process

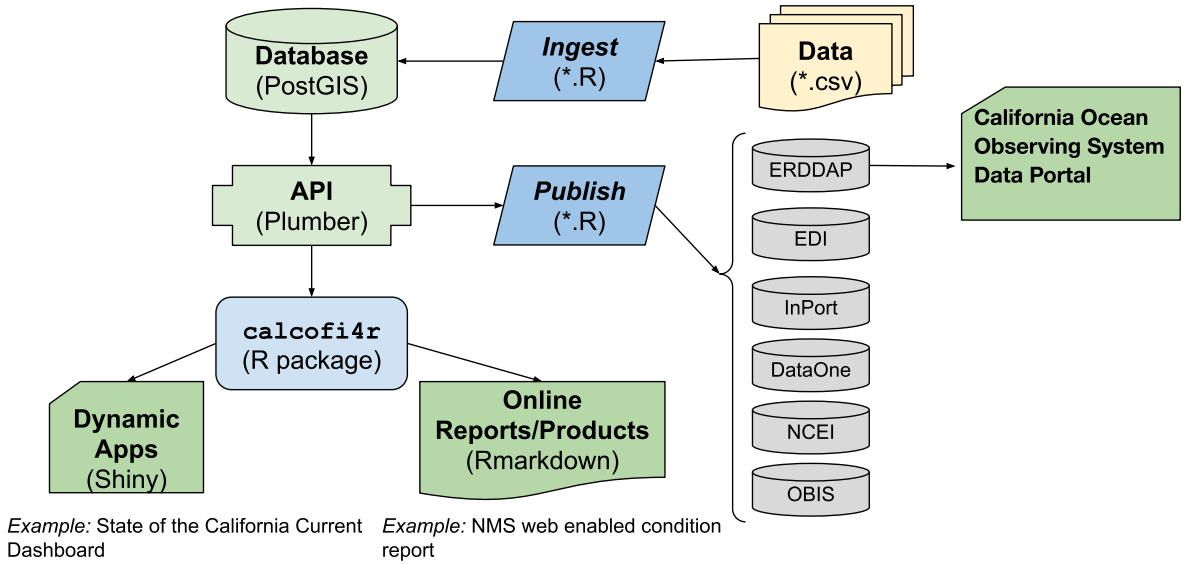

Figure 1.1: CalCOFI data workflow.

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 Apps

- 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 Relational Database Structure

5.1.1 Typography

- `{*}`: indicates variable substitution, e.g. `{mdl_key}_mdls` would evaluate to the value `am_mdls` for `mdl_id = "am"` (AquaMaps)
- `[*]`: optional value, such as `[ply_grp]` is an optional column in the `{mdl_key}_mdls` table
- `<*>`: surrounds the columns used to uniquely identify (and index) each row
- `...`: additional columns, unique to the table

The format below is of the following format where the top line of a bulleted list item describes the table and the columns in that table are directly below, nested in hierarchical order:

- `{table name}` (`{description}`)
  <`{column 1}`, `{column 2}`>, `{column 3}`, `...`

5.1.2 Database Naming Conventions

- Use all **lower-case** column names with underscores (i.e. from using `janitor::clean_names()`) to prevent need to quote SQL statements.
- For short unique **identifiers** use suffix `*_id` for integer and `*_key` for short text.

5.2 Spatial Tips and Conventions

- Set PostGIS geometry fieldname to `geom`.
- 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.