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](https://github.com/CalCOFI/scripts) that use functions and lookup data tables in the R package [**calcofi4r**](https://calcofi.github.io/calcofi4r/reference/index.html) 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](https://docs.ropensci.org/EML)) 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](https://coastwatch.pfeg.noaa.gov/erddap/information.html)), biogeographic analysis ([OBIS](https://obis.org)), long-term archive ([DataOne](https://www.dataone.org), [NCEI](https://www.ncei.noaa.gov)) or metadata discovery ([InPort](https://www.fisheries.noaa.gov/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](#fig-sw-arch))

* 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](https://noaa-onms.github.io/cinms)  
  web-enabled Condition Report
  + [Forage Fish](https://noaa-onms.github.io/cinms/modals/forage-assemblage.html)  
    example of using calcofi4r functions that pull from the API
* [UCSB Student Capstone](https://shiny.calcofi.io/capstone)

# 3. Apps

* [CalCOFI Oceanography](https://shiny.calcofi.io/oceano)  
  oceanographic summarization by arbitrary area of interest and sampling period
* [UCSB Student Capstone](https://shiny.calcofi.io/capstone)

# 4. API

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

* [api.calcofi.io](https://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()](https://postgis.net/docs/ST_Subdivide.html) when running spatial joins on large polygons.

# 6. References

## 6.1 R packages

* API: plumber ([Schloerke and Allen 2024](#ref-R-plumber))
* docs: Quarto ([Allaire and Dervieux 2024](#ref-R-quarto))
* apps: Shiny ([Chang et al. 2024](#ref-R-shiny))

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>.