FAOSTAT 3.0

A revitalisation of a wrapper for the FAOstat API

Sebastian Campbell

March 1, 2024

The FAOSTAT package is an important part of the Food and Agriculture Organization (FAO)'s image that is being maintained, but requires a makeover. Here an updated version, 2.3.0, of the package is presented with repaired access to FAO's API, new functions and preparation for a complete overhaul in 3.0.0. The package has been modernised according to new coding conventions with improved dependencies, documentation and tests. Old useless functions have been pruned and the package is now firmly focused on providing an interface to FAO data to users of R.

> **!** Important
>
> The abstract needs to be updated to 3.0

Other formats

HTML report | PDF report | Presentation | GitHub

Project background

The motivation for this project came from a Data Mining project from UniLaSalle. It was suggested that we use FAO[1] data from their statistical platform FAOstat[2]. As R was the language of

[1] Food and Agriculture Organization of the United Nations

[2] Food and Agriculture Organization Corporate Statistical Database

choice, the obvious port of call was the `FAOSTAT` package[3] (Kao, Gheri, and Gesmann 2022), developed by employees at FAO.

However, the `FAOSTAT` package did not work. It could not download data from the API[4] and could only download bulk data with the entirety of a dataset in one go. For the particular dataset we were interested in, we found that there was a discrepancy between the data in the bulk download and the data on the web platform.[5]

Eventually it became necessary to use the same API that the FAOstat website uses to pull data. This method worked and it became clear that it could be used to revitalise the `FAOSTAT` package and part of an effort to restore it to full functionality.

A first update, `FAOSTAT` 2.3.0 was released by the author in May 2023. `FAOSTAT` 3.0 is a continuation of that effort and realises many of the goals that it set out.

[3] For the purposes of clarity, this document will use the style `FAOSTAT` in monospace for the R package[1] and "FAOstat" for the statistical platform

[4] Application Programming Interface

[5] This discrepancy has been fixed as of 2023-03-10

What is an API?

An API[6] is a method for a host application to allow a client to send it commands in a way the client understands. The API is a layer that converts these client commands to ones that the host understands. To use a metaphor, you could in theory drive a car by manually turning the axle to got left and start it by hitting a spark plug yourself. However, the key (or start button, if you have a newer model) and steering wheel are a much more natural and intuitive way to direct a car for a human being. However, from the car's perspective the axle gets turned and the spark plug is struck. The steering wheel and key essentially work as an API between the server (the car) and the client (the person).

In modern parlance, when we refer to an API, we usually mean a REST[7] API. This is a type of API run over the web via HTTP[8], the same protocol that serves web pages. These APIs are a method of exchanging data and use the following major verbs[9] from HTTP:

GET - Request data from the server

- Request: `GET https://example.com/johnsmith/info`
- Payload: None

[6] Application Programming Interface

[7] REpresentative State Transfer

[8] HyperText Transfer Protocol

[9] There are other verbs (HEAD, CONNECT, OPTIONS, TRACE and PATCH), but they are of much lower importance

- Response: `{username: johnsmith, full_name = "John Smith"}`

PUT - Send data to the server

- Request: `PUT https://example.com/johnsmith/info`
- Payload: `{full_name: "John Allen Smith"}`
- Response: None

POST - Send data to the server and receive data

- Request: `POST https://example.com/johnsmith/send_email`
- Payload: `{subject: "Business opportunity", body = "Dear {fullname}, please reply"}`
- Response: `{result: "success", send_time: "2020-09-23 17:34:15"}`

DELETE - Delete data from the server

- Request: `DELETE https://example.com/johnsmith`
- Payload: None
- Response: None

The key innovation that REST introduces is *statelessness*. The server does not have to keep track of any previous requests. Each request is atomic and is considered in isolation. This distinguishes it from other API patterns such as SOAP[10] (Halili and Ramadani (2018)).

[10] Simple Object Access Protocol

FAOstat

FAOstat is FAO's web-based statistical platform for the free dissemination of food and agriculture statistics. This data is obtained from questionnaires that FAO distributes throughout the world every year (Food and Agriculture Organization of the United Nations 2019). Some of its data also comes from imputations and models where data is not available, but official country data takes precedence.

The FAOstat service is a public-facing aspect of FAO, with an overall trend of increasing citations in academic papers year on year with 21400 citations by 2021 (Figure 1).

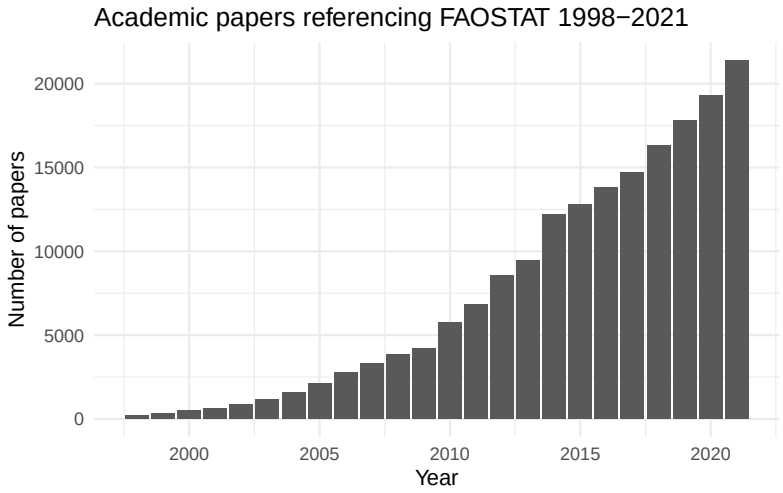

Figure 1: Academic papers referencing FAOSTAT over the last 25 years (Strobel 2018)

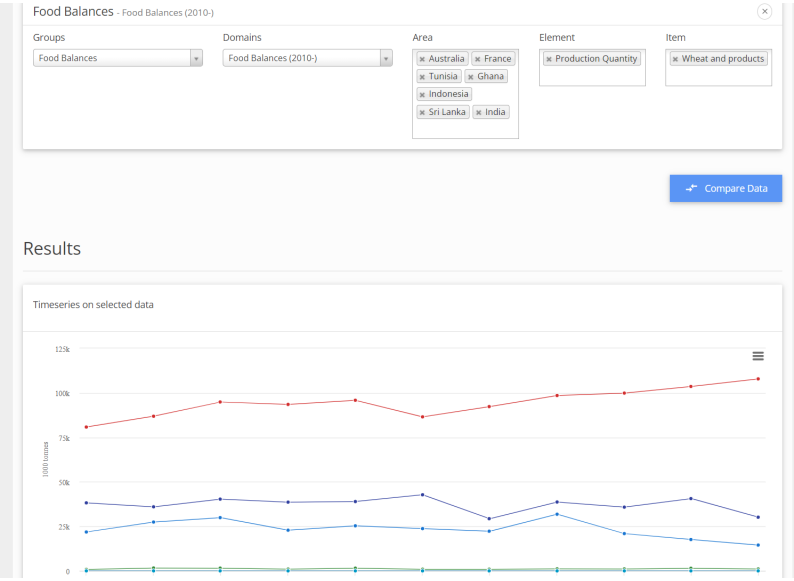

Figure 2: FAOstat interface for exploration of country data

This platform uses a REST API internally to communicate with its database as well as providing a set of zip files with the entirety of certain datasets in order to reduce the load on the database. This REST API allows the website to generate CSVs as well as to allow exploration of the data via interactive graphs (Figure 2).

`FAOSTAT` package

The `FAOSTAT` package is an API wrapper to pull data from FAOstat into a R session. It can also perform small necessary tasks such as country code conversion and coalescing data from different country groups.[11]

[11] For example, China may be just the mainland or may include Taiwan (Chinese Taipei), Hong Kong and Macao

History

The `FAOSTAT` package was originally developed in 2013 as a tool to source data for the SYB[12] project. The yearbooks are yearly summaries of the worldwide state of agriculture for that year. At the time, they were manually typeset and compiled. The new SYB project was to use a combination of `LaTeX`, `knitr` and `R` to automatically pull data from FAOstat and other data sources such as the World Bank. This data would be then be transformed and processed to create graphs and tables before finally formatting and typesetting to create a finished product which could then be printed.[13]. Given that this use case no longer exists, the primary use of this package is for researchers and other R users to read data from FAOstat in a clean way that makes it easier to move to analysis afterwards.

[12] Statistical Year Book

[13] The author has no insight into the current production of the SYB, but they are still being produced and can be found on the FAO website

It is a reasonably popular package; in the 86th percentile of all packages on CRAN on 2023-04-01 by downloads. In total, the package has been downloaded over 50 000 times with a peak 121 daily downloads on 2019-05-15. (Li 2023)

Maintainership

The package was maintained by Michael Kao, the author, from 2013 to 2014. In 2014, it was maintained by Filippo Gheri be-

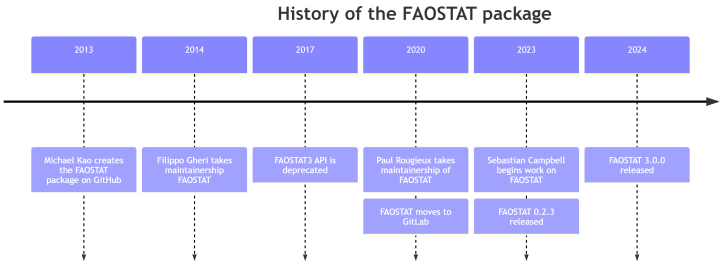

Figure 3: A timeline of the `FAOSTAT` package from its creation to today

fore passing to Paul Rougieux (the current maintainer) in 2020 (Figure 3).

While it was originally hosted on Github under Michael Kao's personal account, It is currently hosted on GitLab under Paul Rougieux's personal account.

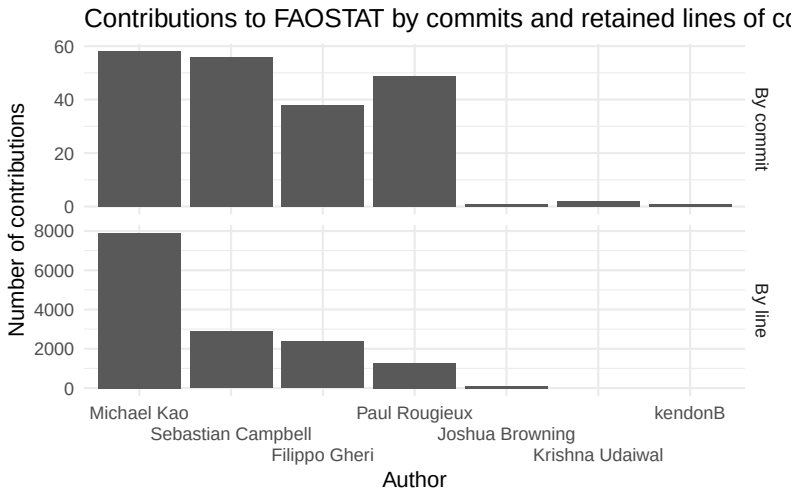

Figure 4: Contributions to the `FAOSTAT` git repository by number of commits and by number of lines contributed that remain in the current master branch. Snapshot taken on 2024-02-26

While there have been other contributors, they have only provided occasional commits and very little of that code remains in the codebase (Figure 4). The major four maintainers (Kao, Gheri, Rougieux and Campbell) are by far the main contributors.

The package's development can be split into two major periods. Between 2013 and 2017 (Figure 5), the package is focused on the Statistical Yearbooks and the FAOSTAT3 API. However, as the package is no longer used for the yearbook and the API is discontinued, the focus is changed to the bulk download functionality to download whole datasets at once.

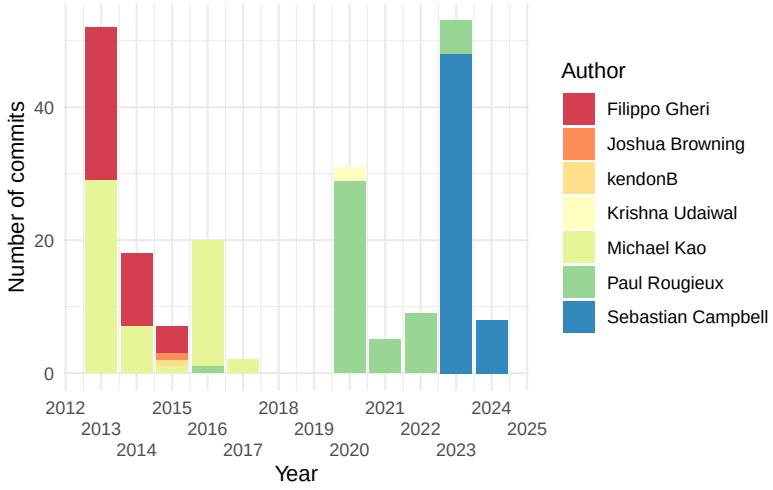

Figure 5: Contributions to the `FAOSTAT` git repository over time by author. Snapshot taken on 2024-02-26

API structure

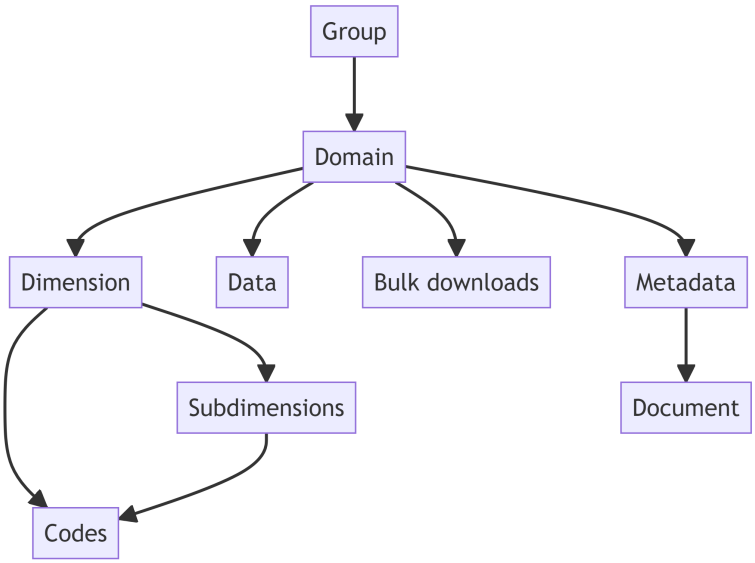

Figure 6: A broad structure of the FAOstat API package

The API is has two main components: dimensions and data (Figure 6). Domains are the objects that hold data itself and these are clustered into groups. Groups link thematically connected domains. These groups correspond to the various internal teams within FAO that are responsible for disseminating data.[14] Dimensions and subdimensions for the definitions for the shape of the data. These are defined separately and can be applied to multiple

[14] Personal communication, 2024

domains. In addition there are utility endpoints:

- `/ping` - simple check to see if the server is up
- `/datasize` - Appended to the end of queries, returns the size of the data

Identified problems

The `FAOSTAT` package has only a shadow of its former functionality. While it has retained the ability to download and process zip files (bulk downloads) and country code processing functions,[15] its capacities are limited by the following issues:

[15] For a full description of the status of individual issues, please see the GitLab issue #20 Remove functions linked to defunct uses of FAOSTAT

Functionality locked to the Statistical Yearbook

A number of functions are simply designed to pull in data from other sources such as the World Bank and to process that data into a format easily consumed by the Statistical Yearbook. As the yearbook no longer uses the `FAOSTAT` package, these functions have no further purpose, serving only to clog up the package and its help files.

Functionality powered by local files

Many uses of FAOSTAT require data outside of the data that comes directly from FAOstat The major use case is for code conversions. There are two main code types that require conversion:

- Country codes
  - **FAO**: FAO's internal codes for countries[16]
  - **M49**: The UN standard country codes
  - **ISO2 & ISO3**: 2 and 3 letter country codes
- Item codes
  - **FAO**: Internal FAO codes to describe commodities
  - **CPC**: Central Product Classification code (United Nations Statistical Division 2015)

[16] For further details about FAO and how it handles country identification, see FAO's NOCS database

The conversions are not dynamically taken from the API but rather stored in a fixed file. This makes them vulnerable to code or name changes in the future such as the name change of Swaziland to Eswatini in 2018 (United Nations 2018).

Inactive API

The `FAOSTAT` package is currently configured to access a now-defunct API[17] (FAOSTAT3). As a result, it has no methods of retrieving data from the FAOstat platform with the sole exception of the bulk zip downloads which have been since adapted to use the current platform.

[17] Originally hosted at faostat3.fao.org

Limited extensibility for queries

When making a query using the API, the request *must* contain exactly four dimensions (area, item, element and year). This means that it cannot be used for domains that do not meet these strict design requirements. For example, the GFDI[18] domain has 6 dimensions (area, category, industry, factor, element, year)

[18] Value shares by industry and primary factors

Stewardship

The `FAOSTAT` package is currently maintained by Paul Rougieux who has done an excellent job of keeping the package afloat. However, it is a small project done in his free time, so he has not been able to make the time to do a full overhaul. In addition, he is not an employee of FAO, but rather of the European Commission. As a face that FAO shows to the world, it seems reasonable that it be placed under its guidance.

Obsolete dependencies

As the package is old, it also has a number of dependencies that are unused or will be, after redundant functions are removed. Other dependencies are simply no longer developed or have been superseded by newer packages. This means that installing the

packages automatically installs a number of other packages that are of no use, increase install time, space used and being generally inelegant.

Project goals

There are four main goals of this project:

- Fix core functions
- Triage existing functions
- Characterise relevant aspects of the new API to wrap
- Transfer maintainership

Fix core functions

The first priority to restore the `FAOSTAT` package is to repair the most basic and most used functions. These are the functions that work directly with the API and that provide data to the user.[19] After this point, functions should not only be repaired but also renamed to have a consistent naming system. The existing functions *do* have a naming convention (Camel case, consistent use of verbs) but it doesn't reflect the terminology used in the API and there's no internal hierarchy to naming.

[19] This was completed in `FAOSTAT` 2.3.0

Triage existing functions

As described in the Current state section, many functions no longer work and need to be removed. This involves a thorough listing of them and their functionality and assessment of their usefulness. A full analysis was completed for the release of `FAOSTAT` 2.3.0 and is available on GitLab.[20]

[20] GitLab issue #20

Characterise aspects of the new API to wrap

Inspecting the API to find all relevant endpoints was completed with `FAOSTAT` version 2.3.0 released in March 2023.[21] This re-

[21] For a full set of changes expected for `FAOSTAT` 3.0.0, see the GitLab milestone page

quires the additional step of inspecting the API manually as it does not have complete documentation.

Transfer maintainership

Uploading a package on CRAN[22] requires the permission of the maintainer. At the upload step, an email is sent to the maintainer of the last recorded package version to confirm that the upload was authorised. Only then can the new version proceed to the registration step. `FAOSTAT` 2.3.0 was released via this process, but maintainership needs to be transferred to do it with full autonomy.

[22] Comprehensive R Archive Network, the centralised repository where the majority of R users download R packages

Methods & Discussion

On 2023-03-30, the `FAOSTAT` package version 2.3.0 was published to CRAN.[23] The process required the resolution of the issues detailed in the Project Goals section.[24]

In order to solve these issues, a full examination of the package context, the API and a redesign of existing functions needed to be made. These efforts permitted the eventual release of `FAOSTAT` 3.0.0.

[23] CRAN package page

[24] Please see GitLab Milestone 2.3.0 for a full list of related issues

Solutions to Identified problems

In the above Identified Problems section, we described the following issues:

- Functionality locked to the Statistical Yearbook
- Functionality powered by local files
- Inactive API
- Limited extensibility for queries
- Stewardship
- Obsolete dependencies

Functionality locked to the Statistical Yearbook

Of all the functions in the `FAOSTAT` package, we identified eleven functions that we inextricably linked to the Statistical Yearbook. They were either linked to text formatting (e.g. `printLab`) or obtaining auxiliary data from third party sources like the World Bank (e.g. `getWDI`). These were all identified[25] and removed[26].

For the most part, they could be easily identified by the use of "SYB" in the function name or reference to non-FAO sources like the World Bank. In more advanced cases, we would simply examine the functions for reference to the FAOstat API and discard all functions that did not and were not called by any functions that *did* use it.

[25] GitLab issue #20

[26] Commit e1f0a910

Functionality powered by local files

The following data is stored in local files:

- Country code conversions. Links between country codes and their M49, ISO 2-character[27] and internal FAO codes (`FAOcountryProfile`)
- Code descriptions and the domains to which those codes apply (e.g. `Q` is the code in the API for the Production domain) (`FAOmetaTable`)
- Region conversions. Similar to *Country code conversions*, but gives correspondences between regions (e.g West Asia, Developing countries) and countries (`FAOregionProfile`)

[27] For more information, see the International Standards Organization glossary on country codes

These data (with the exception of ISO 2-character codes) are all obtainable from the API itself (the various `/dimensions` endpoints), so it does not make any sense to retain these. Country code conversion beyond converting between formats used by the API is not in the scope of this package. The recommended way to convert country codes for the purpose of analysis is to use the excellent `countrycode` package (Arel-Bundock, Enevoldsen, and Yetman (2018)). Not only can this package convert between codes, it is also capable of using regular expressions on country names, converting full text country names into the appropriate codes.

Inactive API

The FAOstat3 API ceased functioning in 2017 and was replaced by the API that is currently being used to feed the the FAOstat data explorer.

The new API however, wasn't designed for other applications to consume it. This manifested in two ways:

1. Lack of of clear documentation
2. Hesitancy to encourage the use of the API for broad consumption

The API is not fully documented as the only official consumer of the API is the FAOstat web application itself. Third party development isn't in scope for the current project cycle. The main documentation is a JSON:API specification,[28] which describes the structure perfectly, but gives no context or recommendations for use, nor does it describe output. This is intentional as the servers which provide the API are not equipped to handle the load that a publicly available API would inflict. In one instance, the response would be cut off half-way through with no warning or error as the server was overloaded.[29]

[28] JSON:API is a standard way to describe the structure of an API as a single json document

[29] Personal communication, 2023

It was thus necessary to use this document to manually examine all the endpoints and characterise all the the data output from them.[30]

[30] A full set of this examination is found in GitLab issue #23

As a result, we have created the most accessible documentation for the FAOstat API. We have also negotiated that the development of this API package continue. The main arguments are that R is mainly used for scripts by individual users rather than for automated routines in a production environment (Morandat et al. (2012)), so it's unlikely that there will be any issues. Furthermore, a server upgrade occurred in February 2024, increasing their capacity to handle additional requests.

Limited extensibility for queries

The limited extensibility of `FAOSTAT` manifests in two ways:

1. The aforementioned locally stored files

2. The inflexibility of its functions

This inflexibility causes problems for unusual data structures or changes to codes in FAOstat. Essentially, any changes that are made could cause complete breakage for certain requests, for example the update to the CPC codes in 2015 (United Nations Statistical Division (2015)). Fixing this issue requires us to depend more heavily on the API.

We solved this issue by using the same endpoints that the API uses to convert codes when providing them to the user for data view or for downloads. The API has `/definitions` endpoints that contains all of the conversions that exist for all codes. This also means that we can be internally consistent with country codes and not have to rely on third-party packages such as `countrycode` to perform conversions for us.[31]

[31] `countrycode` would only work, as the name suggests, for country codes and wouldn't be helpful for other kinds of codes such as Commodity codes

Stewardship

The fact that the API is not managed by someone with links to FAO is not a problem in itself, but it is suboptimal for the purposes of solving technical issues that depend on edge cases or implementation details of FAOstat. It further means that if FAO were to wish to promote this package to third parties, it wouldn't be able to decide any priorities in regards to its development in terms of bug fixes or new features.

The author has worked as a consultant for FAO and is currently in contact with the previous maintainers of `FAOSTAT` as well as developers who are currently working on the FAOstat API. The transfer of maintainership is however, as of the time of writing, incomplete.

Obsolete dependencies

Obsolete dependencies result from three major sources. The packages attached to each point are dependencies that have been removed from `FAOSTAT`[GitLab issue #6] [Commit a1c15150]:

1. Packages which were once used and are no longer used even before the author commenced work on the package

- `ggplot2 (>= 0.9.3)`

2. Packages which have been largely superseded by more modern packages

- `plyr (>= 1.7.1)`
    - Not only have its capabilities been surpassed by `dplyr`, `FAOSTAT` also depends on `data.table` which covers many of the same data manipulations tasks.[32]

3. Packages which are now obsolete because the functions that relied on them were removed as part of the deletion of functionality locked to the Statistical Yearbook

- `RJSONIO (>= 0.96-0)`
    - No longer required as `httr` which we are using now for requests uses the more modern `jsonlite`
- `classInt (>= 0.1-19)`
- `labeling (>= 0.1)`

The use of unit tests ensured there was no regression after removing these dependencies. The origin of these dependencies was difficult to track down as they were not added over time. Rather, they were all added in the first commit of the project.[33] The conventional wisdom is to "Commit early, commit often." This makes these changes granular, so it is easier to tease apart the individual changes in any commit (Eloe (2021)).

[32] And is, in the opinion of the author, a generally superior package

[33]

Learning package context

The two most important developers of the `FAOSTAT` package are the package author, Michael Kao and the current maintainer, Paul Rougieux.

Paul Rougieux has been very helpful in guiding the author's proposed changes to the package - making excellent suggestions in regards to code style and formatting as well as documentation and general usability.

An interview with the Michael Kao[34] confirmed that a number of of unusual functions were due to its previous linkage with the

[34] Personal Communication, 2023

Statistical Yearbook and that the paper attached to the package had never been published and suggested that it should be tidied up and published in future.

Redesigning functions

The existing functions to be refactored to read data from the new API. `FAOSTAT` 2.3.0 only had these core functions in scope.[35] The most core functions include:

- `getFAO`[36] - The heart of the package, pulls a custom slice of data from FAOstat. Renamed to `read_fao`
- `FAOsearch`[37] - Allows a user to find the dataset they're looking for using the directory in FAOstat. Renamed to `search_fao`
- `translateCountryCode`[38] - Translates country codes between formats. Renamed to `translate_countrycodes`
- `download_faostat_bulk` - Downloads bulk zip files - We opted not to change the API of this function as it has been a backbone of the package since the shutdown of the FAOstat API.

The functions were all modified for `FAOSTAT` 2.3.0, but it remained to modify all of the remaining functions so they had a consistent naming

[35] A full list of functions to be refactored or discarded is in GitLab issue #20

[36] GitLab issue #16

[37] GitLab issue #22

[38] GitLab issue #28

Function regression

It was intended that the bulk zip download files remained intact, as they were the most functional part of the package. However, it depends on `FAOsearch/search_fao` which was changed as part of the 2.3.0 release. As a result, it was necessary to refactor some of the bulk download code to restore functionality.

Caching

One of the drawbacks of moving from internal data to pulling from the API is the need to read in data every time a lookup table or other piece of reference data is required. This slows

down scripts as it is forced to wait for a reply from the FAOstat server. To mitigate that, caching was implemented. [39] Now instead of pulling every time, the server is polled only once and the response is kept and referred to for subsequent queries.

[39] GitLab issue #27

Metadata functions

When using the package, it was found that certain functionalities were immediately necessary. It was nigh-impossible to practically make a request for data without knowing what datasets were available and what their column names were. As a last minute addition, functions to perform these tasks were slipped into the 2.3.0 release.[40]

[40] Commit 06548dd0

Testing

When making changes to a codebase it is important to make tests to assure that the function works as intended but also to report when the function breaks as a result of changes to it or its dependencies. This package uses `testthat` for its tests (Wickham 2011).

Making tests is repetitive work however and many tests are structured the same way. This sort of repetitive intellectual work is perfect for an AI. The author used ChatGPT to help generate tests.[41] It was supplied with the function code and responded with valid tests from which relevant ones were selected.

[41] ChatGPTv3 conversation dated 2023-03-28

Documentation

The package is documented using `roxygen2` (Wickham et al. 2022) which allows documentation and code to be kept in the same file. Documentation has been particularly important for this project, specifically examples. As a lot of information has to be supplied to a function to get data in terms of codes, it is important that users have a clear idea of what is required as incorrect codes can result in cryptic empty responses from the server.

Examples of use

Future work

Using Webconnector

API wrappers like the `FAOSTAT` package require a substantial amount of work and are very specific to a particular service. Furthermore, there's no standard to how they should be created, meaning that even two wrappers to the same API have completely different approaches. Another approach is to have a standardised "wrapper" for the user and develop multiple config files to each allow connecting to a different API (Wu et al. 2023).

Publication

The `FAOSTAT` package has never been officially published and it is important to widely publicise it to encourage use. The following avenues are available:

- Publishing a paper in JOSS - JOSS is an online journal with peer review entirely managed by github issues
- Publish news in rweekly - Rweekly is a weekly newsletter that publishes news about R in the last week
- Publish FAO news alert - FAO has internal news services and a FAO product should be publicised through it
- Move package to FAO repository - The package is currently hosted on GitLab and should ideally be moved to a repository with FAO branding such as the the GitHub `FAOSTAT` organisation
    - FAO also has internal hosting of git repositories, so another avenue would be to make it publicly available from there

Funding declaration

This project has been funded by the Food and Agriculture Organization of the United Nations and the author is grateful for their help in reviving it.

Bibliography

Arel-Bundock, Vincent, Nils Enevoldsen, and C J Yetman. 2018. "countrycode: An R package to convert country names and country codes." *Journal of Open Source Software* 3 (28): 848. https://doi.org/10.21105/joss.00848.

Eloe, Nathan W. 2021. "Teach like a git: streamlining assignment administration and enforcing good habits with professional tools and software development practices." *Journal of Computing Sciences in Colleges* 36 (6): 27–36.

Food and Agriculture Organization of the United Nations. 2019. "Data collection." https://www.fao.org/statistics/data-collection/en/#jfmulticontent_c728270-2.

Halili, Festim, and Erenis Ramadani. 2018. "Web services: a comparison of soap and rest services." *Modern Applied Science* 12 (3): 175.

Kao, Michael C J, Filippo Gheri, and Markus Gesmann. 2022. "FAOSTAT: Download Data from the FAOSTAT Database." https://gitlab.com/paulrougieux/faostatpackage https://cran.r-project.org/package=FAOSTAT.

Li, Peter. 2023. *packageRank: Computation and Visualization of Package Download Counts and Percentiles*. https://cran.r-project.org/package=packageRank.

Morandat, Floréal, Brandon Hill, Leo Osvald, and Jan Vitek. 2012. "Evaluating the Design of the R Language." In *ECOOP 2012 – Object-Oriented Programming*, edited by James Noble, 104–31. Berlin, Heidelberg: Springer Berlin Heidelberg.

Strobel, Volker. 2018. "Pold87/academic-keyword-occurrence: First release," April. https://doi.org/10.5281/zenodo.1218409.

United Nations. 2018. "Eswatini." United Nations. https://www.un.org/en/about-us/member-states/eswatini.

United Nations Statistical Division. 2015. “Central Product Classification (CPC): Version 2.1.” United Nations Publications. https://unstats.un.org/unsd/classifications/Econ/Download/InText/CPCv2.1_complete(PDF)_English.pdf.

Wickham, Hadley. 2011. “testthat: Get Started with Testing.” *The R Journal* 3: 5–10. https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf.

Wickham, Hadley, Peter Danenberg, Gábor Csárdi, and Manuel Eugster. 2022. *roxygen2: In-Line Documentation for R*. https://cran.r-project.org/package=roxygen2.

Wu, Weiyuan, Pei Wang, Yi Xie, Yejia Liu, George Chow, and Jiannan Wang. 2023. “Web Connector: A Unified API Wrapper to Simplify Web Data Collection.” *Proc. VLDB Endow.* 16 (12): 4042–45. https://doi.org/10.14778/3611540.3611616.