Tooling for internationalisation of R help pages

Elio Campitelli and Renata Hirota

2023-09-28

Signatories

Project team

- Elio Campitelli: Ph.D. student at the University of Buenos Aires in atmospheric sciences and an R package developer. They maintain several open-source R packages (e.g., ggnewscale; metR) and contribute to other packages, such as data.table and ggplot2. They contributed to the translation of the book R for Data Science to Spanish, and to the translation of rOpenSci materials, including rOpenSci Packages: Development, Maintenance, and Peer Review.
- Renata Hirota

Contributors

- Maëlle Salmon: research software engineer at rOpenSci, previously in charge of four ISC funded projects (Catalyzing R-hub adoption through R package developer advocacy, HTTP testing in R, R-Ladies organizational guidance, Tooling and Guidance for Translations of Markdown-Based R Content (Quarto, R Markdown)) funded by the ISC, R blogger, volunteer editor for rOpenSci's system of package peer-review.
- Yanina Bellini Saibene: Community Manager at rOpenSci, R-Ladies Project Lead, Member of The Carpentries Executive Council and RForwards Core Team. Co-founder of MetaDocencia, LatinR, and R-Ladies Santa Rosa. She lead the collaborative translation of Teaching Teach Together to Spanish and MetaDocencia educational materials to Portuguese. She was involved in the translation to Spanish of R for Data Science, R-Ladies's Rules and Guidelines, some lessons by The Carpentries and severals RStudio Cheat Sheets. She is leading the Multilingual Publishing project at rOpenSci

Consulted

- Yanina Bellini Saibene

The Problem

English is the de-facto international language and this is reflected in R by the use of English for function and argument names (e.g. `mean()` instead of `promedio()` or `Mittelwert()`) and documentation language. And while contributed packages can be documented in other languages, the vast majority are documented in English.

There is a small number of packages documented in other languages, seemingly tailored to their target audience. For example, the labstatR package serves as a companion to the Italian book "Laboratorio Di Statistica Con R" and is documented in Italian. Similarly, the chilemapas package provides simplified maps for Chile, with documentation and function names in Spanish.

Although these packages are more accessible for their intended users, they are much less accessible to the wider community. Users who do not speak the language may find it difficult to discover and use the valuable functions and algorithms that these packages provide. Package authors face the dilemma of either making their package inaccessible to their target demographic or isolating it from the wider R ecosystem.

The developer of the utilsIPEA package publicly expressed the need for bilingual documentation, recognising that his package would be used by people in Brazil, who might prefer documentation in Portuguese, and the broader international community.

Interestingly, a few packages have tried to solve this dilemma by documenting their package in English and publishing a second version of the package documented in another language. At least two packages on CRAN use this strategy: ExpDes and ExpDes.pt, as well as orloca and orloca.es.

This approach is very hard to maintain, doesn't scale well to multiple languages, and requires users to load a different package to access the documentation in their language. A more effective solution would be for R support for multilingual documentation as a standard feature.

The proposal

Overview

We propose to extend the R help system to enable multiple help pages for the same function in different languages by installing translation modules. This was discussed at the R Project Sprint 2023 with R Core members Martin Mächler, Deepayan Sarcar and Michael Lawrence. By default, `help()` would display the documentation in the user's preferred language, if available, and fall-back to the canonical documentation otherwise (most likely, in English). It would also include a link to the canonical documentation and warnings if translations are not up to date.

Detail

The system would require three parts

Translation of help pages

The translation of help pages would be performed using .pot files. This would require a parser that takes .Rd files and creates .pot files ready for translation. These files could be translated using existing infrastructure such as weblate.

Installation of translation modules

The translated .po files would be hosted in translation modules. These modules would piggyback on the package infrastructure but declare `PackageType: translation` and the name and version of the package they are translating in `Depends`.

Translation modules wouldn't necessarily maintained by the original package authors. In principle, multiple translation modules for the same package and the same language could exist and the user can chose which version they install. They could be hosted on CRAN or a CRAN-like repository. Optionally, CRAN could develop quality checks on translation modules to ensure that they are up-to-date, such as a maximum percentage of missing strings or maximum percentage of fuzzy matches.

The user would install them like regular packages (e.g. `install.pacakges("mein.paket.en")`. At installation the .Rd files of the original package would be parsed and translated using `gettext()` and the .po files in the translation module, and serialised into binary help pages (like regular packages have).

Accessing documentation

The `help()` function would search for the loaded topics and if any translation is available, then it would use a new `language` argument to disambiguate. This argument would default to `Sys.getenv("LANGUAGE")`.

Help pages would include a link to the original (canonical) documentation and add warnings if strings are untranslated/outdated.

Minimum Viable Product

???

Project plan

Start-up phase

Before starting the project proper we will consult on the R help system internals, .Rd parsing, and .pot creation and translation. Depending on scheduling with the people involved, this could take around 2 weeks.

Technical delivery

Target date is realtive to after the start-up phase.

Delivery	Hours	Target date
.Rd parsing.	25	2 weeks
Creation of .pot files	10	3 weeks
Translation of .Rd files using .po files	10	4 weeks
Installation of translation modules	30	6 weeks
Rendering of help pages in selected language	30	8 weeks

Other aspects

We will promote the work via blogposts in the rOpenSci blog during development and announce the package when it's done. 16 hours in total.

We will also write an article for JOSS. 20 hours.

Requirements

People

Elio Campitelli and Renara Hirota would be responsible for implementation. Several other people would give their know as consultants: Michael Lawrence would serve as R Core contact, Gergely Daroczi would add his expertise with .po files and the `gettext()` infrastructure, and Maëlle Salmon would help with their knowledge of translation and internationalisation workflows.

Processes

This project will be covered by a Code of Conduct adapted from rOpenSci's Code of Conduct.

Tools & Tech

We would use a GitHub repository for collaboration and code hosting.

Funding

Summary

Success

Definition of done

Success of the project will be to have one or more R packages that implement the following functionality:

1. Parsing .Rd files into .pot files.
2. Translation of .Rd files using .po files.
3. Installation of translation modules.
4. Access of translated documentation using a `help()`-like function.
5. Modification of rendered help pages to include links to the canonical documentation.

Additionally, toy packages to showcase and test the functionality will be created.

Measuring success

Use of this functionality in a real world package, such as agromet.

Future work

Test in real world packages and workflows and gather feedback to improve the system. Make changes in the help rendered by `help.start()` to support multiple languages. Once the project is mature, incorporate the functionality into R itself.

Key risks

.Rd files can be very complex and even dynamically created. Parsing them could be to hard or brittle to allow for robust creation of .pot files. Alternatively, the `gettext()` system is designed for short pieces of text and might not prove ergonomical. In those cases, an alternative system relying on whole .Rd files hosted in translation modules could be explored.