Tooling for internationalisation of R help pages

Elio Campitelli and Renata Hirota

2023-09-29

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.

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

- Gergely Daróczi
- Hadley Wickham
- Michael Lawrence

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.

At least two packages have tried to solve this dilemma by documenting their package in English and publishing a second version documented in another language: 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 a system in which either package maintainers or community members could create translation modules of specific packages. Users would then be able to install those translation modules and browse their documentation. By default, `help()` would display the documentation in the user's preferred language, if available, and fall-back to the canonical documentation otherwise. It would also include a link to the canonical documentation and warnings if translations are not up to date.

This system would require tooling or changes in three aspects.

1. **Create and maintain.** There are established tools for localisation, but these don't use the .Rd format used by R. We would provide a tool to go from .Rd files to other formats, such as .pot files and back.
2. **Distribute and install.** Translation modules would be hosted as normal packages with special metadata and installed with the usual tools. Depending on the implementation, these tools might need to be tweaked to recognise translation modules.
3. **Access.** The `help()` function would need to recognise topics with multiple help pages and show the user the appropriate help page. These pages would also need to be changed to include links to the original documentation and warnings for out-of-date translations.

Adding support for multilingual documentation in R itself would be a big task, so this proposal aims to implement parts of this system in package space. This would enable quick prototyping and iterating with the long-term goal of merging the implementation into R once it matures. This was discussed at the R Project Sprint 2023 with R Core members Martin Mächler, Deepayan Sarkar and Michael Lawrence.

Detail

In this project we will create a minimum viable product and test its functionality by translating the documentation of the agromet package into multiple languages, as well as a few important base R functions, such as `mean()` or `lm()`. We will implement the installation of the created translation modules and a `help()`-like function to access and render them.

We will also set up an R Consortium Working Group to ensure long-term support for the overall goal of maturing this system and integrating it into R itself.

Code will be hosted in a GitHub repository governed by a Code of Conduct adapted from rOpenSci's Code of Conduct and published under an open licence.

We will promote the work via blogposts in the rOpenSCi blog and/or R-Hub blog during development. We anticipate about 3 posts –roughly touching on the three aspects of the project– and one announcement post when the package is available.

We will also write an article for JOSS or The R Journal.

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 relative to after the start-up phase.

Delivery	Hours	Target date
Creation of Working Group	8	2 weeks
Writing blog posts	16	Throughout the project
Creation of sample translation modules	56	4 weeks
Installation of translation modules	56	8 weeks
Rendering help pages in selected language	56	16 weeks
Writing of article for JOSS/R Journal	16	20 weeks
Total	208	20 weeks

At $100 per hour used in previous R Consortium projects, this would cost $20,800 in total.

People

Elio Campitelli 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.

Success

Success of the project will be to produce at one or more package that provide the full cycle of translating, installing and accessing translated documentation.

Measuring success

Use of this functionality in more real world packages.

Future work

- Test in real world packages and workflows and gather feedback to improve the system.
- Test support of multiple languages, including CJK and right-to-left languages.
- Exploring scalability to multiple packages and multiple languages (e.g. can a single translation module be used for multiple packages?).
- Explore other methods, like dynamic delivery of translations from remote source, machine translation.
- Make changes in the help rendered by `help.start()` to support multiple languages.
- Extend the system to support vignettes.

- Explore methods of automatically searching, installing and updating translation modules for installed packages.
- 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.