rOpenSci Localization and Translation Guidelines

Yanina Bellini Saibene, Paola Corrales, Elio Campiteli, Maëlle Salmon

Table of contents

Preface

Welcome!

This book explains all the steps and tools involved in the localization of our materials, including translations. We designed, built, and tested a process for maintaining different language versions of our materials which borrows ideas from our software peer review system and the maintenance of open-source projects.

We know that code peer review ensures the quality of our software; thus with the same goal of ensuring the quality of our content in different languages, we implemented a system to review and maintain the localization of our guides, books, blog post, and other content.

We also know that by using roles (such as reviewers and maintainers) and tools (such as GitHub, including Pull Requests) known to our community, we make this process easier to understand, apply and, therefore, easier to contribute to.

Intended Audience

This book is a guide for people who would like to *localize rOpenSci materials* or *contribute to maintaining localized material.* It is also intended for people that have to localize material for another community or group. You don't need to be a professional translator or developer to contribute to this effort. The following personas are examples of the types of people that are our target audience.

Ana is a Colombian student doing her Ph.D. at an university in Argentina. She works with geospatial data. She is not an expert in programming, but she took one of The Carpentries workshops for scientists and used some of the rOpenSci suite packages for handling and accessing geospatial data for her thesis. Language was one of the barriers she faced when learning to code. Ana wants to contribute by translating rOpenSci material into her native language so others can access this knowledge with less effort. This book will show her how to do a new translation and how to review an existing one.

Francesco will send their packages to the rOpenSci review process in Spanish. They know that this will ensure the quality of their software and will also facilitate its publication in JOSS, which will give them academic credit for their job at their university. They read the Spanish version of the book "rOpenSci Packages: Development, Maintenance, and Peer Review" to prepare their submission. If their review experience is good, they will volunteer to be a reviewer in their native language and contribute to the book. This guide will explain how to become a contributor and maintainer of translated material and the infrastructure that allows translations.

Bauti is the co-founder of a user group in his city in Brazil. With a couple of university classmates, he teaches and learns together how to code and do reproducible analysis using different languages. He uses rOpenSci Portugese material during his workshops and would like to have the material from other programming languages in Portuguese. He wonders how rOpenSci manages translation. This book will show him how we organize translation and what tools we use so that he can apply in his community.

What You Will Learn

The guide contains:

- Why we localize, translate and use multilingual publishing.
- General guide with the infrastructure and roles, step-by-step technical processes, and localization considerations, including the technical terms to translate and the technical terms we will not translate.
- Language-specific guides with localization considerations.
- Language-specific glossaries.

At rOpenSci, we use R as our main programming language and GitHub to host our code, including our packages, books, webpage and review process. The step-by-step technical process refers to working with GitHub, like making Pull Requests, reviewing and commenting on Pull Requests, and creating Issues. It also explains how to use the {babeldown} R package which facilitates working with translations.

Localization considerations and glossaries are language-specific and detail the agreements the community has reached about how to localize and translate the material. This chapter is different for each language present in this guide.

We hope that you'll find the guide useful and clear, and welcome your suggestions in the issue tracker of this book. Please see the Contributing and Re-Use section to learn more about how to contribute.

Happy localization!

Contributing and Re-Use

This book is a living document. If you want to contribute to this book, corrections, additions, and suggestions are very welcome. Everyone whose work is included will be credited in the acknowledgements. Please refer to the GitHub repository, particularly the contributing guidelines and our Code of Conduct, for more information on how to contribute.

You can cite this book:

rOpenSci Localization and Translation Guidelines v1.0. Yanina Bellini Saibene, Paola Corrales, Elio Campitelli, Maëlle Salmon (November, 2023)

You can re use this this work under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 License. If you do, we would love to hear about it! Please let us know by adding a use case in our public forum.

Thanks!

You can also read the PDF version of this book.

Attributions

These translation guidelines are based on the experience and documents generated during the translation of Teaching Tech Together.

1 Motivation

At rOpenSci we work to transform science through open data, software and reproducibility. We provide community support, standards, and infrastructure for scientists and research software engineers working in R to develop, maintain, and publish high-quality open-source scientific software. In addition, we develop and maintain high-quality documentation and resources to support these activities.

We also believe that science and the research software we create to support it, should serve everyone in our communities, which means it needs to be sustainable, open, and built by and for all groups. Currently, however, there is a dismaying lack of diversity in scientific and open source communities in general. This lack of diversity is potentially detrimental to the sustainability, utility and productivity of projects.

rOpenSci is carrying out a series of activities and projects to transform science to ensure that it serves everyone in our communities. One of these projects is our multilingual publishing project.

1.1 Language barrier

English is the *lingua franca* for science creating a significant barrier for non-English speakers wanting to join the field. It is also the predominant language of open source in code, content, and community interactions making *language access* one of the *environmental barriers to equity in open source.*[1]

The UNESCO's recommendation on open science highlights the need to overcome language barriers in order to achieve the open science core values and guiding principles of Equity and fairness, Diversity and inclusiveness, Equality of opportunities, and Collaboration, participation and inclusion[2].

[1]Hilary Carter and Jessica Groopman, "Diversity, Equity, and Inclusion in Open Source: Exploring the Challenges and Opportunities to Create Equity and Agency Across Open Source Ecosystems", foreword by Jim Zemlin, The Linux Foundation, December, 2021, https://www.linuxfoundation.org/research/the-2021-linux-foundation-report-on-diversity-equity-and-inclusion-in-open-source

[2]UNESCO. UNESCO recommendation on open science. Paris, France; 2021.

A recent study[3] quantified the consequences of language barriers on the career development of researchers. The authors found that non-native English researchers need:

- Up to **91% more time to read** English papers.
- Up to **51% more time to write** papers and their papers will be **rejected 2.6 times more often**. If accepted, they'll have to **revise it 12.5 more times**.
- When they present in English, need up to **94% more time** to prepare their presentations. They will **often avoid** English speaking conferences and oral presentations.

Furthermore, the Linux Foundation report states, "English proficiency is a metric by which performance and personality can be judged"[4].

These studies also highlight that "the magnitude of the disadvantage seems far beyond the level that can be overcome with individuals' efforts."[5]

By developing the technical and social infrastructure to publishing multilingual resources and publish our own resources in several languages, we can lower these barriers by *increasing access to knowledge* and *democratizing access to quality resources*, thereby *increasing the potential for individuals to contribute* to software and open science projects.

1.2 Community driven localizations and translations

Community-driven translations and localizations are efforts by community members to create resources in a language of their choice. The community organizes and agrees on different aspects of localization projects[6].

The Spanish-speaking R community has been very active and growing in recent years and has undertaken various translation activities for technical materials such as books, cheat sheets, guides, and datasets.

[3] Amano T, Ramírez-Castañeda V, Berdejo-Espinola V, Borokini I, Chowdhury S, Golivets M, et al. (2023) The manifold costs of being a non-native English speaker in science. PLoS Biol 21(7): e3002184. https://doi.org/10.1371/journal.pbio.3002184

[4] Hilary Carter and Jessica Groopman, "Diversity, Equity, and Inclusion in Open Source: Exploring the Challenges and Opportunities to Create Equity and Agency Across Open Source Ecosystems", foreword by Jim Zemlin, The Linux Foundation, December, 2021, https://www.linuxfoundation.org/research/the-2021-linux-foundation-report-on-diversity-equity-and-inclusion-in-open-source

[5] Amano T, Ramírez-Castañeda V, Berdejo-Espinola V, Borokini I, Chowdhury S, Golivets M, et al. (2023) The manifold costs of being a non-native English speaker in science. PLoS Biol 21(7): e3002184. https://doi.org/10.1371/journal.pbio.3002184

[6] Yanina Bellini Saibene and Natalia Soledad Morandeira. Multilingual Data Science: Ten Tips to Translate Science and Tech Content. Chapter at Our Environment. A collection of work by data designers, artists, and scientists. ISBN:979-8-218-20191-3. http://datasciencebydesign.org/blog/multilingual-data-science

In 2017, several Latin American R-Ladies began translating the R-Ladies' Code of Conduct and their Rules and Guidelines into Spanish.

In 2018, the R community in Latin America collectively translated the R for Data Science book into Spanish. This included the translation of all the data sets used in the book, which were compiled in the datos package, making it an excellent tool for teaching.

The community continued with the translation of Teaching Tech Together and contributed to the translations into Spanish of the Posit Cheatsheets, The Carpentries' and The Programming Historian lessons.

Driven by this active and growing Spanish-speaking community, rOpenSci successfully piloted our first Spanish-language peer review, where the submission, reviews, and editorial responses were in Spanish.

These works and previous community experiences created the conditions for us to start our multilingual publishing in Spanish. Spanish is the second most-spoken native language in the world and is one of the most geographically widespread languages, being an official language in many countries[7].

1.3 Building community

At rOpenSci, we understand review as a way of building community and hope our review process in Spanish and Portuguese, will allow us to continue building the community in regions that speak these languages, to increase the number of contributors, and to get feedback on how our tools and processes can be improved to better serve these community members.

In addition to using our material to learn how to contribute to open source as a developer, maintainer, reviewer, or editor, people can also contribute through translation. This type of contribution is a good way to engage in open source and is recognized as valuable by the community.

We also expect that the multilingual project's documentation and tooling will be useful for extending this effort to other languages and for other communities and projects undertaking translation efforts.

1.4 References

[7] List of languages by number of native speakers. Accessed on December 1, 2022. https://en.wikipedia.org/wiki/List_of_languages_by_number_of_native_speakers

2 Overview

“Meet your community members where they are.”

2.1 Introduction

rOpenSci provides community support, standards, and infrastructure for scientists and research software engineers working in R to develop, maintain, and publish high-quality open-source scientific software. In addition, we develop and maintain high-quality documentation and resources to support these activities. Many of our materials are defined as *living documents*, meaning that they are constantly being improved and updated.

Multilingual publishing of documentation and resources involves two aspects, Internationalization and Localization[1]:

- *Internationalization* refers to technological solutions which allow software to adapt to different regions without requiring changes to the source code. This infrastructure is what *enables* us to localize our content.
- *Localization* is the process of taking a piece of content in its original form and converting it into something accessible and suitable for another region, country, or audience. This includes language, date formats, currency, measurement units, and support for different character sets.

In addition, localization of living documents has two well-defined stages involving different resources:

Stage 1. Translating the material in the first place.

Stage 2. Keeping the material updated and synchronized among the supported languages[2].

Translation is typically the most time-consuming component of these efforts[3].

[1] Internationalization and localization. Accessed November 1, 2022. https://en.wikipedia.org/wiki/Internationalization_and_localization

[2] Yanina Bellini Saibene and Natalia Soledad Morandeira. Multilingual Data Science: Ten Tips to Translate Science and Tech Content. Chapter at Our Environment. A collection of work by data designers, artists, and scientists. ISBN:979-8-218-20191-3.

[3] Internationalization and localization. Accessed November 1, 2022. https://en.wikipedia.org/wiki/Internationalization_and_localization

2.2 Technical infrastructure and workflows

There are many solutions and tools for internationalizing and localizing content and software. For example, translation management systems (Crowdin, Transifex, Weblate), automatic translators (Google Translate, DeepL), version control systems (GitHub, GitLab), markup languages (LaTeX, Markdown), and tools for writing these languages (Overleaf, Quarto).

These technological solutions are continuously evolving. The choice of technology impacts the paths we create for contributions, so care should be taken to pick technology that reduces barriers to participation as much as possible[4].

We developed our infrastructure for our localization effort using the tools that best suit our team, contributors and materials, and thus our community.

- We created the babeldown package to create automatic translations of documents, including a workflow for updating existing translations.
- We created the babelquarto package to configure and generate multilingual Quarto books or websites.
- We use GitHub projects to track each localization project's progress and the people who have different contribution roles, like *reviewers*, *editors,* and *maintainers.*
- We also created (and documented in these guidelines) a workflow that follows the same idea of using the tools our community already knows and uses in their activities.

2.3 General aspects of the stage 1 of the translation process

The translation process starts with an initial machine translation using DeepL via the babeldown package. This provides a first draft that is then reviewed by a human, who correct errors and incorporates the localization and language-specific guidelines.

To minimize errors and promote a broader perspective of the translation, at rOpenSci we ask that each chapter or section goes through at least two reviews in sequence (first a review of the automatic translation and second a review of the first review) followed by an overall review of the book or document as a whole. If possible, reviewers should come from different countries, in order to consider the different ways in which the language is spoken around the world.

The translation and review process is done on GitHub using *pull requests* (see the Pull Request section for details). We chose this workflow as this is the infrastructure we use in our community. All our packages and books are hosted on GitHub and use *issues* and *pull*

[4] Yanina Bellini Saibene and Natalia Soledad Morandeira. Multilingual Data Science: Ten Tips to Translate Science and Tech Content. Chapter at Our Environment. A collection of work by data designers, artists, and scientists. ISBN:979-8-218-20191-3.

requests in their development. In addition, this allows the process to be open so that others can contribute and provide feedback.

Of course, as in all areas of rOpenSci, this is process is subject to our code of conduct to create a friendly and safe environment.

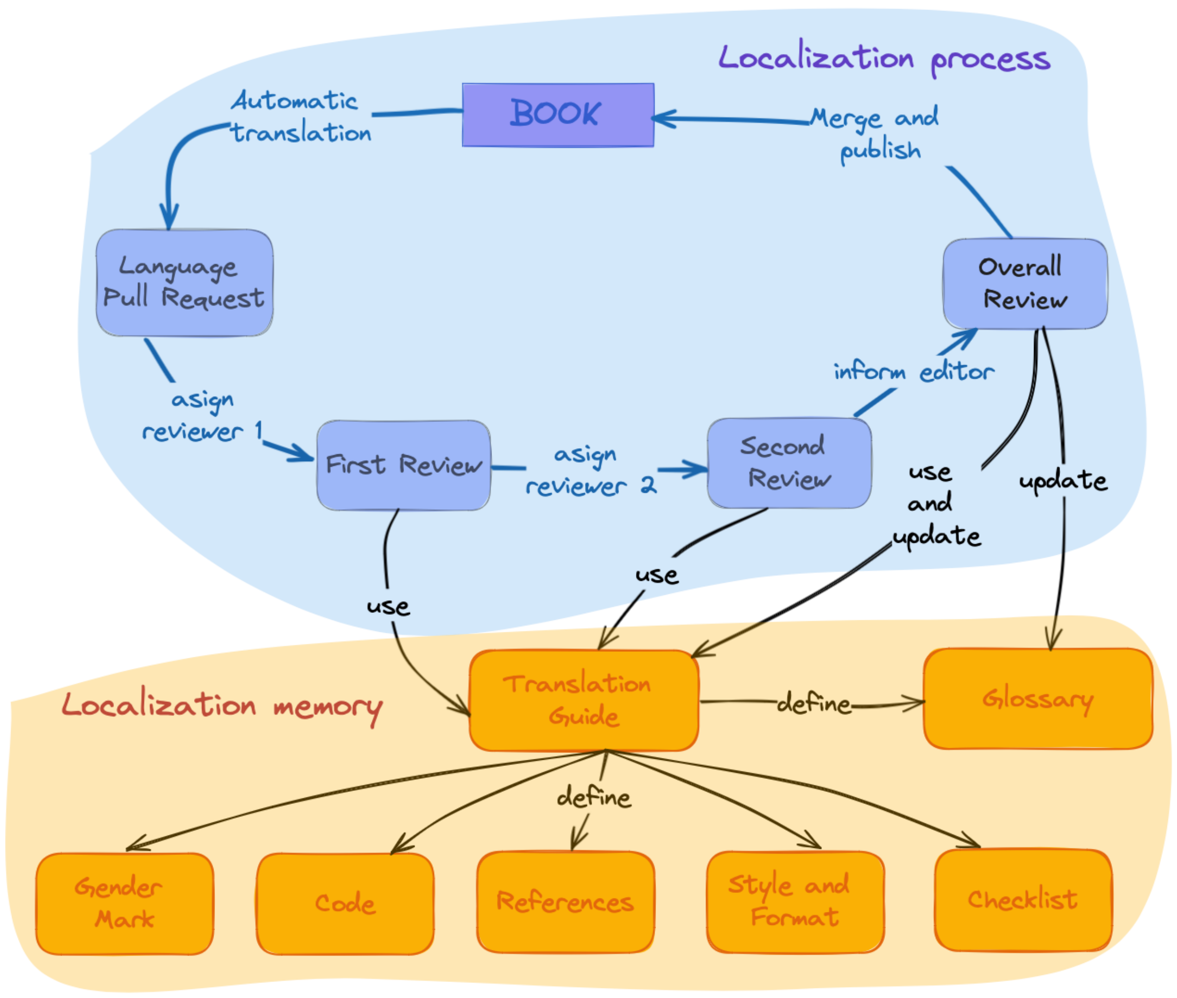


Figure 2.1: rOpenSci Localization Process

2.4 General aspects of the stage 2 of the translation process

In this case we are updating changes in the original language of a content that has already been translated and we have to reflect those changes in the translated languages.

The process is similar to the stage 1 translation process, but with some differences:

1. After the content change in the original language, the maintainer of the translation project creates a new automatic translation using the babeldown package.
2. The language maintainer reviews the pull request with the automatic translation and apply the necessary changes.
3. The language maintainer merges the pull request with the review translation.
4. The maintainer updates the language specific translation guidelines and the glossary if necessary.

Maëlle Salmon's blog post How to Update a Translation with Babeldown explain this process in more detail.

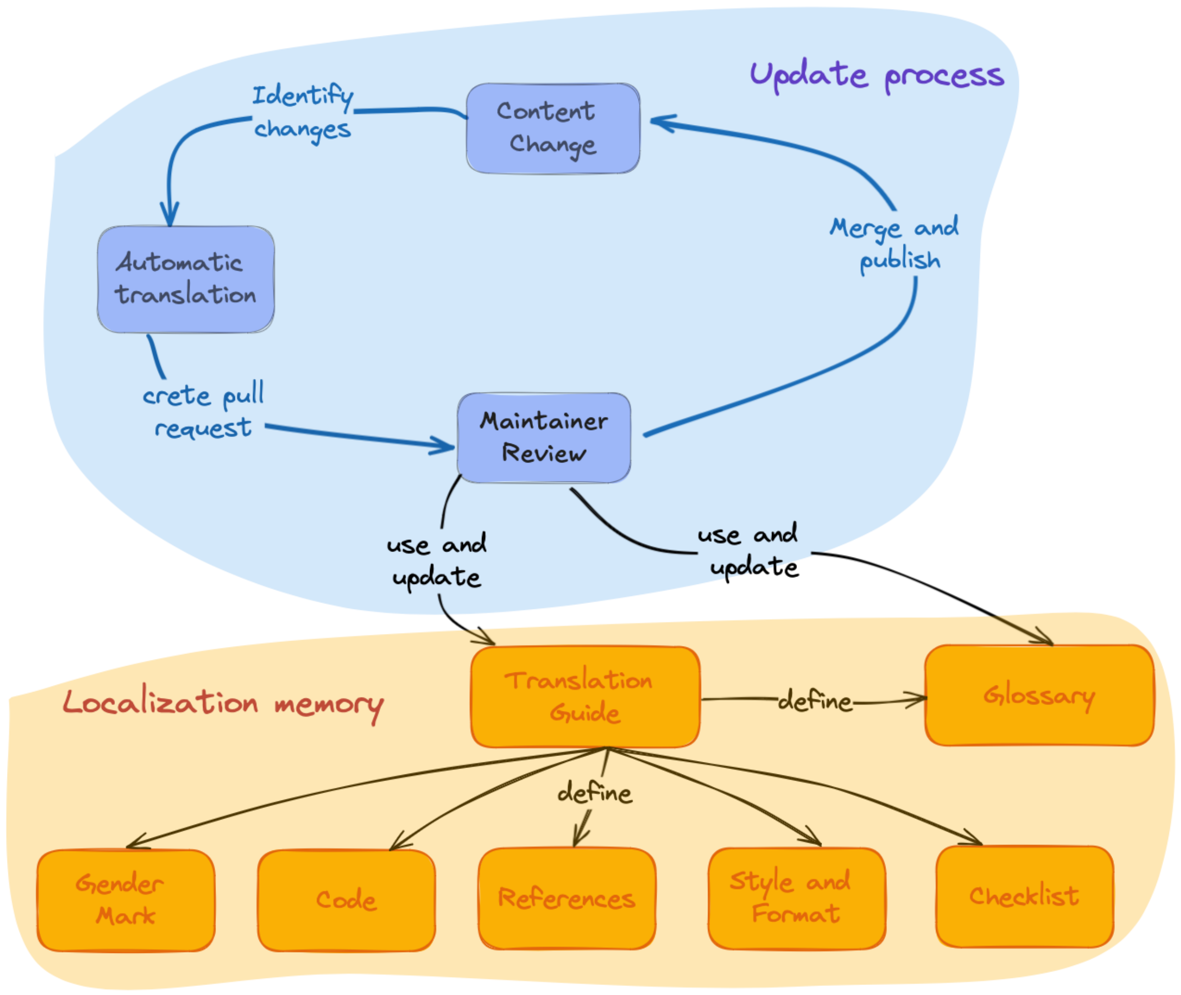


Figure 2.2: rOpenSci Update Localization Process

2.5 Referencing localizations materials

Here we give suggestion for citing a translation or referencing your translation work in a CV. These examples specifically refer to the translation of "rOpenSci Packages: Development, Maintenance, and Peer Review", but you can apply these recommendations to any translation.

2.5.1 Citing a translation

The general format is[5]:

> {Authors of original} ({Year of translation publication}). {Translated title} [{Original title}] (Translation to {Language}: {Authors of translation}). {DOI} (Original work published {Year of publication})

The text *Translation to* and *Original work published* should be written in the language of the translations.

Example of a citation using the Spanish translation of the rOpenSci Packages: Development, Maintenance, and Peer Review book:

> rOpenSci, Anderson, B., Chamberlain, S., DeCicco, L., Gustavsen, J., Krystalli, A., Lepore, M., Mullen, L., Ram, K., Ross, N., Salmon, M., Vidoni, M., Riederer, E., Sparks, A., & Hollister, J. (2021). Paquetes rOpenSci: Desarrollo, mantenimiento y revisión por pares [rOpenSci Packages: Development, Maintenance, and Peer Review] (Traducción al español: {Authors of translation}). https://doi.org/10.5281/zenodo.6619350 (Trabajo original publicado en 2021)

2.5.2 Citing your contribution in your CV

In general, you can use something along the lines of this format:

> {Start date} – {End date}. Collaborative localization to {Language} of "{Book Name}" (Lead editor {Name of lead editor}). Contribution as {Role}. Project details: {Link to the project}.

Here is a specific example for a CV in English referencing work on a Spanish translation project:

[5] Based on the APA article "How to cite translated works". We also suggest including the original title in English, following this example for Piaget (1950).

May, 2022 – July, 2020. Collaborative localization to Spanish of "rOpenSci Packages: Development, Maintenance, and Peer Review" (Lead editor Yanina Bellini Saibene). Contribution as reviewer. Project details: https://github.com/ropensci/dev_guide.

2.6 References

3 Translation and Review Guidelines

3.1 Contributing

3.1.1 Contributing to an existing project

If you're interested in contributing to an existing translation project, you'll need to

1) get in touch with the people maintaining the project,
2) familiarize yourself with the source material to be translated, and
3) read this guide, paying close attention to the language-specific guidelines

You can contact a project maintainer to express your interest through email, Slack, or GitHub. On the list of active projects, you will find information on each project, including the maintainer. Each translation project also has a channel on the rOpenSci Slack where translation decisions are discussed and made, which are reflected in the language-specific guidelines in this guide and a glossary. If you join a translations project, we will make sure you get an invitation to the Slack channel.

It's a good idea to ask how review tasks are being divided so that everyone knows what they are responsible for.

Do not hesitate to ask any questions you may have, such as general questions like "What is the expected time commitment?", or technical questions like "What do I do if I run into problems on GitHub?"

3.1.2 Proposing a new translation project

Starting a new translation project requires work and a commitment to the process, but can be a really rewarding experience!

Because multiple reviewers are required for all translations and there must be at least one active project maintainer, we recommend that those interested in starting a new translation project invite their local community to participate and form an initial team before starting the process.

If the language you are interested in translating does not have an active project, please contact the rOpenSci team to propose it. You can send an email to info@ropensci.org or open an *issue* in this repository.

Please note that starting a new project requires a commitment to manage the repository where all the reviews will take place. We ask that those proposing a new translation project commit to maintaining it for at least 2 years, or to finding a new maintainer if they must give it up within 2 years. This is the same commitment we ask of our package maintainers.

3.2 Technical aspects

3.2.1 General Guidelines

In each step of this process, we use the following guidelines:

1. Use a conversational voice rather than a formal or academic voice.
2. If appropriate, specify the dialect or regional language variation used. For example, the Spanish translation uses Latin American conventions.
3. Try to be gender-neutral. If the language you are working on has a strong grammatical gender, the translation adjusts wording to avoid assigning a gender. Where a gender mark cannot be avoided, follow the specific language agreement in the use of inclusive language or non-sexist language.
4. Try to be idiomatic. Don't worry about literal translations. Bring the message closer to your audience[1] by choosing the text and form in the target language that best expresses the *meaning* of the fragment from the original.
5. Take breaks when reviewing. When we work for a long time on a text, it is difficult to identify typing errors. As a suggestion, once you finish the revision of the chapter or section, let some time pass (a few hours or a day) before doing the final reading and sending it for the next review. This makes it easier for this type of detail to jump out and allows those who do the review to focus on the quality of the translation rather than on orthotypographic corrections. It is also useful to review the text in a formatted version (compiled as pdf or html), since some errors will be more visible than in the raw version. If you have the possibility, a printout on paper, both of the raw version and the compiled version, also helps to see the errors more clearly.

[1]Yanina Bellini Saibene and Natalia Soledad Morandeira. Multilingual Data Science: Ten Tips to Translate Science and Tech Content. Chapter at Our Environment. A collection of work by data designers, artists, and scientists. ISBN:979-8-218-20191-3.

3.2.2 Organization

In this section we will use the rOpenSci Packages: Development, Maintenance, and Peer Review guide as an example, but all technical aspects apply to all other rOpenSci translations.

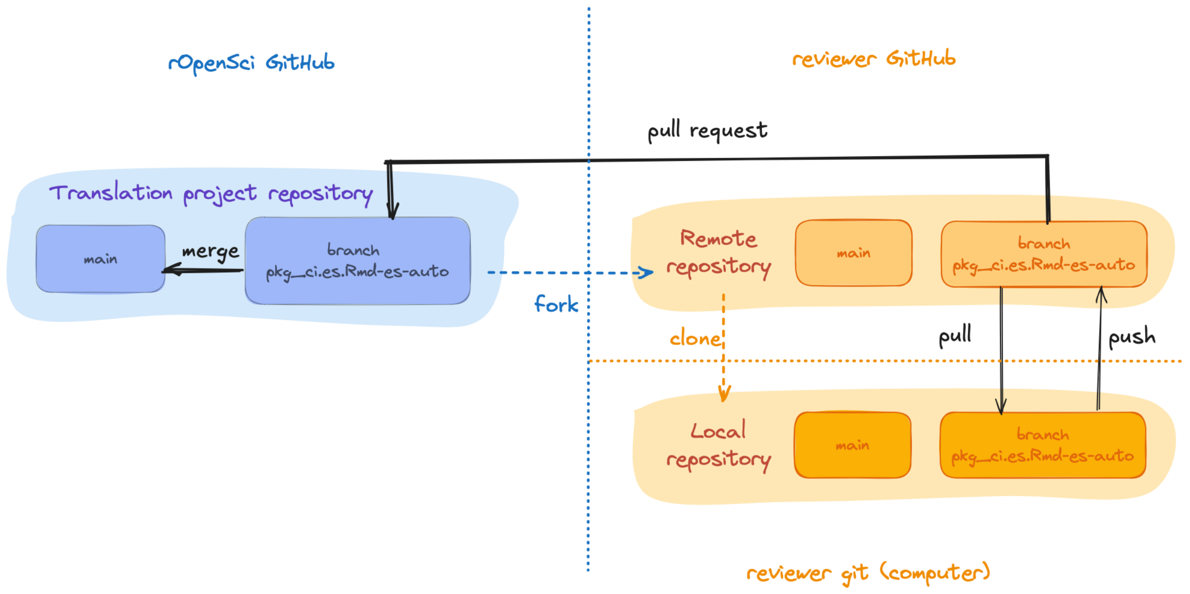
The source code that generates this material to be translated lives at github.com/ropensci/dev_guide, and is organized in .Rmd (R Markdown) files that contain text and sometimes also code. In this case there are also some configuration files that need to be translated so that the workflow for creating the guide in the new language is fully translated.

Each chapter has one .Rmd file per language, with a suffix that identifies the language using its two-letter code (according to ISO 639-1). For example, the chapter "Software Peer Review Policies" will have its original English version in `softwarereview_policies.Rmd` and its translated Spanish version in `softwarereview_policies.es.Rmd`.

If the translation project is in progress, you will find a series of *pull requests*, **one for each file to translate**, generated with the babeldown package, containing the automatically translated text. **Each of these *pull requests* has an associated *branch*.** For example, translations of the `pkg_ci.Rmd` file to Spanish would be found in the PR merging the `pkg_ci.es.Rmd-es-auto` branch, and called "Add Spanish automatic translation pkg_ci.es.Rmd".

In fact, if there are several translation projects going on at the same, you may encounter many *pull requests* for different files and different languages. We recommend that you filter the *pull requests* using the tag for your language, for example the tag for Spanish translation is "traducción ". You can also check the status of each *pull request* on the GitHub project board of each active language.

For security and organization, reviewers do not have write permission in the main repository. The project maintainer will make a *fork* (we call this the **working repository**) and will give write permission to this fork to all the project contributors. This *fork* will have the same *branches* as the main repository; this is key since the **review of each file will be done on the *branch* associated to its translation.** This means that a review of the Portuguese translation would happen on the `pkg_ci.pt.Rmd-pt-auto` branch, whereas the review of the Spanish translation would happen on the `pkg_ci.es.Rmd-es-auto` branch.



Once you've contacted the project maintainer and you're ready to start contributing, the first thing you should do is to clone the working repository. Once cloned, you can start working with the repository locally on your computer. If you haven't already had a discussion about what you will be reviewing, you can ask in the Slack channel or comment on a *pull request* to ask if you can take the review of that chapter. We strongly suggest that you ask **before** starting work on your review.

Now let's start a review!

3.2.3 Review 1

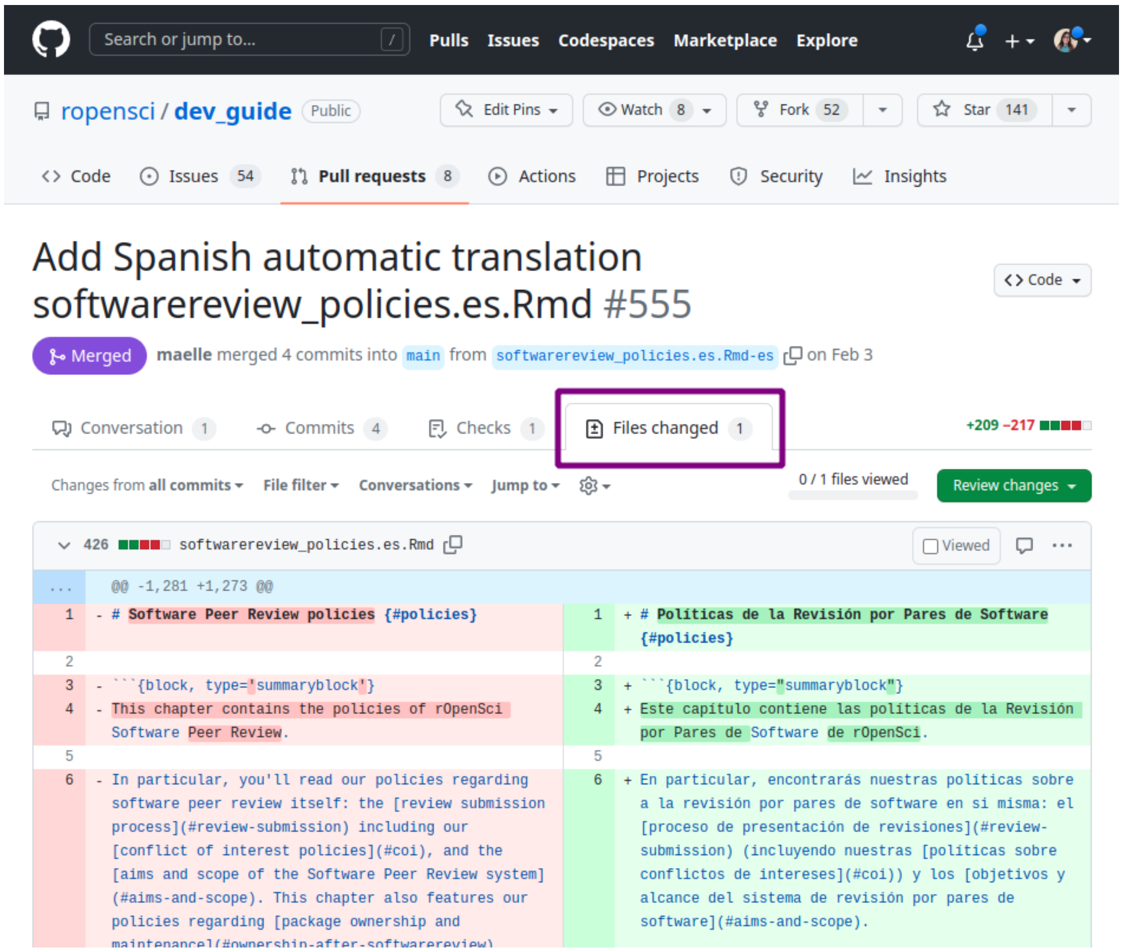
The purpose of this first review is to make a detailed reading of the machine translation in order to:

- Ensure that the meaning of the sentence or paragraph is maintained.
- Check that the glossary terms agreed upon by the community have been used according to the context.
- Apply stylistic decisions whenever possible. For example, in Spanish we decided to avoid gender marking of nouns and adjectives by paraphrasing sentences or using "las/los" if paraphrasing is not possible.
- Check that code blocks or Markdown markup have not been affected.
- Translate variable names and comments visible in the code blocks whenever appropriate.

Often there will still some questions and it is possible that some things may be overlooked. Style decisions are often the most difficult to implement and sometimes it is necessary to consult with the rest of the team or discuss it later with the second reviewer.

Working locally on your computer, here are the steps to follow when performing a first review of a file:

1. **Check out the *branch* associated with the file you are going to review.** For example, if you are reviewing `softwarereview_policies.en.Rmd`, the branch would be `softwarereview_policies.es.Rmd-es`.
2. **Open the file.** For example, `softwarereview_policies.es.Rmd`. You may see things to improve immediately, as the automatic translation does a good but imperfect job.
3. **Start reviewing the text.** For this we suggest you have the English version next to you as you may need to go back to the original version to understand the context of some phrase or word that the translator has not been able to translate correctly. You can use the web version of the material you are translating or the *"Files changed"* tab of the PR in the main repository.



Add Spanish automatic translation softwarereview_policies.es.Rmd #555

maelle merged 4 commits into `main` from `softwarereview_policies.es.Rmd-es` on Feb 3

426 softwarereview_policies.es.Rmd

```
@@ -1,281 +1,273 @@
- # Software Peer Review policies {#policies}

- ```{block, type='summaryblock'}
- This chapter contains the policies of rOpenSci
  Software Peer Review.

- In particular, you'll read our policies regarding
  software peer review itself: the [review submission
  process](#review-submission) including our
  [conflict of interest policies](#coi), and the
  [aims and scope of the Software Peer Review system]
  (#aims-and-scope). This chapter also features our
  policies regarding [package ownership and
  maintenance](#ownership-after-softwarereview)
```

```
+ # Políticas de la Revisión por Pares de Software
  {#policies}

+ ```{block, type="summaryblock"}
+ Este capítulo contiene las políticas de la Revisión
  por Pares de Software de rOpenSci.

+ En particular, encontrarás nuestras políticas sobre
  a la revisión por pares de software en si misma: el
  [proceso de presentación de revisiones](#review-
  submission) (incluyendo nuestras [politicas sobre
  conflictos de intereses](#coi)) y los [objetivos y
  alcance del sistema de revisión por pares de
  software](#aims-and-scope).
```

4. ***Commit* your changes.** Many times the content of the files will be extensive and you will need a lot of time to complete the revision. In these cases we suggest you make periodic *commits* at least after each work session indicating in the commit message which line of the file you have reached. This way, if you need to delegate the review to someone else, that person can review the file history and pick up where you left off.

5. ***Push* to the working repository.** So far the review only lives in your local repository, pushing it moves it online. It is very important that you *push* to the *branch* associated to the file you are working on, otherwise we will not be able to connect your review with the original translation.

6. **Notify the project maintainer or the second reviewer.** Once you have completed your review, notify the project coordinator or the person assigned to review 2, if there is one.

3.2.4 Review 2

You're going to do the second review, great!

Whoever did the first review, whom we will call **R1** for now. most likely generated one or more *commits* with changes over the original translation. This means that your work does not start from scratch. This second review seeks builds upon the first to generate a text that maintains the original content but also sounds natural in the new language.

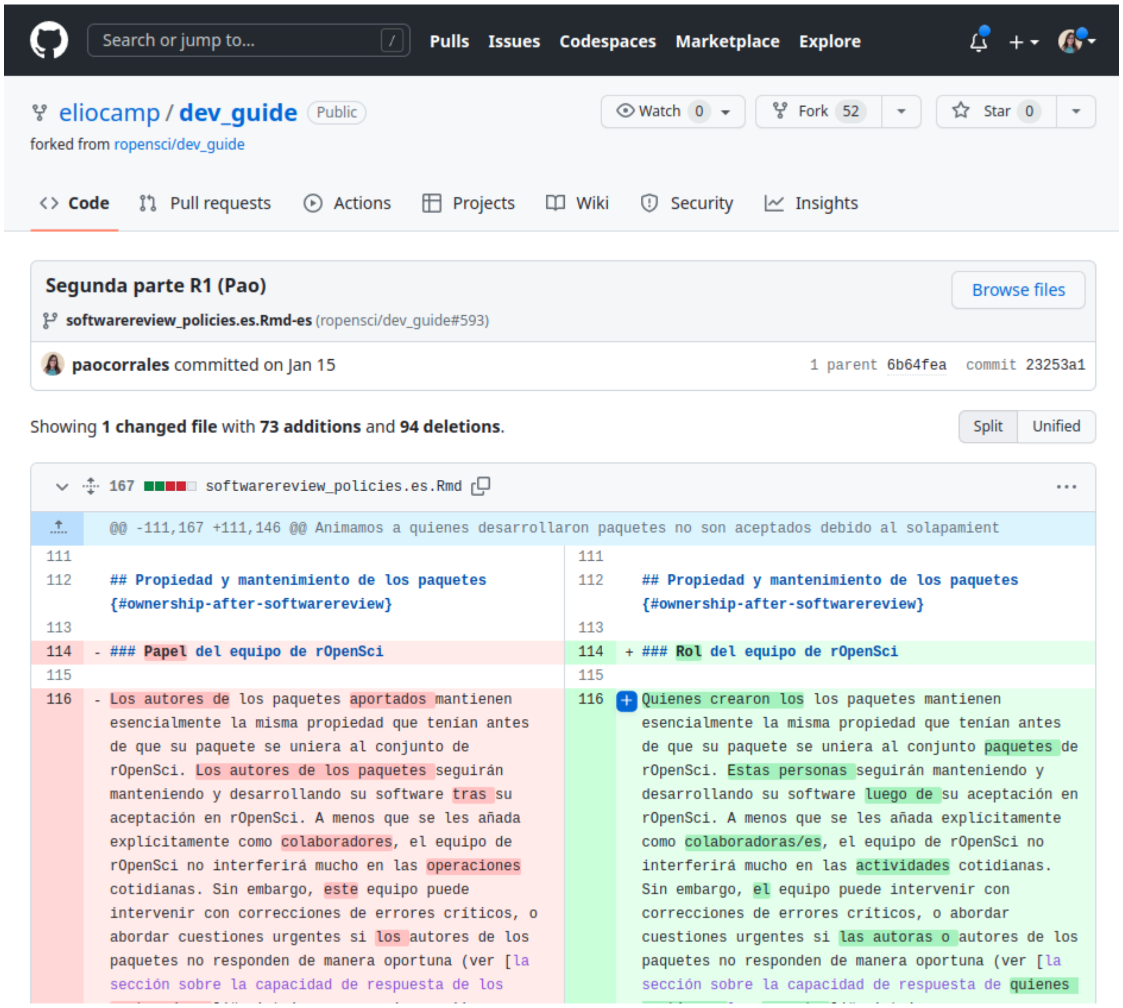
In order to achieve this, the following must be taken into account:

- Use terms according to the glossary and the specific context.
- Use the format defined by the community, for example, in the Spanish translation, use *italicized* format for words that remain in English.
- Apply style decisions missed by the first reviewer. For example, in Spanish try to replace any "las/los" with a phrase without gender marking.
- Check that sentences are generally understandable and sound good while avoiding literal translations.

In the process of performing the second review you may need to consult with the first reviewer about the decisions they made or with the rest of the community if there is a phrase or term that raises doubts. You may then suggest adding new terms to the glossary or updating this translation guide.

Working locally on your computer, here are the steps to follow when performing a second review of a file:

1. **Checkout the *branch*** associated to the file you are going to check.** For example, if you're helping to translate `softwarereview_policies.en.Rmd`, you would look for the `softwarereview_policies.es.Rmd-es` branch.
2. **Pull the changes.** Do a *pull* to download all the changes the first reviewer made to update your local copy.
3. **Start reviewing the text.** To do this, we suggest you open the file history to view the changes the first reviewer made. This will allow you to know if a phrase or word comes from the machine translation or from another human being.



Segunda parte R1 (Pao)

softwarereview_policies.es.Rmd-es (ropensci/dev_guide#593)

paocorrales committed on Jan 15 — 1 parent 6b64fea — commit 23253a1

Showing **1 changed file** with **73 additions** and **94 deletions**.

167 softwarereview_policies.es.Rmd

```
@@ -111,167 +111,146 @@ Animamos a quienes desarrollaron paquetes no son aceptados debido al solapamient

111                                                     111
112   ## Propiedad y mantenimiento de los paquetes       112   ## Propiedad y mantenimiento de los paquetes
      {#ownership-after-softwarereview}                       {#ownership-after-softwarereview}
113                                                     113
114 - ### Papel del equipo de rOpenSci                   114 + ### Rol del equipo de rOpenSci
115                                                     115
116 - Los autores de los paquetes aportados mantienen    116 + Quienes crearon los los paquetes mantienen
      esencialmente la misma propiedad que tenían antes       esencialmente la misma propiedad que tenían antes
      de que su paquete se uniera al conjunto de              de que su paquete se uniera al conjunto paquetes de
      rOpenSci. Los autores de los paquetes seguirán          rOpenSci. Estas personas seguirán manteniendo y
      manteniendo y desarrollando su software tras su         desarrollando su software luego de su aceptación en
      aceptación en rOpenSci. A menos que se les añada        rOpenSci. A menos que se les añada explícitamente
      explícitamente como colaboradores, el equipo de         como colaboradoras/es, el equipo de rOpenSci no
      rOpenSci no interferirá mucho en las operaciones        interferirá mucho en las actividades cotidianas.
      cotidianas. Sin embargo, este equipo puede              Sin embargo, el equipo puede intervenir con
      intervenir con correcciones de errores críticos, o      correcciones de errores críticos, o abordar
      abordar cuestiones urgentes si los autores de los       cuestiones urgentes si las autoras o autores de los
      paquetes no responden de manera oportuna (ver [la       paquetes no responden de manera oportuna (ver [la
      sección sobre la capacidad de respuesta de los          sección sobre la capacidad de respuesta de quienes
```

4. ***Commit* your changes.** Many times the content of the files will be extensive and you will need a lot of time to complete the revision. In these cases we suggest you make periodic *commits* at least after each work session indicating in the commit message which line of the file you have reached. This way, if you need to delegate the review to someone else, that person can review the file history and pick up where you left off.

5. ***Push* to the working repository.** So far the review only lives in your local repository, pushing it moves it online. It is very important that you *push* to the *branch* associated to the file you are working on, otherwise we will not be able to connect your review with the original translation.

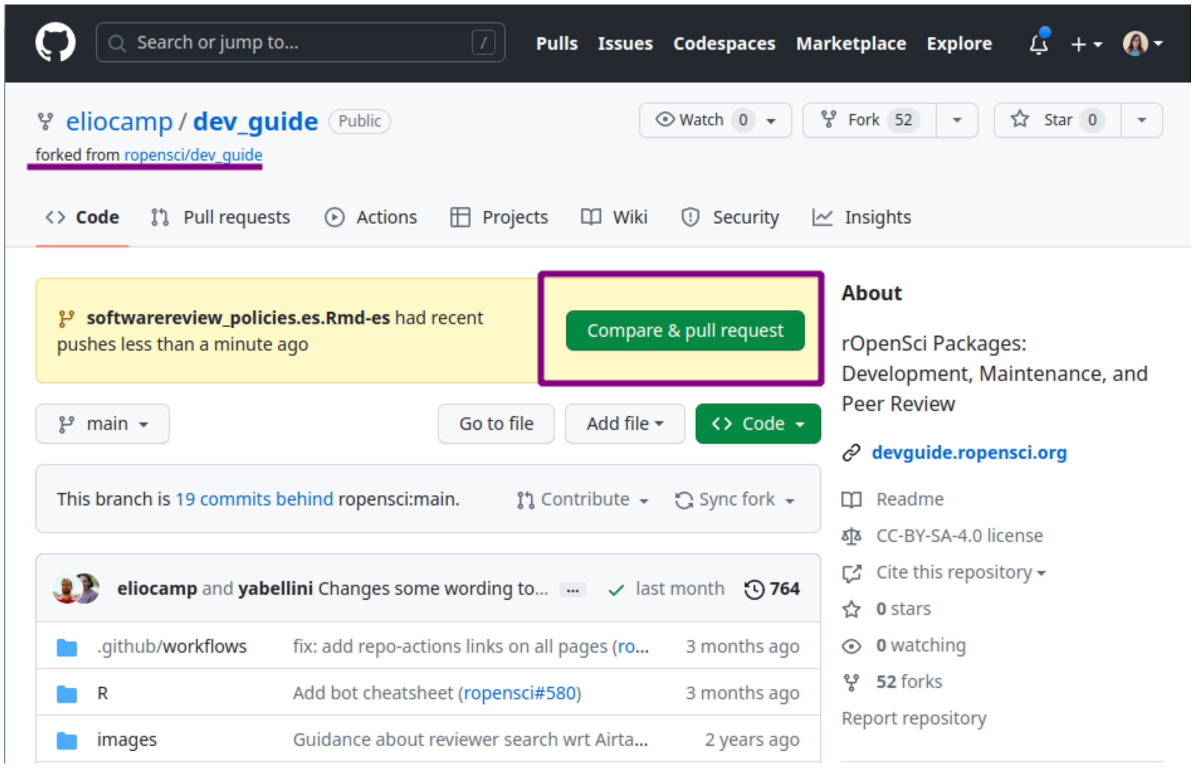
6. **Open a *pull request*** Once you have completed your revision, the chapter is ready! However, this new version only lives in the working repository. You will have to open a *pull request* in the main repository to have it reviewed and approved by someone in an editor role. In the next section we will see how to open the *pull request*.

3.2.5 Pull Request and final edits

Once the review process is complete, it is time to transfer the revised translation from the working repository to the main rOpenSci repository. For this, the second reviewer will have to open a *pull request* that compares the revised version with the original machine translation version.

1. **Check your repository and branch**. Make sure that all changes are in the working repository and that you are working on the *branch* associated to the file under review, for example `softwarereview_policies.es.Rmd-es`.

If you are on the correct *branch*, in the GitHub web interface you will see the following:

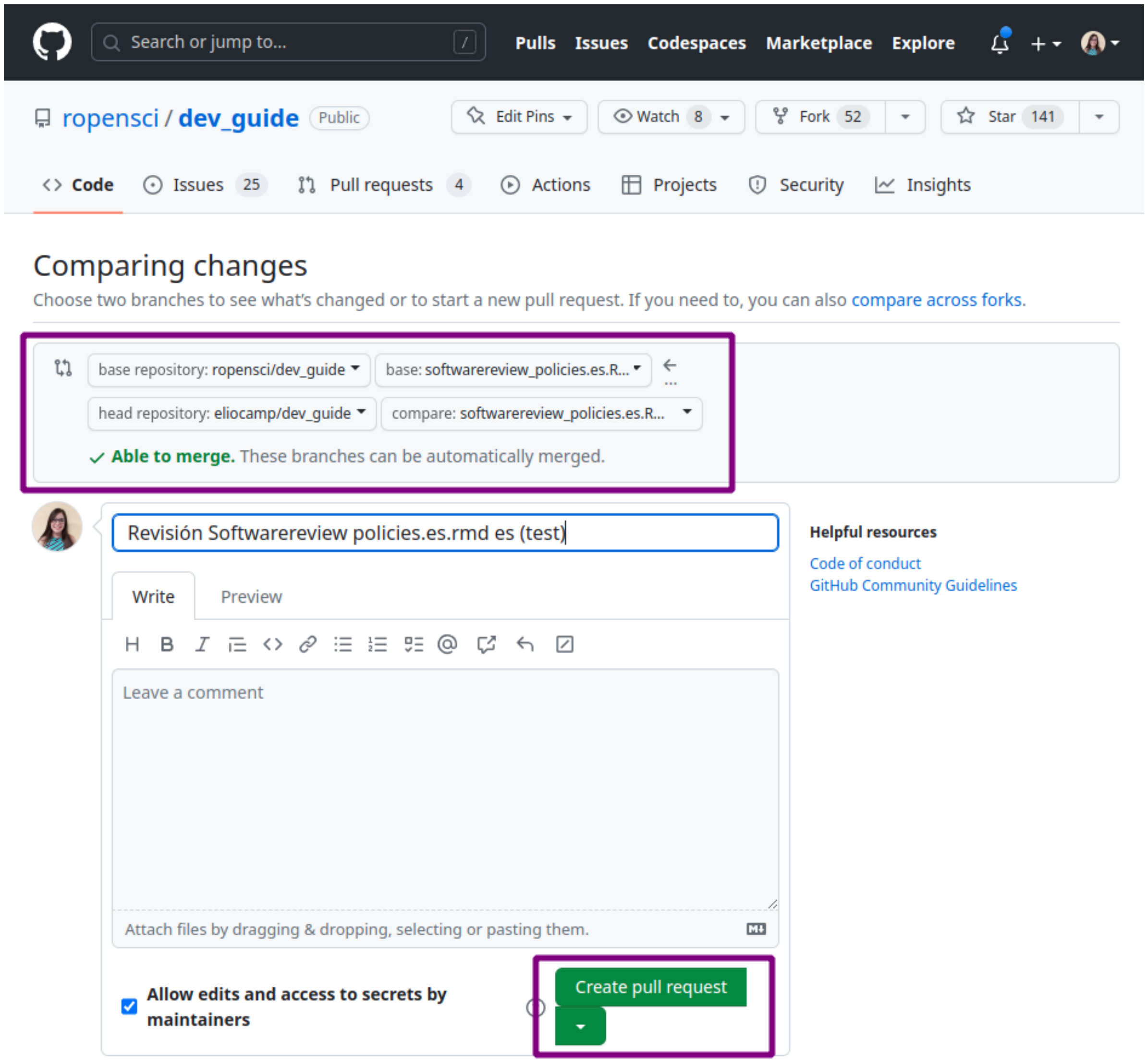

Search or jump to...
Pulls
Issues
Codespaces
Marketplace
Explore
eliocamp / dev_guide
Public
forked from ropensci/dev_guide
Watch 0
Fork 52
Star 0
Code
Pull requests
Actions
Projects
Wiki
Security
Insights
softwarereview_policies.es.Rmd-es had recent pushes less than a minute ago
Compare & pull request
main
Go to file
Add file
Code
About
rOpenSci Packages: Development, Maintenance, and Peer Review
devguide.ropensci.org
This branch is 19 commits behind ropensci:main.
Contribute
Sync fork
Readme
CC-BY-SA-4.0 license
Cite this repository
0 stars
0 watching
52 forks
Report repository
eliocamp and yabellini Changes some wording to... last month 764
.github/workflows fix: add repo-actions links on all pages (ro... 3 months ago
R Add bot cheatsheet (ropensci#580) 3 months ago
images Guidance about reviewer search wrt Airta... 2 years ago

GitHub will detect new changes (*pushes*) in the *branch* and will suggest comparing them and making a *pull request* to the main repository.

3. **Click on the "*Compare & pull request*" button.**
4. **Edit the *pull request*.** You will then have the option to modify the title of the *pull request* and leave a message. We suggest 2 things:

    1. Change the title to identify that it is a "Review". It is important to differentiate it from the *pull request* of the automatic translation.

2. In the body of the message, tag (use `@github-name`) the maintainer of the translation project so that they are aware of the progress and can then continue with the process.



Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base repository: ropensci/dev_guide
base: softwarereview_policies.es.R...
head repository: eliocamp/dev_guide
compare: softwarereview_policies.es.R...

✓ Able to merge. These branches can be automatically merged.

Revisión Softwarereview policies.es.rmd es (test)

Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting or pasting them.
Helpful resources
Code of conduct
GitHub Community Guidelines

✓ Allow edits and access to secrets by maintainers

5. **Double check the *branch*.** Before creating the *pull request* it is **very important** that you check that it is done from the correct *branch*. This is the only way to connect the review work with the automatic translation and to be able to unify the versions correctly.
6. **Create the pull request.** And that's it! Click on "*Create pull request*".

From this point on, the review is in the hands of the **maintainer** of the translation project. This person must review the *pull request* and make queries to first and second reviewers before *merging* the *pull requests.

3.3 General review

The review process is a collaborative process in which, we hope, many people will participate. This has the great advantage of incorporating different perspectives on the use of language and generating a translation that represents the greatest number of people. However, with many contributors, and despite the specific guidance of each language, it is possible that sometimes different criteria may be used or that there are simply typing errors.

For all these reasons, it is important that the translation project is completed with an overall review of the material. We strongly suggest that this be carried out by one person in order to maintain consistency throughout the text.

Now **the translation is ready** and rOpenSci staff will take it from here to ensure it is included along side the original material.

Thank you and congratulations!

4 Specific Language Guidelines

Note

This chapter is different for each language. It contains the specific guidelines for translating the content according to consensus in each language community.
To read the guidelines for a specific language, please change the version of the Translation Guide to that language using the language list on the left (under the symbol).
These guidelines apply when translating content to **English**.

4.1 Dialect

We will use the American English dialect in the translation.

4.2 Technical terms and citations

1. Maintain a list of terms that are translated and another list of terms that are not. We propose a initial list and we consult our community (Slack first, and then social networks if there is no consensus).
2. For bibliographic references, the original title/name is left in the original language, with the words in italics; a translation of the title in English is added in parentheses. When a reference to a Wikipedia entry, a Carpentries lesson, or other online resource occurs, the English version is added to the link if it exists. If one cannot be found, the reference is left in the original language.

4.3 Code

Do not translate function or argument names from packages, but do translate:

1. variables names;
2. constant names;

3. comments;
4. user function names and parameters names: if a code snippet creates a function as example, we can name the function and its parameters in English.

For example:

```
# Original code:

adentro <- function(punto, mas_bajo, mas_alto) {
        if (punto <= mas_bajo) {
            return(FALSE)
        } else if (punto >= mas_alto) {
            return(FALSE)
        } else {
            return(TRUE)
        }
   }

# Translation to English:

 inside <- function(point, lower, higher) {
        if (point <= lower) {
            return(FALSE)
        } else if (point >= higher) {
            return(FALSE)
        } else {
            return(TRUE)
        }
   }
```

4.4 Diagrams

1. Text in figures and diagrams should be translated.
2. Screenshots should be translated only if the IDE or UI has an English localization.

4.5 Data

If there is a translated version of the data, for example with the variable names in English, use that data.

If there is not, use the original data, mentioning what the variable names mean.

Finally, a project can be initiated to translate the data, or new English data, meaningful to the target audience, can be used to replace the original data (see project dados for an example of translation of data from English to Portuguese).

4.6 Glossary

This glossary is being developed as part of the translations of rOpenSci material. It supports the task of determining which technical terms are translated and which are not. It is also outlines which words we prefer to use where there are more than one option.

We use the following reference sources to generate our glossary and keep it up to date:

- Wikipedia
- Glosario
- the bilingual dictionary of Teaching Tech Together

5 NEWS