{geotargets}: Enabling geospatial workflow management with {targets}

Eric R. Scott ⓘ
Communcations & Cyber Technologies Data Science, University of Arizona; Tucson, AZ

Nicholas Tierney ⓘ
Telethon Kids Institute; Perth, Western Australia

Signatories

Project team

Nicholas Tierney will serve as primary author and maintainer of `geotargets` during its development. Eric Scott (University of Arizona) will be a main contributor of code. Both anticipating using `geotargets` in ongoing and upcoming projects and have strong motivation to contribute to it's development.

Contributors

We already have contributions in the form of detailed reproducible examples in GitHub issues. In particular, Andrew Gene Brown at USDA-NRCS, has contributed code for dealing with reading and writing shapefiles, and choosing alternative filetypes for targets. Anthony North at Queensland Fire and Emergency Services has contributed code for using `geoarrow` as a backend for reading and writing targets. Dewey Dunnington at Voltron Data has also made suggestions on using `geoarrow`. Michael Sumner and Ben Raymond at Integrated Digital East Antarctica program, Australian Antarctic Division, have agreed to provide support and guidance in handling and managing geospatial data formats, in particular navigating GDAL.

Consulted

The idea for `geotargets` originated in a discussion posted to the discussion forum for `targets` where the author and maintainer of `targets` and other "Targetopia" packages showed strong support.

Community contributions to `geotargets` via issues and comments on issues have been numerous despite the repository only having existed since the beginning of March 2024. Additionally, we plan to consult with geospatial experts throughout development to be sure we are addressing the most pressing issues and addressing them in ways that will fit with common geospatial analysis workflows in R.

The Problem

Geospatial computations in R are made possible by a group of R packages aimed at spatial and spatio-temporal data such as `terra`, `sf`, and `stars` (Hijmans, 2024; Pebesma & Bivand, 2023). Computations on spatial or spatio-temporal data can often be computationally intensive and slow when the underlying data is large (e.g. high-resolution global rasters). Depending on the data source and operation, this can range from taking seconds, to days, or even weeks. Managing complex geospatial workflows can be confusing and re-running entire data pipelines is likely to be very time-consuming. The `targets` R package aims to aid with confusing and long-running workflows by automatically detecting dependencies among steps and only re-running steps that need to be re-run (Landau, 2021). This seems like a natural fit for complex geospatial workflows in R. However, geospatial packages like `terra` and `sf` don't work well with `targets` without extensive customization.

One notable difficulty is that `targets` saves the R objects generated by computational steps to disk, by default to an .rds file. However, R objects generated by the `terra` package contain a C++ pointer to the data, rather than the data itself. When one of these `terra` objects is saved (e.g. as a .rds) and read back into R, it loses information about the data it represents and no longer works. To make these R objects portable and suitable for use with `targets` they need to be "marshaled" and "unmarshaled" requiring more complex code, for a custom format.

A second obstacle is that often geospatial data is written in multiple files. For example, shapefiles are actually a collection of up to 12 files with different extensions. This limits compatibility with `targets` because the intermediate objects stored in a `targets` pipeline are required to be single files with no file extension.

Both of these challenges (and others) have been solved in bespoke ways for individual projects, but to date these solutions have not been formalized and distributed as an R package. The `targets` package is, by design, extensible and a number of packages have been created to allow `targets` to work for specialized needs. For example, `stantargets` allows for using `targets` with Stan models (Landau, 2021). We hope that `geotargets` can be a widely used addition to the "Targetopia" of packages that extend `targets`. We believe this will unlock a powerful workflow management tool for a large group of R users that have previously been unable (or unwilling) to use it because of these challenges.

The proposal

Overview

Our goal is to create a package that makes using `targets` for geospatial analysis in R as seamless as possible. To that end, `geotargets` will provide custom functions for defining geospatial targets that take care of translating and saving R objects for the user. In addition, we will create vignettes demonstrating how to use various geospatial R packages with `targets`. Where appropriate, we

will identify contributions to existing R packages to make them easier to use with `targets` and `geotargets`.

Detail

In the `targets` package, analysis steps, or "targets", are defined with the `tar_target()` function. Targetopia packages provide additional `tar_*()` functions that extend `targets` by providing target archetypes for specialized workflows. A new `tar_*()` function requires that you specify reading, writing, marshalling, and unmarshaling of these new formats. Briefly, reading and writing could be thought of as functions that read and write data, like `read.csv()` and `write.csv()`. Marshalling and unmarshalling are to do with handling the objects as serialised data - similar to read write. The main contribution of `geotargets` will be a series of alternative `tar_*()` functions that create targets with pre-defined formats that take care of the details of how these R objects are written out and read in by downstream targets. For example, to write a target that creates a raster using the `terra` package, one would use `geotargets::tar_terra_rast(name, command)`. `tar_terra_rast()` would provide a pre-defined format created with `targets::tar_format()` with functions for reading, writing, marshaling, and unmarshaling `terra` `SpatRaster` objects. In this case, marshaling/unmarshaling involves running `terra::wrap()` and `terra::unwrap()`, respectively, to make the R object "self-contained" rather than just containing a C++ pointer to the data. This is especially necessary for parallel computing with `targets` since `SpatRaster` objects don't work outside of the R session they were created in without `wrap()`ing them first. Reading and writing the target, in this case, would be specified to happen via `terra::rast()` and `terra::writeRaster()`, respectively.

As a minimum viable product, we hope to deliver an R package, hosted on GitHub, supporting raster and vector data objects from the `terra` and `sf` packages with custom target functions. Support for additional geospatial packages will be added based on feedback from the user community and through consultation with geospatial specialists. In initial development we will choose sensible defaults for what file types targets will be stored as (e.g. GeoTIFF for raster data). In the future we will develop a `filetype` argument for each `tar_*` function, since there are many options for how geospatial data can be stored on disk by these packages. For example, "netCDF", "HEIF", and "BMP", and 161 other options listed in the GDAL raster driver. This will offer flexibility in light of trade-offs between file size, read/write speed, and dependency requirements similar to the existing options for how objects are stored by the `targets` package (i.e. default '.rds' with options for faster/smaller file types).

Project plan

Start-up phase

We have already created a geotargets package on GitHub that uses GitHub actions to run package checks. In the start-up phase, we will focus on making design decisions about what the package will offer and research and discuss what r-spatial packages and object types will we support. Answering these questions will lay the groundwork for efficient collaborative development of the package.

Technical delivery

Our goal is to deliver a package that allows users to use the `targets` package with various r-spatial packages (`terra`, `sf`, `stars`, etc.) with as little friction as possible.

Milestone 1: July 31

- Basic package with functionality for `terra SpatRaster` and `SpatVector` objects
- Well documented functions with high test coverage ensuring a solid start to the project
- Creation of a `pkgdown` website for the package hosted on GitHub pages

Milestone 2: September 30

- Add support for objects from a second r-spatial package such as `sf`
- Benchmarking of various file type options for storing targets including file size and read and write speed. Our findings will be published as a package vignette and an article on the `geotargets` website.

Milestone 3: November 30

- Add support for objects from a third r-spatial package.
- Prepare for submission to rOpenSci software review
- Prepare short paper for submission to Journal of Open Source Software (following rOpenSci software review)

Other aspects

Throughout the project we will seek feedback from users on social media (Mastodon and Twitter) and on the `targets` discussion forum. Each milestone will coincide roughly with a release on GitHub and a short blog post on https://www.njtierney.com/. We intend to submit the package to rOpenSci for software review, and subsequently submit for publication to the Journal of Open Source Software.

Requirements

People

Nicholas Tierney and Eric Scott will take the lead in developing code for this project. Both have experience developing R packages and working with `targets` as well as some experience with geospatial analysis in R. Additionally we plan to consult with geospatial experts who are more familiar with the "rough edges" related to doing geospatial analysis in R and with the `targets` package.

Processes

We will adopt a code of conduct and contribution guideline based on guidelines used by rOpenSci. Communication regarding development will happen primarily via GitHub issues, with additional communication not regarding code happening via the rOpenSci Slack where both Eric and Nicholas are members. We may hold occasional video meetings as well.

Funding

We request funding to pay for the time of the personnel working on this project.

- Eric Scott, 108 hours totaling $5,184
- Nicholas Tierney, 108 hours totaling $7,128

Additionally, we will consult with geospatial experts and would like to be able to pay them for their time:

- Consulting, 36 hours totaling $3600

For a total of: $15,912
Funding timeline

- After milestone 1 (July 31): $5,304
- After milestone 2 (September 30): $5,304
- After technical delivery (November 30): $5,304

Summary
These costs are a one-time investment to pay for initial development of the `geotargets` package totaling $15,912.

Success
In order to understand and evaluate the success of the `geotargets` project, it is worthwhile visiting our definitions of done and success, as well as exploring future work, and potential risks to the project.

Definition of done
Success in the `geotargets` project looks like supporting fundamental geospatial raster and vector operations for the `terra` and `sf` packages. We will include examples of using these targets workflows for `terra` and `sf` in the form of vignettes, as well as link to existing repositories that demonstrate varying degrees of complexity in usage of `geotargets`. See for example demo-geotargets, and map-ir-pipeline.

Measuring success
We will use github issues to identify key tasks that require completing. For example, identifying target components for rasters and vectors for the `terra` package were outlined in https://github.com/njtierney/geotargets/issues/12. The pull requests that resolved this were linked in the issue.

We will ensure the package is has high test coverage of over 80% (currently test coverage is 97%). This will make the package easier to maintain, as any breaking changes will be identified early. We will use github actions to ensure `geotargets` can be installed on windows, macOS, and linux. High test coverage and github actions together will indicate to users this package is reliable and well maintained, which will hopefully improve trust and user takeup.

Another way we will measure success is by community uptake. If the project is used by the community then we will know that the project is successful, because it means it is solving a real need for the people using it. We can measure this success via a few metrics such as GitHub stars,

downloads from CRAN, and citations to the software. Currently the project was created on 2024-03-03, and as of 2024-03-22 it has 35 stars. This indicates that the project is already gathering interest from the community.

We will regularly engage with the community to see that we are meeting their needs in geospatial workflow management. For example, posting on mastodon, responding to issues and discussions on github, and delivering talks on using geotargets to local R communities.

Future work

This work could be extended in the future by developing new geospatial target factories for other spatial formats. For example, formats such as geoarrow, vapour, and geoparquet.

Key risks

Some of the risks in this project are potential changes in geospatial technology. Although it is unlikely that projects such as GDAL will produce breaking changes, if they do, we will need to make substantial changes.

We have decided to only support currently maintained geospatial projects. For example, many people still use the `raster` package, but we have decided to avoid the risk of supporting `raster` as it is no longer actively maintained and has been superseded by `terra`.

We may encounter challenges that cannot be solved directly by `geotargets` and require changes to be made in dependencies such as `targets` or the geospatial packages we intend to support. If these cases arise, we can open issues or offer pull requests, but ultimately such changes will be out of our control. In these cases we will do our best to provide workarounds or offer documentation of such challenges to our users.

Another potential risk is key team members having other commitments arise that mean they may need to pause their work on the project temporarily. We can protect against this risk by having more than one key team member who can contribute to the work. This will reduce our "bus factor". We will also adopt professional standards for managing our project on git and GitHub. This means that we will regularly push our work onto GitHub, so the work will not be at risk of not being available by way of only being on someone's specific machine.

References

Hijmans, R. J. (2024). Terra: Spatial data analysis. https://CRAN.R-project.org/package=terra

Landau, W. (2021). The targets r package: A dynamic make-like function-oriented pipeline toolkit for reproducibility and high-performance computing. *Journal of Open Source Software*, *6*(57), 2959. https://doi.org/10.21105/joss.02959

Pebesma, E., & Bivand, R. (2023). Spatial Data Science: With applications in R, 352. https://doi.org/10.1201/9780429459016