2025-09-18

momer: Extending momepy to R users

Claudiu Forgaci
Department of Urbanism, Delft University of Technology

Elviss Dvinskis
Digital Competence Center, Delft University of Technology

Executive Summary

Urban morphometrics is an emerging field that leverages computational techniques and large open datasets to analyse and understand the physical form of cities at unprecedented scale. The Python open-source library `momepy` (Fleischmann 2019) has been instrumental in advancing this field as it has enabled researchers to carry out complex morphometric analyses and workflow with relative ease. Arguably, the potential of `momepy` reaches way beyond Python, and its adoption in the broader scientific community and practice could be significantly enhanced by integrating it with R.

`momer` is an R package meant to bridge this gap by providing a seamless interface to `momepy` functionalities within the R environment. This integration will empower R users to use the capabilities of `momepy` for urban morphometric analyses without needing to switch between programming languages. Support from the Rbanism community, the r-spatial community, and the Spatial Data Science across Languages community confirms the domain-specific relevance for R users and the potential cross-language impact of this project.

`momer` will be implemented as a wrapper around `momepy` using the `reticulate` package. The package will mirror `momepy`'s modules and core functionalities. The package will also include comprehensive documentation and vignettes to guide users through typical workflows. Our ambition is to set up a workflow triggered by every `momepy` release that automatically updates the documentation of `momer` to reflect any changes in `momepy`. This will ensure that `momer` remains up-to-date with the latest features and improvements in `momepy`. We expect this to be feasible considering the upcoming release of `momepy` v1.0.0, which is expected to have a stable API and documentation structure.

Signatories

Developer of `moempy` Martin Fleischmann, as well as members of the Rbanism community, represented by community leads Clémentine Cottineau, Daniele Cannatella and Claudiu Forgaci; the r-spatial community, represented by Tim Appelhans confirm the relevance of this project and its expected impact on the field of urban morphometrics and the R spatial ecosystem.

Project team

Claudiu Forgaci is assistant professor of urban design and analytics engaged in urban morphology research and experienced in R package development (see rcrisp, rcoins, and visor). He is also a member of the Spatial Data Science across Languages (SDSL) community with a special interest in cross-language projects. He is also one of the initiators of the Rbanism community promoting reproducibility, automation and scalability in urbanism research using R. Claudiu will lead the development of `momer` and will maintain it after completion.

Elviss Dvinskis is a research data engineer at the TU Delft Digital Competence Center (DCC) experienced in Python and R package development. With an extensive project history and expertise, the DCC provides in-kind support for researchers at TU Delft. Elviss will carry out the software engineering work planned in this project with a commitment of 0.4FTE for the planned duration of the project.

Consulted

ISC member **Josiah Parry** also member of the SDSL community, provided feedback on the proposal.

The Problem

Urban morphometrics is a growing sub-field of urban morphology that enables scalable quantitative analysis of urban form. The field of urban morphometrics is inherently programmatic and has been heavily driven and promoted by the Python library `momepy` (Fleischmann 2019). While `momepy` has played an important role in making the quantitative analysis of urban form accessible to many, it is limited to users of the Python programming language.

Moreover, a large part of urban morphology research is conducted with GUI-based tools and qualitative interpretation, which limits the kind of analytical questions that researchers and practitioners can ask. Automated, reproducible and scalable urban morphology research can only be effectively achieved in a programmatic way and `momer` is an opportunity to further promote those standards and enable quantitative research alongside `momepy`.

Currently, a number of R packages provide some functionality for morphometric analysis, but they do not meet the level of comprehensiveness of `momepy`. Most prominent examples are the `foot` package (WorldPop Research Group, University of Southampton 2021), which focuses on processing building footprints, and the `vectormetrics` package (Matuszek et al. 2024), which provides tools for landscape and shape metrics. The former is limited to building footprint analysis, while the latter is is only meant to analyse polygon data and is not specifically designed for the analysis of urban form elements.

> **Note**
>
> Are there any other packages which should be mentioned here?

The proposal

Overview

The project aims to develop `momer`, an R package that wraps the Python library `momepy` using the `reticulate` package and a mirrored documentation. `momer` will enable R users to perform urban morphometric analyses in a fully `sf`-compatible way, without needing to switch to Python, thereby broadening the reach of `momepy` and increasing the adoption of morphometric methodologies. The development of `momer` is timely as `momepy` is approaching a major release (v1.0.0) which will stabilise its API, making it easier to maintain the R wrapper in the long term. The project will be structured around four main milestones over a 12-month period, from initial setup and basic wrappers to final review and release. The project will follow the latest best practices for research software development and comply with CRAN requirements from the outset.

Detail

Minimum Viable Product

The minimum viable product is an R package that (1) wraps all core functions of `momepy`, (2) ensures reliable and correct bidirectional data conversion between Python `geopandas` and R `sf` objects, as well as Python `networkx` and R `sfnetwork` objects, (3) mirrors its documentation, and (4) includes a detailed documentation for maintainers on how to update the wrappers and documentation with future `momepy` releases.

Architecture

`momer` will wrap exported `momepy` functions with `reticulate` and will be organised into the same nine modules, namely: `elements`, `dimension`, `shape`, `distribution`, `intensity`, `diversity`, `connectivity`, `streetscape`, and `preprocessing`. All functions will be `sf`-compatible and bidirectional conversion between Python and R will be achieved in memory via WKB. We also aim to set up a workflow to automate the generation of `roxygen2` documentation to be used for the generation of a `pkgdown` website.

Assumptions

One of the main assumptions is that with the upcoming major release, the `momepy` API will remain stable, which will allow `momer` to be feasibly maintained. The need for a stable, non-conflicting Python environment also poses a challenge which we assume we can overcome with detailed documentation. We also assume that there is a feasible way of keeping the documentation of `momer` in sync with every `momepy` release.

External dependencies

The project primarily depends on the wrapped Python library `momepy` and the R package `reticulate` used for wrapping. The user needs to have Python installed with `momepy` and its dependencies.

Project plan

Start-up phase

To enable collaboration and contributions, we will set up the project on GitHub. The repository will include guidelines for contributors, a code of conduct, and a `README.md` file with an overview of the project.

A project team will be set up to track tasks and milestones, and issues will be used for feature requests and bug reports. Reporting will be done quarterly, with updates shared on the R Consortium blog and social media platforms to keep the community informed of progress.

We aim to release `momer` under the permissive Apache 2.0 license which is compatible with the BSD-3-Clause license of `momepy`. `momer` will only wrap `momepy`, i.e., call it at runtime, which means that no BSD-licensed code is redistributed and so mentioning the license in the documentation should suffice.

Technical delivery

The project will be structured by the following milestones:

1. Initial setup and basic wrappers (Month 1-4):
   - Set up the GitHub repository with contribution guidelines.
   - Set up R package skeleton with installation instructions.
   - Configure `reticulate` and document environment setup instructions.
   - Reuse or write reliable helper functions for bidirectional data conversion between Python `geopandas` and R `sf` objects, as well as Python `networkx` and R `sfnetwork` objects, also considering edge cases.
   - Benchmark processing overhead of data conversion layer.
   - Wrap 5-10 core `momepy` functions.
   - Test wrapped functions on sample dataset and workflow.
   - Add basic documentation.
2. Comprehensive wrappers and testing (Month 5-7):
   - Develop a suite of unit tests and integration tests to ensure reliability and correctness.
   - Expand wrappers to cover major functionalities of `momepy`.
   - Ensure compatibility with tidy workflows.
   - Draft detailed documentation for each wrapped function.
3. Documentation and vignettes (Month 8-10):
   - Set up documentation synchronisation workflow between `momepy` and `momer`.
   - Generate comprehensive documentation, including examples.
   - Set up `pkgdown` website with API reference.
   - Create vignettes demonstrating typical workflows and applications of `momer`.
   - Conduct user testing and gather feedback for improvements.
4. Final review, maintenance setup, and release (Month 11-12):
   - Address any issues or feedback from user testing.
   - Perform a final review of the code base and documentation.
   - Set up GitHub Actions for CI.

- Prepare for the official release of `momer` on CRAN and GitHub.
- Publicize the release through blog posts, social media, and relevant forums.
- Disseminate in the developer community (e.g., SDSL, ISUF, Rbanism) to attract contributions.

Other aspects

The project will be open-source, hosted on GitHub, and licensed under the Apache 2.0 license. This will enable broad use and contributions from the community.

Updates on progress and milestones achieved will be shared on the R Consortium blog and on the Rbanism community's blog on a quarterly basis. Updates will also be shared on the social media platforms Mastodon (Fosstodon), LinkedIn and Bluesky. To promote the wide adoption of `momer`, we will announce its release at the UseR! conference and other similar events where the R spatial community is present. We also aim to introduce the package in the International Seminar of Urban Form (ISUF) where most of the research community conducting urban morphology analysis is present.

Budget & funding plan

The requested budget will be used for labor costs divided by period, as follows:

1. Initial set-up and basic wrappers (Month 1-4): $
2. Comprehensive wrappers and testing (Month 5-6): $
3. Documentation and vignettes (Month 7-10): $
4. Final review and release (Month 11-12): $

Total: $

Success

Definition of done

A successful project will deliver a functional and well-documented `momer` package. The package will provide R users with access to the functionalities of the Python library `momepy`, enabling them to perform urban morphometric analyses within R.

Measuring success

Success can be measured against the following indicators:

- Timely completion of all project milestones;
- Positive feedback from user testing and community engagement;
- Adoption of the `momer` package by the R community, as seen in the number of downloads, citations, and contributions;
- Successful integration and functionality of all wrapped `momepy` features;
- Successful synchronisation of the documentation between `momepy` and `momer` triggered at every new `momepy` release;
- Synchronised, comprehensive and clear documentation, including vignettes and examples, aligned with CRAN policies;

- Active post-release maintenance and updates, reflecting ongoing community needs and developments in the field of urban morphometrics.

Future work

The `momer` project is instrumental for the development of a cross-language infrastructure as it will grow a critical mass of users beyond Python and will help identify pressing high-level cross-language challenges that can serve other similar projects as well.

Fleischmann, Martin. 2019. "Momepy: Urban Morphology Measuring Toolkit." *Journal of Open Source Software* 4 (43): 1807. https://doi.org/10.21105/joss.01807.

Matuszek, Tomasz, Jakub Nowosad, Marco Sciaini, Maximillian H. K. Hesselbarth, and Yunyao Ma. 2024. *Vectormetrics: Landscape Metrics for Categorical Map Patterns in Vector Data*.

WorldPop Research Group, University of Southampton. 2021. *Foot: An r Package for Processing Building Footprint Morphometrics*. https://github.com/wpgp/foot.