SoftwareX 35 (2026) 102870

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original Software Publication

ETHOS.GeoKit: A Python toolkit for analyzing and altering geospatial data for energy systems modeling and beyond

Shitab Ishmam [a,b,1], Julian Belina [a,b,1,*], Christoph Winkler [a,b], Jann M. Weinand [a], Noah Pflugradt [a], Heidi Heinrichs [a,c], Jochen Linßen [a]

[a] Forschungszentrum Jülich GmbH, Institute of Climate and Energy Systems– Jülich Systems Analysis (ICE-2), 52425, Jülich, Germany
[b] RWTH Aachen University, Faculty of Mechanical Engineering, 52062, Aachen, Germany
[c] University of Siegen, Chair for Energy Systems Analysis, Department of Mechanical Engineering, 57076, Siegen, Germany

ARTICLE INFO

Keywords:
Geospatial data processing
Geospatial workflow automation
Land eligibility analysis
Renewable energy potential
Combined raster and vector analysis
Region-based masking

ABSTRACT

This article introduces ETHOS.GeoKit (hereafter GeoKit), a Python-based toolkit for analyzing and modifying geospatial data across various fields. GeoKit's primary strength is its ability to combine vector and raster data sources within a unified interface with low boilerplate code – a frequently recurring task, especially in global renewable energy potential analysis. Originally developed to facilitate land eligibility analysis for determining potential locations for renewable energy plants, GeoKit can be applied to any kind of georeferenced data. It provides functions for loading, analyzing and modifying both vector and raster datasets, as well as for converting between the two data types. Building upon the Geospatial Data Abstraction Library (GDAL), GeoKit offers a simpler, high-level interface that makes geospatial data processing more accessible.

Metadata

Nr	Code metadata description	*Metadata*
C1	Current code version	*v1.9.0*
C2	Permanent link to code/repository used for this code version	https://github.com/FZJ-IEK3-VSA/geokit
C3	Permanent link to reproducible capsule	N/A
C4	Legal code license	MIT
C5	Code versioning system used	Git
C6	Software code languages, tools and services used	Python
C7	Compilation requirements, operating environments and dependencies	Conda or mamba as package managers, these are used to install jupyter, Descartes, pip, matplotlib, pandas, numpy, scipy, scikit-learn, gdal, Smopy, geopandas, psutil, structlog, rich, pandas-stubs, scipy-stubs, tqdm, pytest, pytest-xdist, pytest-cov, ruff, typeguard, nbconvert, jupyter-book, sphinx-autoapi, asteroid, codespell and nbval. The tool is tested on Ubuntu, Windows, and macOS., Windows, and macOS.
C8	If available, link to developer documentation/manual	https://geokit.readthedocs.io/
C9	Support email for questions	j.belina@fz-juelich.de, s.ishmam@fz-juelich.de

(*continued on next column*)

(*continued*)

1. Motivation and significance

Geospatial data plays a vital role in environmental research, climate modeling, and energy systems analysis. However, integrating diverse geospatial datasets into consistent analytical workflows poses a significant methodological challenge. Existing Python tools, such as GDAL/OGR [1], GeoPandas [2], and Rasterio [3], each provide specialized functionalities for either vector or raster data, but they operate largely in isolation. As a result, researchers often need to implement extensive pre-processing routines to utilize information from one geospatial dataset to analyze another one — efforts that can be repetitive, error-prone, and time-consuming. The scale of this challenge is illustrated by representative studies: Ryberg et al. [4] integrated nine heterogeneous geospatial data collections, including land cover, elevation,

* Corresponding author at: Forschungszentrum Jülich GmbH, Institute of Climate and Energy Systems– Jülich Systems Analysis (ICE-2), 52425, Jülich, Germany.
E-mail address: j.belina@fz-juelich.de (J. Belina).
[1] These authors contributed equally to this work

https://doi.org/10.1016/j.softx.2026.102870
Received 20 February 2026; Received in revised form 25 June 2026; Accepted 5 July 2026
Available online 10 July 2026

hydrological, and protected area datasets, to evaluate 36 land eligibility constraints for renewable energy across Europe, while Ishmam et al. [5] and Winkler et al. [6] applied a similarly diverse set of data collections to assess 33 eligibility criteria across Sub-Saharan Africa. Harmonizing such a large number of datasets from different sources, formats, and resolutions into a consistent analytical framework is highly laborious when relying on existing libraries alone, and often translates into additional costs in research projects.

GeoKit, the Geospatial toolkit for Python, addresses this challenge by offering a unified and high-level interface for handling both vector (points, lines, polygons) and raster (gridded) datasets within a single Python package. It harmonizes various geospatial data sources—such as land-use, topographical, meteorological, and infrastructure datasets—into a common analytical framework suitable for energy systems modeling and other applications requiring spatially explicit information. The toolkit builds upon GDAL's robust low-level capabilities, transforming them into an intuitive set of Python functions and object abstractions that eliminate redundant setup steps and reduce the likelihood of user error. GeoKit is part of the **ETHOS** model suite (**E**nergy **T**ransformation Pat**H**way **O**ptimization **S**uite) for systematically analyzing energy systems.

Beyond this unified interface, GeoKit accepts spatial inputs in a variety of formats (e.g., coordinate tuples, WKT strings, Pandas DataFrames) and abstracts common GDAL patterns into reusable functions, significantly reducing the boilerplate required for repetitive workflows that combine raster and vector data. To illustrate this concretely, the North Sea example introduced in Section 3 has been reimplemented using GeoPandas and Rasterio and as a pure GDAL/OGR workflow. As shown in Table 1, the same data processing steps require 2.93× and 3.64× more lines of code respectively, compared to the GeoKit implementation. Full implementations of all three workflows are available in the GeoKit documentation.

From the user's perspective, the workflow is straightforward. A researcher defines a RegionMask or Extent using either a vector dataset (e.g., administrative boundaries) or a raster layer (e.g., land cover map), loads additional datasets through built-in functions, and performs common spatial operations—such as overlaying, rasterizing, or buffering—using intuitive, high-level commands. Outputs can be saved to disk in widely supported formats such as Shapefile, GeoPackage, GeoTIFF, or GeoJSON, making them compatible with both scientific modeling frameworks and conventional GIS software. The consistent object structure and integration with NumPy and Pandas also allow for direct embedding of GeoKit outputs into quantitative models, effectively bridging geospatial analysis and numerical computations.

GeoKit has already made significant contributions to multiple research initiatives in large-scale renewable energy modeling and geospatial data science. It underpins recent studies on mapping green hydrogen potential and optimizing infrastructure in Sub-Saharan Africa (e.g., Ishmam et al. [5], Winkler et al. [6]) and is utilized in ongoing projects such as the IEA Global Hydrogen Reviews [7,8], where it facilitates the combination of global land, climate, and energy generation infrastructure datasets to determine hydrogen generation potentials. Other frameworks, such as *ETHOS.GLAES* [9] *and ETHOS.RESKit* [10] focus on energy system applications and use GeoKit internally for spatial preprocessing, underscoring its role as a core enabling layer in geospatial model development. These studies also demonstrate that GeoKit has been successfully deployed on HPC infrastructure for continental- and global-scale analyses.

Table 1
Lines of Code Comparison for an Application Example implemented in GeoKit, Rasterio and GeoPandas and in pure GDAL/OGR.

Implementation	Lines of Code	
	Data processing	Relative to GeoKit
GeoKit	28	1.00×
GeoPandas + Rasterio	82	2.93×
Pure GDAL/OGR	102	3.64×

2. Software description

GeoKit provides functionality to create, load, analyze, and export geospatial data within traceable Python scripts. Geospatial data generally exists in two fundamental forms: vector and raster. Vector data represent spatial geometries based on coordinate pairs within a defined coordinate reference system, linking features to specific real-world locations. Typical examples include points, lines, and polygons, each of which can be associated with arbitrary attribute information. In contrast, raster data consist of regularly spaced grid cells that store numeric or binary values representing spatially continuous phenomena, such as elevation or land cover. To associate raster data with a geographic location, metadata describing its spatial extent, resolution, and coordinate system are required. The need to handle these two data structures—individually and in combination—forms the foundation of GeoKit's architecture, which is described in the following section.

2.1. Software architecture

GeoKit is organized as a modular stack that abstracts low-level GDAL/OGR functionality into a small set of composable Python modules and high-level classes. The base layer consists of util.py for common helpers and custom errors, alongside srs.py, which centralizes all Spatial Reference System (SRS) loading, transformation, management, and convenient equal-area presets (e.g., Europe-centric Lambert azimuthal equal-area projection (LAEA)), ensuring projections are valid and consistent across workflows.

Fig. 1 depicts the main modules of GeoKit, which are built on top of previous Python modules. geom.py includes geometry-centric operations that operate independently of file input and output. These operations encompass creation, reprojection, buffering, and conversions between raster masks and vector polygons (e.g., using functions such as polygonizeMask / polygonizeMatrix). These building blocks are reused consistently by higher layers whenever vector–raster interoperability is required, as depicted in Fig. 1.

On top of this structure are two parallel modules dedicated to data I/O and processing: raster.py handles all grid-based operations like loading, creating, warping, and extracting a matrix from raster data, while vector.py manages feature-based operations, including loading, filtering, creating, and mutating vector datasets. These two modules

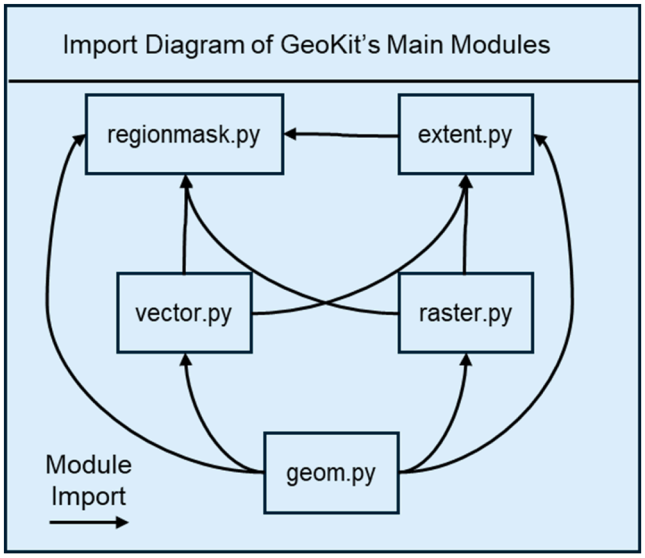

Fig. 1. Main software architecture of GeoKit.

provide the essential bridge between data types through functions like rasterize and polygonizeRaster.

This entire stack is orchestrated by a high-level abstraction layer. The Extent class defines a specific geospatial bounding box and spatial reference system, acting as a controller to execute operations from the raster and vector modules within its defined boundaries. The primary user-facing class used in many GeoKit workflows, RegionMask, combines an Extent with a numpy boolean mask or a polygon to represent a "region of interest". It maintains consistent context (bounds, SRS, and pixel resolution) and acts as a facade, simplifying complex workflows like indicateValues or indicateFeatures by calling the appropriate methods from the raster, vector, and geom modules. Additionally, location.py provides simple, high-level Location and LocationSet classes, which abstract point-based geometries for ease of use in functions like interpolateValues. The four core functionalities of GeoKit are presented in the next section.

GeoKit includes a comprehensive test suite built on pytest, with parallel execution supported via pytest-xdist and coverage reporting via pytest-cov, currently achieving 74% code coverage. Tests are integrated into a GitHub Actions CI/CD pipeline and can be run locally using standard pytest commands, allowing users to validate the software's correctness in their own environments.

Test cases were written for most of GeoKit's functionality to pin its output to known, expected results and thereby guard against errors. Because GeoKit delegates its core geospatial operations, including reprojection, warping and resampling, rasterization, polygonization etc., to the well-established GDAL/OGR and OSR libraries, the numerical and spatial correctness of these operations is largely inherited from mature, independently validated software. GeoKit's tests therefore focus on verifying that its orchestration of these libraries produces the expected results. Beyond this automated suite, the software has been validated extensively through use: the datasets produced for the studies presented in Section 4 were carefully reviewed, and where bugs or inconsistencies were identified they were corrected in GeoKit and the affected test cases updated accordingly. The resulting reference outputs are checked across successive versions of GeoKit and its dependencies, ensuring that behavior remains as consistent as possible as the software and its underlying libraries evolve.

2.2. Software functionalities

The primary purpose of GeoKit is to enable the analysis and integration of multiple geospatial data sources within a unified framework; the main workflow is illustrated in Fig. 2. Geospatial datasets can be loaded, created, analyzed, and exported in a variety of formats, allowing users to seamlessly combine diverse inputs such as shapefiles, GeoTIFFs, or in-memory datasets. In addition to importing existing data, GeoKit allows users to generate simple geometries or raster layers directly within their analysis scripts, which is particularly useful for testing, spatial masking, or defining custom regions of interest.

A key distinguishing feature of GeoKit is its ability to handle both vector and raster data simultaneously through the RegionMask module. This module defines an abstract region that integrates multiple datasets within a shared spatial context. Raster data can be warped or resampled to match a vector-defined region, enabling direct overlay and comparative analysis between datasets, while multiple vector datasets can be combined or intersected and complex geometric queries can be performed on raster layers. These operations support arithmetic (e.g., addition, subtraction) or logical (e.g., intersection, exclusion) combinations of spatial information. Moreover, GeoKit provides robust conversion utilities between raster and vector formats—such as polygonizing raster regions or rasterizing vector geometries — ensuring consistent interoperability between data types.

Almost all operations in GeoKit 1.9.0 are built upon the feature-rich GDAL 3.10 library. GeoKit exposes the core functionality of these operations and allows all GDAL-supported options to be passed through. This enables fine-grained control over results, such as no-data treatment, while providing sensible default settings. Full details are documented in GeoKit's docstrings and its online documentation.

The warp and interpolate functions offer a variety of algorithms. For warping, the algorithm is implemented by GDAL, while interpolation is built on SciPy (versions 1.10–1.18). The appropriate method depends on the input data and the intended application. The algorithms differ in how strongly they smooth neighboring data, how well they preserve extreme values, and how no-data values propagate into the results. Point-based algorithms such as “near” and “bilinear” mark output pixels as no-data whenever any contributing input pixel is no-data. Aggregating algorithms such as “average” and “rms” disregard no-data values in the input and compute results from the remaining valid pixels.

Full documentation, tutorials, and examples are available at https://geokit.readthedocs.io. The main functionalities of GeoKit are demonstrated in the following section through a practical example workflow.

3. Illustrative example

To demonstrate GeoKit’s ability to analyze and combine heterogeneous geospatial datasets, we present an example estimating the foundation depth of offshore wind turbines in the North Sea. This information is critical for assessing the investment costs and technical

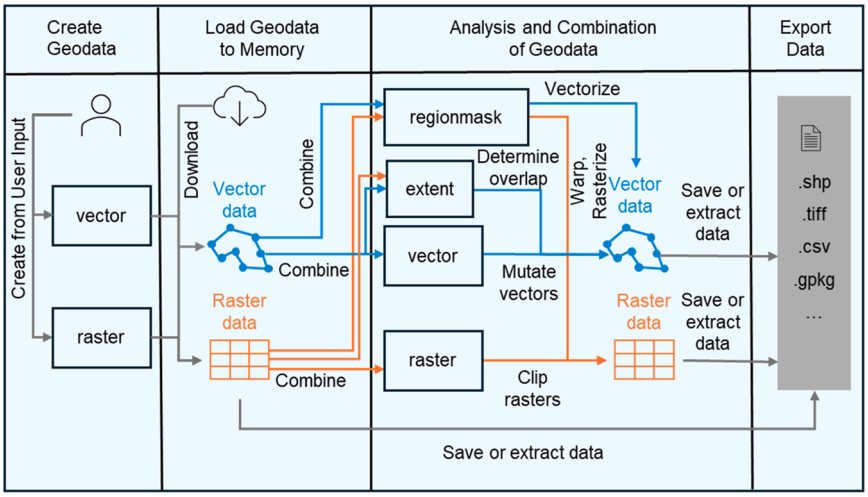

Fig. 2. Depiction of the main workflows with GeoKit.

feasibility of offshore wind energy projects. The example was selected because it requires integrating data from multiple sources and formats — exactly the kind of multi-source harmonization GeoKit is designed to streamline.

Three data sources were used in this analysis. The first is the International Hydrographic Organization (IHO) Sea Areas shapefile [11], which provides global marine boundaries and is used here to extract the North Sea as the area of interest. The second dataset is the GEBCO (General Bathymetric Chart of the Oceans) 2020 dataset [12], which provides high-resolution global bathymetric data in raster format. These elevation values are used to estimate the seabed depth and, consequently, the required foundation height of offshore wind turbines. The third dataset consists of plausible offshore wind turbine locations for the North Sea in 2050, as proposed by Waldman et al. [13]. This dataset provides point-based turbine coordinates that serve as the query points for the corresponding sea-depth values, which then inform subsequent analyses of the economic implications of different sea depths for specific wind turbine projects.

In the GeoKit workflow, the IHO shapefile is first loaded and filtered to isolate the North Sea basin, since the original file contains all global sea areas. The extracted region is then used to create a RegionMask, defining the spatial boundary of the North Sea basin for subsequent operations. The extracted North Sea basin is illustrated in Fig. 3.

In the next step, the GEBCO raster data is loaded and filtered to focus on the North Sea area. Since the global bathymetry dataset is divided into multiple raster tiles, overlapping tiles are first identified and clipped using the extracted sea basin shapefile. All relevant tiles are then merged into a single, continuous raster layer. Using the RegionMask module, the merged raster is warped to match the extent and coordinate reference system of the North Sea basin vector. The resulting spatially aligned bathymetry is shown in Fig. 4.

The third step determines the depth at the offshore turbine locations. GeoKit applies an interpolation step to estimate depth values between neighboring GEBCO raster cells because the turbine points do not exactly coincide with raster grid cells. Fig. 5 shows the combined datasets, where turbine locations are overlaid on the bathymetric surface. The resulting depth information for all turbine sites can then be exported as a Pandas DataFrame [14] and saved in common tabular formats (e.g., .csv or .xlsx) for further techno-economic analysis or visualization.

4. Impact

GeoKit has had a significant impact on the practice of geospatial data processing in scientific modeling by enabling simplified workflows for analyses that would otherwise demand heavy manual preprocessing.

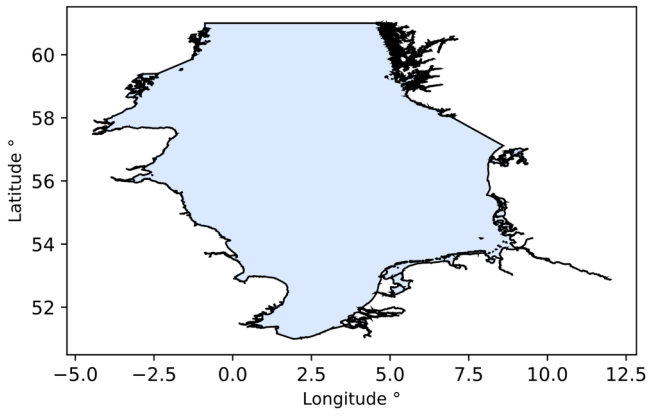

Fig. 3. Depiction of the North Sea basin based on the IHO (International Hydrographic Organization) [11] in the coordinate system EPSG (European Petroleum Survey Group) 4326.

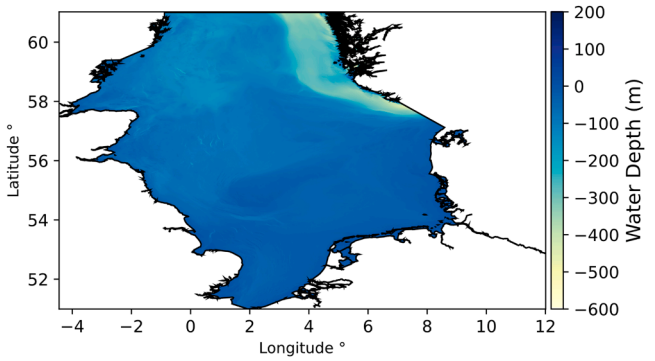

Fig. 4. Depth of the North Sea basin based on the GEBCO (General Bathymetric Chart of the Oceans) 2020 dataset [12] and the data from Fig. 3.

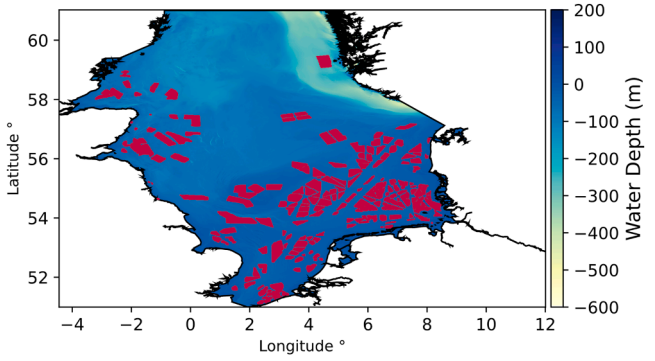

Fig. 5. Depiction of "Plausible 2050 offshore wind locations in the North Sea" by Waldman et al. [13] and the data used in Fig. 4.

The software directly addresses one of the most time-consuming and error-prone aspects of spatially explicit modeling—data harmonization—and, in doing so, has opened new avenues for scientific inquiry, faster analysis, and methodological rigor.

GeoKit has become a fundamental tool for geospatial analysis in energy system modeling, underpinning numerous recent studies. As already mentioned in Section 1, it has been used to quantify green hydrogen production potentials in Africa [5,6], where its raster warping and RegionMask-based spatial filtering were applied to harmonize multi-resolution and multi-format land-use and infrastructure datasets across national boundaries. In Ryberg et al. [4], GeoKit's rasterization and vector extraction functionalities were used to develop the *Geospatial Land Availability for Energy Systems (ETHOS.GLAES)* model [9], enabling the uniform assessment of land eligibility constraints for renewable energy sources across Europe. Similarly, Ryberg et al. [15] used GeoKit to preprocess multi-year weather datasets—harmonizing projections, resolutions, and regional subsets—which fed the ETHOS.RESKit [10] workflow for estimating onshore wind power potential. Key steps included loading onshore wind turbine placements, region-based warping, tiling/subsetting, and vector–raster sampling to produce consistent inputs across historical weather years. GLAES and RESKit are themselves used in most analyses of the ETHOS model suite, for example as parts of renewable energy potential assessment workflows (e.g., ETHOS.REFLOW [16,17]) or as input for holistic energy system optimizations [18,19].

In Caglayan et al. [20], GeoKit's vector–raster interoperability, buffering, and coordinate transformation capabilities were used to identify suitable geological formations for hydrogen storage in salt caverns, ensuring consistent integration of geospatial layers such as salt deposit maps, land cover, and infrastructure. Caglayan et al. [21] used GeoKit to preprocess and harmonize vector network layers—specifically European road networks and existing pipeline corridors—by clipping

them to the study extent, aligning coordinate reference systems, and preparing candidate grids. These inputs were then used to design hydrogen transmission networks within a fully renewable European energy system.

GeoKit's primary strength is its ability to ensure consistent treatment of geospatial inputs across diverse geographical regions, offering a lower barrier to entry for users. By standardizing projections, resolutions, and data formats, GeoKit not only enables robust cross-regional comparisons that were previously infeasible but also allows users to process a large volume of diverse datasets by eliminating repetitive preparatory steps. By providing a unified framework for spatial preprocessing, GeoKit allows researchers to explore scientific questions that were previously constrained by data heterogeneity or preprocessing limitations. It enables high-resolution, regionally consistent analyses of renewable resource availability, land eligibility, and infrastructure deployment, thereby supporting new lines of inquiry in energy system modeling, spatial analysis, and land–climate interactions. GeoKit is crucial here because it allows energy system modelers to leverage detailed geospatial data without needing to become experts in complex geospatial data handling or spending valuable time on laborious data preparation.

5. Conclusions

GeoKit has proven its worth by making geospatial analysis more easier to conduct and understandable, which is crucial for providing high-resolution assessments of renewable energy systems – a key challenge in the ongoing energy transition. Its flexible design ensures that it is not limited to energy systems analysis; instead, it serves as a versatile and consistent tool for any workflow involving georeferenced data. This makes it an invaluable resource for tackling diverse issues, including energy systems, land-use competition, environmental protection, and urban development, among others.

Declaration of generative AI and AI-assisted technologies

During the preparation of this work, the author(s) used Grammarly and DeepLWrite to perform grammar checks and improve sentence structure. Additionally, GitHub Copilot and Claude Code was used to suggest code improvements during the development of GeoKit. Following the use of these tools, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the final content of the publication.

CRediT authorship contribution statement

Shitab Ishmam: Writing – original draft, Visualization, Software, Methodology, Conceptualization. **Julian Belina:** Writing – original draft, Visualization, Software, Conceptualization. **Christoph Winkler:** Writing – review & editing, Software, Methodology, Conceptualization. **Jann M. Weinand:** Writing – review & editing. **Noah Pflugradt:** Writing – review & editing, Supervision. **Heidi Heinrichs:** Writing – review & editing, Supervision, Funding acquisition. **Jochen Linßen:** Writing – review & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work received primary support from the Helmholtz Association through the Joint Initiative "Energy System 2050: A Contribution of the Research Field Energy" and the program "Energy System Design". Additionally, parts of this work were supported by the H2Atlas-Africa project (03EW0001), funded by the German Federal Ministry of Research, Technology, and Space (BMFTR). The authors would like to thank Severin Ryberg for his central contributions to the initial development of the tool.

References

[1] E. Rouault, F. Warmerdam, K. Schwehr, A. Kiselev, H. Butler, M. Łoskot, T. Szekeres, E. Tourigny, M. Landa, I. Miara, B. Elliston, K. Chaitanya, L. Plesea, D. Morissette, A. Jolma, N. Dawson, D. Baston, C. de Stigter, H. Miura, GDAL, (2025). https://doi.org/10.5281/ZENODO.17098582.

[2] Joris Van den Bossche, Kelsey Jordahl, Martin Fleischmann, Matt Richards, James McBride, Jacob Wasserman, Adrian Garcia Badaracco, Alan D. Snow, Pieter Roggemans, Brendan Ward, Jeff Tratner, Jeffrey Gerard, Matthew Perry, Mike Taves, carsonfarmer, Geir Arne Hjelle, Ray Bell, Ewout ter Hoeven, Micah Cochran, Nicholas YS Tan, rraymondgh, Giacomo Caria, Lucas Culbertson, Matt Bartos, Sergio Rey, John Flavin, Nick Eubank, sangarshanan, Sean Gillies, geopandas/geopandas: v1.1.1, (2025). https://doi.org/10.5281/ZENODO.15750510.

[3] S. Gillies, others, Rasterio: Geospat raster I/Python Program, (2013). https://github.com/rasterio/rasterio.

[4] Ryberg DS, Robinius M, Stolten D. Evaluating land eligibility constraints of renewable energy sources in Europe. Energies 2018;11:1246. https://doi.org/10.3390/en11051246.

[5] Ishmam S, Heinrichs H, Winkler C, Bayat B, Lahnaoui A, Agbo S, Pena Sanchez EU, Franzmann D, Oijeabou N, Koerner C, Michael Y, Oloruntoba B, Montzka C, Vereecken H, Hendricks Franssen H, Brendt J, Brauner S, Kuckshinrichs W, Venghaus S, Kone D, Korgo B, Ogunjobi K, Chiteculo V, Olwoch J, Getenga Z, Linßen J, Stolten D. Mapping local green hydrogen cost-potentials by a multidisciplinary approach. Int J Hydrog Energy 2024;87:1155–70. https://doi.org/10.1016/j.ijhydene.2024.08.501.

[6] Winkler C, Heinrichs H, Ishmam S, Bayat B, Lahnaoui A, Agbo S, Peña Sanchez EU, Franzmann D, Oijeabou N, Koerner C, Michael Y, Oloruntoba B, Montzka C, Vereecken H, Hendricks Franssen H, Brendt J, Brauner S, Kuckshinrichs W, Venghaus S, Kone D, Korgo B, Ogunjobi K, Olwoch J, Chiteculo V, Getenga Z, Linßen J, Stolten D. Participatory mapping of local green hydrogen cost-potentials in Sub-Saharan Africa. Int J Hydrog Energy 2025;112:289–321. https://doi.org/10.1016/j.ijhydene.2025.02.015.

[7] IEA (2025), Global hydrogen review 2025, IEA, Paris, 2025. https://www.iea.org/reports/global-hydrogen-review-2025.

[8] IEA (2024), Global hydrogen review 2024, IEA, Paris, 2024. https://www.iea.org/reports/global-hydrogen-review-2024.

[9] GLAES. FZJ-IEK3-VSA/glaes: 1.2.1, https://github.com/FZJ-IEK3-VSA/glaes/tree/v1.2.1; 2024.

[10] C. Winkler, P. Dunkel, S. Chen, J. Belina, H. Heinrichs, J. Linßen, D. Stolten, FZJ-IEK3-VSA/RESKit: v0.4.3, (2025). https://doi.org/10.5281/ZENODO.17668775.

[11] Flanders Marine Institute (VLIZ), Belgium, IHO Sea Areas, version 3, (2018). https://doi.org/10.14284/323.

[12] GEBCO Bathymetric Compilation Group 2020, The GEBCO_2020 Grid - a continuous terrain model of the global oceans and land., (2020). https://doi.org/10.5285/A29C5465-B138-234D-E053-6C86ABC040B9.

[13] S. Waldman, P. Munro, R. Forster, Plausible 2050 offshore wind locations in the North Sea, (2023). https://doi.org/10.5281/ZENODO.10259046.

[14] The pandas development team, pandas-dev/pandas: Pandas, (2025). https://doi.org/10.5281/ZENODO.3509134.

[15] Ryberg DS, Caglayan DG, Schmitt S, Linßen J, Stolten D, Robinius M. The future of European onshore wind energy potential: detailed distribution and simulation of advanced turbine designs. Energy 2019;182:1222–38. https://doi.org/10.1016/j.energy.2019.06.052.

[16] Weinand JM, Pelser T, Kleinebrahm M, Stolten D. Countries across the world use more land for golf courses than wind or solar energy. Env Res Commun 2025;7:021012. https://doi.org/10.1088/2515-7620/adb7bd.

[17] Pelser T, Weinand JM, Kuckertz P, Stolten D. ETHOS.REFLOW: an open-source workflow for reproducible renewable energy potential assessments. Patterns 2025;6:101172. https://doi.org/10.1016/j.patter.2025.101172.

[18] Maier R, Behrens J, Hoffmann M, Kullmann F, Weinand JM, Stolten D. Impact of foresight horizons on energy system decarbonization pathways. Adv Appl Energy 2025;18:100217. https://doi.org/10.1016/j.adapen.2025.100217.

[19] Tsani T, Pelser T, Ioannidis R, Maier R, Chen R, Risch S, Kullmann F, McKenna R, Stolten D, Weinand JM. Quantifying the trade-offs between renewable energy visibility and system costs. Nat Commun 2025;16:3853. https://doi.org/10.1038/s41467-025-59029-1.

[20] Caglayan DG, Weber N, Heinrichs HU, Linßen J, Robinius M, Kukla PA, Stolten D. Technical potential of salt caverns for hydrogen storage in Europe. Int J Hydrog Energy 2020;45:6793–805. https://doi.org/10.1016/j.ijhydene.2019.12.161.

[21] Caglayan DG, Heinrichs HU, Robinius M, Stolten D. Robust design of a future 100% renewable European energy supply system with hydrogen infrastructure. Int J Hydrog Energy 2021;46:29376–90. https://doi.org/10.1016/j.ijhydene.2020.12.197.