SpectralIndices.jl: Streamlining spectral indices access and computation for Earth system research

Francesco Martinuzzi[1, 2, 3], Miguel D. Mahecha[1,2,3,5], David Montero[2,3,5], Lazaro Alonso[4], Karin Mora[2,3]

[1] Center for Scalable Data Analytics and Artificial Intelligence (ScaDS.AI), Leipzig, Germany – martinuzzi@informatik.uni-leipzig.de
[2] Institute for Earth System Science & Remote Sensing, Leipzig University, Leipzig, Germany
– (miguel.mahecha, david.montero, karin.mora)@uni-leipzig.de
[3] Remote Sensing Centre for Earth System Research (RSC4Earth), Leipzig University, Leipzig, Germany
[4] Max Planck Institute for Biogeochemistry, Jena, Germany – lalonso@bgc-jena.mpg.de
[5] German Centre for Integrative Biodiversity Research (iDiv), Halle-Jena-Leipzig, Germany

Keywords: Spectral Indices, Remote Sensing, Earth Observation Data, Julia.

Abstract

Remote sensing is an essential technology in environmental science to study Earth surface processes. In optical remote sensing, spectral indices (SI) are widely used to quantify the properties of specific surface characteristics. SI mathematically combine reflectance values measured at different wavelengths. To gain an overview and access to such indices, comprehensive catalogs have been published and implemented in various programming languages. However, there is no Julia-based tool available for efficiently managing and using these indices. Here we introduce `SpectralIndices.jl`, a Julia package designed to retrieve and compute SI. Built on the Awesome Spectral Indices (ASI) catalog, our package enables rapid computation of SI using native functions. The multiple dispatch capability of Julia optimizes data handling across various storage types, ensuring quick load times. While primarily based on the ASI collection, `SpectralIndices.jl` also accommodates custom-made indices, offering users the flexibility to explore and compare alternative indices. The software is open source and available on `github.com/awesome-spectral-indices/SpectralIndices.jl`

1. Introduction

Remote sensing has become essential in many branches of science and applications that deal with geospatial objects. In environmental science, remote sensing helps scientists to monitor, assess for instance, ecosystem status via vegetation health (Kureel et al., 2022), monitor aquatic systems, e.g., algae blooms (Köhler et al., 2024), or manage natural resources (Pinter Jr et al., 2003; Franklin, 2001), to name only a few. With the growing availability of optical Earth observation data, researchers have developed many spectral indices (SI) to analyze specific surface features and phenomena in areas such as vegetation (Zeng et al., 2022), water bodies (Ma et al., 2019), urban environments (Zha et al., 2003), and snow-covered regions (Salomonson and Appel, 2004). Additionally, these indices are used as a basis for environmental machine learning (ML) applications either as targets (Li et al., 2022; Luo et al., 2022; Martinuzzi et al., 2023) or as features (Pabon-Moreno et al., 2022; Montero et al., 2024).

The constantly increasing number of SI has made it essential to develop comprehensive catalogs. The Awesome Spectral Indices (ASI; Montero et al. 2023) suite provides one solution by offering a curated, machine-readable catalog of SI. Additionally, the ASI suite includes a Python library that users can use to query and compute these indices, along with an interface for the Google Earth Engine JavaScript application programming interface (Montero et al., 2022), catering to a diverse user base and a variety of use cases.

There is, however, no SI package developed for Julia, a programming language known for its high-performance computing capabilities (Bezanson et al., 2017). The Julia language has fostered a growing community for geographic and climate applications, as demonstrated by the development of numerous packages such as `Oceananigans.jl` (Ramadhan et al., 2020), `GriddingMachine.jl` (Wang et al., 2022), and `GeoStats.jl` (Hoffimann, 2018) among others. Given that Julia's popularity is rising in Earth and climate science, having no dedicated package for computing SI is a major gap. So far, Julia users can only use a workaround to access Python's extensive libraries and tools from Julia, including the specialized ASI Python package, through `PyCall.jl`(Johnson and PyCall Development Team, 2013), but this comes at a cost of efficiency and flexibility.

Here we introduce `SpectralIndices.jl`, a Julia package that simplifies accessing and computing SI in remote sensing applications. `SpectralIndices.jl` provides a user-friendly, efficient solution to immediately access the collection of SI contained in ASI and includes dedicated functions for their computation. We designed it with flexibility in input data types, ensuring ease of extension and maintenance.

This paper is organized as follows: Section 2 introduces the `SpectralIndices.jl` package. It details how SI are collected in Section 2.1, describes the computational processes in Section 2.2, and outlines the supported data formats for input data in Section 2.3. Section 2.4 provides an overview of the code quality of the library. An applied use case is presented in Section 3. Points of discussion are explored in Section 4, and conclusions are drawn in Section 5.

2. Library Overview

`SpectralIndices.jl` offers direct access to and enables the computation of all SI available within the ASI suite. It is

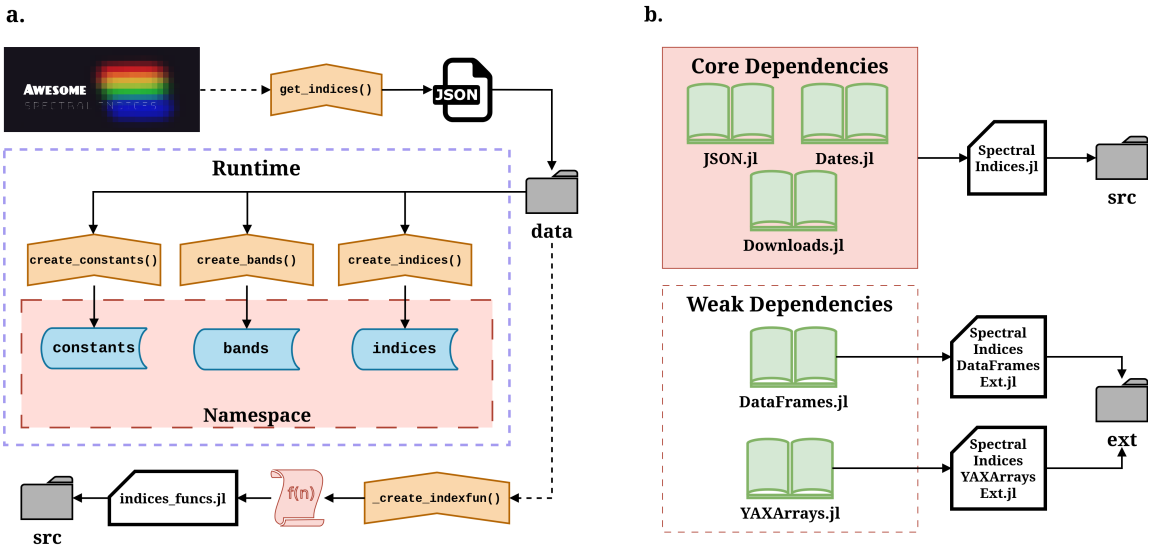

Figure 1. Code and dependencies structure. In panel a, we show the structure of offline and online building blocks of the library. Dashed lines indicate that an action is required from the user, while full lines indicate that functions are executed automatically. Starting from the top left, the function `get_indices()` with the argument set as `true` retrieves the JSON file from the Awesome Spectral Indices (ASI) GitHub page. This file is saved in `data`. The function `_create_indexfun()` builds the SI functions from the formula strings and writes them into a `indices_funcs.jl` in `src`. During runtime, three functions are executed, namely `create_constants()`, `create_bands()`, `create_indices()`. The results are loaded into namespaces and consist of three dictionaries containing the built structs `SpectralIndices`, `Bands`, and `Constants`. These structs contain the information retrieved from the ASI Github page. Not depicted is the additional loading into namespace of all the structs contained in `indices`. Panel b shows the dependencies structure. The core dependencies are imported by default when `SpectralIndices.jl` is imported. In contrast, the weak dependencies and the accompanying functions and methods are only loaded when the user explicitly calls them in the namespace.

built around four key data structures (struct): `SpectralIndex`, `PlatformBand`, `Band`, and `Constant`. The functions `compute_index` and `compute_kernel` form the core of the computational components. Additionally, helper functions such as `get_indices`, `get_dataset()`, and `load_dataset()` provide further functionalities for the library.

2.1 Accessing and Importing Spectral Indices

SI are accessed and imported through the ASI collection. This process is conducted offline using the `get_indices()` function with the argument `true`. If the argument is set to `false` (default setting), the SI list is imported from the one saved in the `data` folder. When new indices are added to ASI, the local list is updated, and the package is subsequently released with a new version number. With the addition of new indices, the function list is updated through `_create_indicesfun()`. This function reads formula strings from the JSON file and converts them into native Julia functions.

When importing the library into a Julia session, the complete list of SI is loaded in two distinct ways. The first method imports a dictionary of `SpectralIndex` objects. A single `indices` variable is added to the namespace, which contains the comprehensive list of indices. In this dictionary, the keys are the acronym of the index, and the values are the `SpectralIndex` structs that contain the SI information sourced from ASI. Similar dictionaries, named `bands` and `constants`, are also imported. These contain information about the bands and constants used in the SI computation. A schematic representation of this process is shown in Figure 1a. In the second method, all `SpectralIndex` structs included in the `indices` list are automatically imported into the namespace, offering a quicker and more convenient method to access information about a specific SI. Calling the index acronym in the Julia read-eval-print loop (REPL) will display its main features, such as the application domain, bands, parameters, formula, and article reference.

2.2 Computing Spectral Indices

`SpectralIndices.jl` provides a primary function, `compute_index()`, to calculate SI. The first positional argument of this function specifies the index (or indices) the user wishes to compute. This argument can be supplied in two ways. The first method involves passing the index's acronym as a string, for example, `compute_index("NDVI")`. However, this method is limited as it does not accommodate indices not present in the `indices` dictionary. The alternative method allows users to specify a `SpectralIndex` struct directly, such as `compute_index(NDVI)`. As discussed in 2.1, all the indices' structs are imported in the namespace as the library is called. Moreover, to compute multiple indices simultaneously, users can pass them as an array to the function, for instance, `compute_index([NDVI, NDWI])`. Unlike the first method, the second one supports the computation of custom indices.

The second component of the `compute_index()` function is the reflectance data (bands) required to compute the SI. The bands can be provided to the function in two ways. The first method involves passing the data as the second positional argument: `compute_index(NDVI, params)`, where `params` is a data structure containing the necessary bands for the NDVI computation. This approach presumes adherence to specific guidelines, such as using the same nomenclature for the bands used in the struct information. For more specific details on

data structures, please refer to the documentation. The alternative approach employs keyword arguments, where using the correct spectral band nomenclature as defined in the chosen `SpectralIndex` struct for computation is necessary. It is important to note that, despite the differences in methodologies for computing SI, all methods are equivalent in terms of computational speed. Many choices are given to accommodate a wide range of potential applications.

Finally, the library offers a `compute_kernel()` function for calculating kernel-based indices, such as the kernel normalized vegetation index (kNDVI) (Camps-Valls et al., 2021). This function operates similarly to `compute_index()`. The first positional argument specifies the kernel required for computation, with options including `linear`, `poly`, and `RBF`. Parameters and data necessary for the computation can be supplied as a second positional argument or via keyword arguments.

2.3 Support for Data Formats

The library supports the computation of indices from bands stored in multiple data formats. The integration of external data types, such as `Tables` or `DataFrames`, is facilitated through external dependencies. Packages declared external are treated as conditional imports, meaning they are only loaded with `SpectralIndices.jl` when explicitly invoked by the user in the script. We use this feature of Julia to maintain a minimal number of core dependencies in the library. Consequently, this approach improves flexibility and ensures a fast load time for `SpectralIndices.jl`. Figure 1b provides a schematic depiction of this dependency structure.

The data types supported by the library, as of v0.2.9, include:

- Native formats: Single values, arrays, matrices, and named tuples are comprehensively supported. These can be used to perform either index or kernel computations in the desired format.
- `DataFrames`: Leveraging `DataFrames.jl` (Bouchet-Valat and Kamiński, 2023), this extension computes indices in a two-dimensional, named format. It additionally supports the use and computation of data with `GeoDataFrames.jl`.
- `YAXArrays`: This format computes three-dimensional named arrays, implemented via `YAXArrays.jl` (Gans et al., 2023).

2.4 Code Quality

The code is open source and hosted on GitHub. It undergoes rigorous testing through the continuous integration tools provided by the platform, covering over 90% of the codebase. The test cases encompass various precision levels (Float64, Float32, Float16) and all the implemented data types for input bands (arrays, matrices, DataFrames, etc.). Furthermore, it involves computation tests for each spectral index, individually and in two random permutations. This thorough testing process not only ensures robustness but also aids in identifying potential errors in the upstream ASI collection. Finally, the tests incorporate Aqua.jl (Arakaki and Aqua Development Team, 2019), an automated quality assurance test suite explicitly designed for Julia packages.

The package documentation is available online and offers straightforward copy-paste examples for quick starts and more complex use cases to accommodate advanced applications. Additionally, it includes a detailed guide to the application programming interface (API) that serves as a comprehensive resource for users. The documentation can be accessed at `https://awesome-spectral-indices.github.io/SpectralIndices.jl/`.

3. Case Study

We demonstrate the use of `SpectralIndices.jl` by predicting the next time step for 16 different SI. The full list of the indices used is available in Table 1. The prediction task is as follows. Let $\mathbf{y}_t$ be the SI at time t. The goal is to predict the SI at the next time step $t+1$, denoted as $\mathbf{y}_{t+1}$. The predictive model, f, is given by

$$\mathbf{y}_{t+1} = f(\mathbf{y}_t; \theta), \tag{1}$$

where θ represents all the model parameters, which are learned from historical data through a training process. The function f might be implemented using various ML techniques such as neural networks, decision trees, or support vector machines. We employ three prediction approaches. In the first approach, we train the model using only the necessary bands. After predicting these bands, we use `SpectralIndices.jl` to compute the SI. In the second approach, we train the model on a single index at a time and sequentially predict all 16, deliberately avoiding parallel computing to simulate conditions where the process is parallelized across different pixels. In the third approach, we use all 16 bands to train the model.

3.1 Prediction Method

Echo state networks (ESNs; Jaeger 2001) are an efficient ML model for prediction of time series (Kim and King, 2020; Akiyama and Tanaka, 2022). They are based on expanding the input data into a higher-dimensional space, called the reservoir. The computational power of an ESN is determined by the size and spectral radius of the matrix that represents this reservoir (Jiang and Lai, 2019). This reservoir is a recurrent neural network with internal weights, which are set randomly and remain fixed. After expanding the input data into higher-dimensional states, linear regression is used to train only the outer layer to make predictions. We chose ESNs for our demonstration due to their straightforward design and effective performance in time series prediction. We implemented the ESNs using `ReservoirComputing.jl` (Martinuzzi et al., 2022). Performance is quantified by the root mean square error (RMSE), defined as $\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$, where n is the number of observations, y_i is the actual value of the i-th observation, and $\hat{y}_i$ is the predicted value of the i-th observation.

All experiments were run on a Dell XPS 9510 fitted with an Intel Core i7-11800H CPU and 16 GB of RAM. No GPU acceleration was used.

3.2 Data

In this study, we used spectral bands from the moderate resolution imaging spectroradiometer (MODIS) (Vermote, 2015) for the time period 2000-2017. We focused on time series data from a single pixel at latitude 51.075 and longitude 10.425, located in the Hainich Forest in central Germany. This forest predominantly consists of deciduous trees. We applied a Savitzky-Golay filter with a 7-day window and a third-order polynomial to smooth and denoise the data (Savitzky and Golay, 1964).

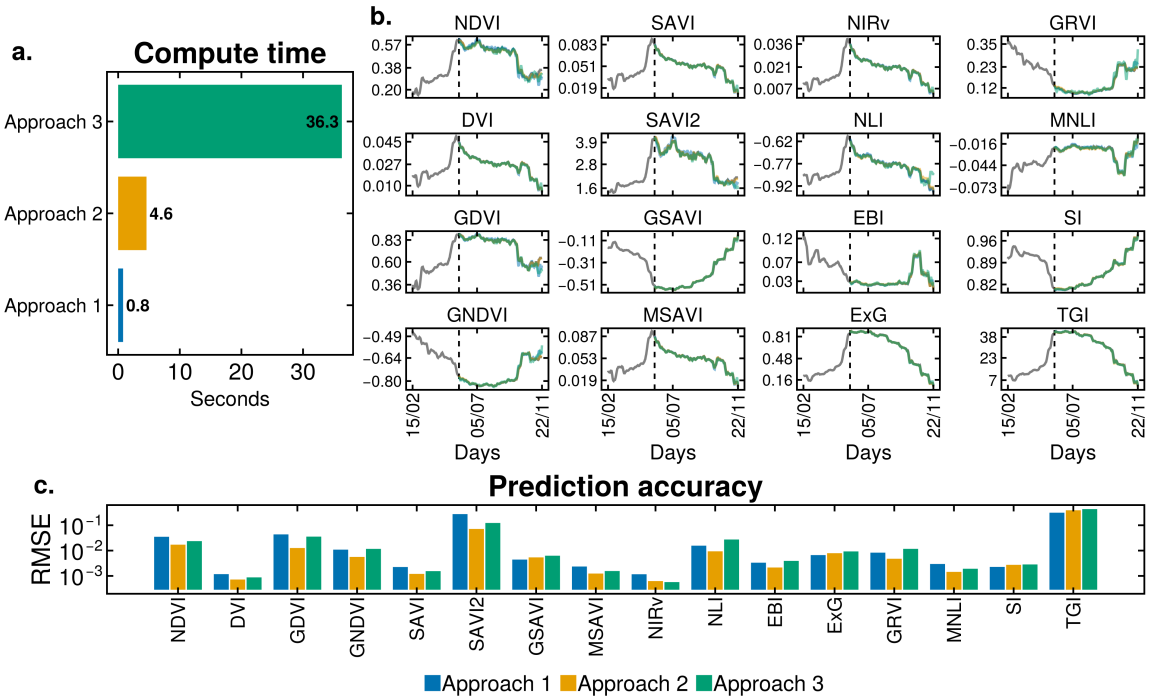

Figure 2. Comparison of three approaches for predicting the next step of 16 spectral vegetation indices (VI). Approach 1 utilizes spectral bands for training and prediction, followed by the calculation of VI using `SpectralIndices.jl`. Approach 2 involves predicting each VI sequentially. In Approach 3, all VIs are used for both training and prediction. Panel a presents the computation times for all three approaches, with mean times derived from an average of 100 runs. Panel b displays the next-step predictions for all 16 VIs from a single run for a pixel located in Hainich, Germany, in the year 2017. The predicted and original signals overlap. Panel c quantifies the performance of the different approaches using the root mean square error (RMSE), with the mean RMSE calculated over 100 runs. The plot is done using `Makie.jl` (Danisch and Krumbiegel, 2021),

3.3 Results

Figure 2 presents the main findings of our case study. Computations using the bands only (approach 1) are performed faster than the alternatives. Specifically, predicting the bands in sequence (approach 2) is about 5.9 times slower than predicting the bands and then computing the indices (approach 1). Predicting all 16 bands (approach 3) is 47 times slower than approach 1. This slower speed stems from the increased number of input variables, which requires a larger reservoir matrix. Consequently, matrix multiplications become more costly, increasing the computation time.

Despite the gains in computational speed, there is no loss in prediction accuracy. For example, the original time series for the year 2017 and the corresponding single run prediction overlap, Figure 2b. Across a hundred runs, the RMSE indicates that no single approach consistently outperforms the others, with all approaches yielding comparable performance, Figure 2c. This study case illustrates how using `SpectralIndices.jl` simplifies SI calculations and can accelerate ML tasks, such as time series prediction.

4. Discussion

This manuscript shows how `SpectralIndices.jl` can assist practitioners in directly accessing SI computation. We illustrate the package's structure and modularity. In this section, we will discuss how the package and its features can contribute to a remote sensing scientist's computational toolkit.

The library is integrated within the growing Julia community and is dedicated to Earth system and climate software, specifically designed to support remote sensing developers. Our case study demonstrates how this package can reduce computational costs in ML applications by minimizing the number of variables required for training and prediction, Figure 2. More extensive ML investigations could capitalize on the wealth of information in the ASI collection by exploring various models and properties of SI (Reinhardt et al., 2024).

The software offers multiple data input options. However, this selection is not exhaustive, and expanding these options would increase the adaptability of the package. Plans for future expansions are already in progress, including extensions like `Rasters.jl` (Schouten and Rasters Development Team, 2019) and `Tables.jl` (Quinn et al., 2021), which will offer alternatives for the existing extensions. Future data types that are being considered for inclusion in `SpectralIndices.jl` are `AxisArrays.jl` (Bauman and AxisArrays Development Team, 2014), `NamedDims.jl` (White and NamedDims Development Team, 2019) and `NamedArrays.jl` (van Leeuwen and NamedArrays Development Team, 2013). A full list is available in the `SpectralIndices.jl` Github repository in issue number 8. Additionally, computationally intensive applications benefit from parallel computation, a feature not yet fully integrated into `SpectralIndices.jl`. While it is possible to achieve parallel computation of SI using `Distributed.jl` (Bezanson and Distributed Development Team, 2011) and adapting internal functions with `pmap`, integrating this capability natively within the package would improve its functionality.

5. Conclusion

In this work, we introduced `SpectralIndices.jl`, a user-friendly package to efficiently compute SI from the ASI collection. Designed with modularity in mind, our software aims to simplify the process for researchers. Our case study demonstrates how this library can help streamline current research projects by facilitating easy access and computation of SI.

Acknowledgements

F.M. acknowledges support from the Federal Ministry of Education and Research of Germany and by Sächsische Staatsministerium für Wissenschaft, Kultur und Tourismus in the program Center of Excellence for AI-research "Center for Scalable Data Analytics and Artificial Intelligence Dresden/Leipzig", project identification number: ScaDS.AI. K.M. acknowledges funding by the Saxon State Ministry for Science, Culture and Tourism (SMWK) – [3-7304/35/6-2021/48880]. M.D.M thanks the German Federal Ministry for Economic Affairs and Climate Action in the framework of the "national center of excellence ML4Earth" (grant number: 50EE2201C). KM and M.D.M. acknowledge funding from the European Space Agency for the projects DeepExtremes and DeepFeatures.

References

Akiyama, T., Tanaka, G., 2022. Computational efficiency of multi-step learning echo state networks for nonlinear time series prediction. *IEEE Access*, 10, 28535–28544. http://dx.doi.org/10.1109/ACCESS.2022.3158755.

Arakaki, T., Aqua Development Team, 2019. Aqua.jl. `https://github.com/JuliaTesting/Aqua.jl`.

Badgley, G., Field, C. B., Berry, J. A., 2017. Canopy near-infrared reflectance and terrestrial photosynthesis. *Science advances*, 3(3), e1602244. http://dx.doi.org/10.1126/sciadv.1602244.

Bauman, M., AxisArrays Development Team, 2014. AxisArrays.jl. `https://github.com/JuliaArrays/AxisArrays.jl`.

Bezanson, J., Distributed Development Team, 2011. Distributed.jl. `https://github.com/JuliaLang/Distributed.jl`.

Bezanson, J., Edelman, A., Karpinski, S., Shah, V. B., 2017. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1), 65–98. https://doi.org/10.1137/141000671.

Bouchet-Valat, M., Kamiński, B., 2023. DataFrames.jl: Flexible and Fast Tabular Data in Julia. *Journal of Statistical Software*, 107(4), 1–32. https://www.doi.org/10.18637/jss.v107.i04.

Camps-Valls, G., Campos-Taberner, M., Moreno-Martínez, Á., Walther, S., Duveiller, G., Cescatti, A., Mahecha, M. D., Muñoz-Marí, J., García-Haro, F. J., Guanter, L. et al., 2021. A unified vegetation index for quantifying the terrestrial biosphere. *Science Advances*, 7(9). http://dx.doi.org/10.1126/sciadv.abc7447.

Chen, B., Jin, Y., Brown, P., 2019. An enhanced bloom index for quantifying floral phenology using multi-scale remote sensing observations. *ISPRS Journal of Photogrammetry and Remote Sensing*, 156, 108–120. http://dx.doi.org/10.1016/j.isprsjprs.2019.08.006.

Danisch, S., Krumbiegel, J., 2021. Makie. jl: Flexible high-performance data visualization for Julia. *Journal of Open Source Software*, 6(65), 3349. http://dx.doi.org/10.21105/joss.03349.

Franklin, S. E., 2001. *Remote sensing for sustainable forest management*. CRC press.

Gans, F., Cremer, F., Alonso, L., Kraemer, G., Dimens, P. V., Gutwin, M., Pabon-Moreno, D. E., Kong, D., Martin, Martinuzzi, F., Chettouh, M. A., Loos, D., Zehner, M., Roy, P., Zhang, Q., ckrich, Glaser, F., linamaes, 2023. JuliaDataCubes/YAXArrays.jl: v0.5.2. https://doi.org/10.5281/zenodo.8414000.

Gitelson, A. A., Kaufman, Y. J., Merzlyak, M. N., 1996. Use of a green channel in remote sensing of global vegetation from EOS-MODIS. *Remote sensing of Environment*, 58(3), 289–298. http://dx.doi.org/10.1016/S0034-4257(96)00072-7.

Goel, N. S., Qin, W., 1994. Influences of canopy architecture on relationships between various vegetation indices and LAI and FPAR: A computer simulation. *Remote Sensing Reviews*, 10(4), 309–347. http://dx.doi.org/10.1080/02757259409532252.

Gong, P., Pu, R., Biging, G. S., Larrieu, M. R., 2003. Estimation of forest leaf area index using vegetation indices derived from Hyperion hyperspectral data. *IEEE transactions on geoscience and remote sensing*, 41(6), 1355–1362. http://dx.doi.org/10.1109/TGRS.2003.812910.

Hoffimann, J., 2018. GeoStats.jl – High-performance geostatistics in Julia. *Journal of Open Source Software*, 3(24), 692. http://dx.doi.org/10.21105/joss.00692.

Huete, A. R., 1988. A soil-adjusted vegetation index (SAVI). *Remote sensing of environment*, 25(3), 295–309. http://dx.doi.org/10.1016/0034-4257(88)90106-X.

Hunt, E. R., Doraiswamy, P. C., McMurtrey, J. E., Daughtry, C. S., Perry, E. M., Akhmedov, B., 2013. A visible band index for remote sensing leaf chlorophyll content at the canopy scale. *International Journal of Applied Earth Observation and Geoinformation*, 21, 103–112. http://dx.doi.org/10.1016/j.jag.2012.07.020.

Jaeger, H., 2001. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148(34), 13. https://www.ai.rug.nl/minds/uploads/EchoStatesTechRep.pdf.

Jiang, J., Lai, Y.-C., 2019. Model-free prediction of spatiotemporal dynamical systems with recurrent neural networks: Role of network spectral radius. *Physical review research*, 1(3), 033056. http://dx.doi.org/10.1103/PhysRevResearch.1.033056.

Johnson, S. G., PyCall Development Team, 2013. PyCall.jl. `https://github.com/JuliaPy/PyCall.jl`.

Kim, T., King, B. R., 2020. Time series prediction using deep echo state networks. *Neural Computing and Applications*, 32(23), 17769–17787. http://dx.doi.org/10.1007/s00521-020-04948-x.

Köhler, J., Varga, E., Spahr, S., Gessner, J., Stelzer, K., Brandt, G., Mahecha, M. D., Kraemer, G., Pusch, M., Wolter, C. et al., 2024. Unpredicted ecosystem response to compound human impacts in a European river. https://doi.org/10.21203/rs.3.rs-3792221/v1.

Kureel, N., Sarup, J., Matin, S., Goswami, S., Kureel, K., 2022. Modelling vegetation health and stress using hypersepctral remote sensing data. *Modeling Earth Systems and Environment*, 1–16. https://doi.org/10.1007/s40808-021-01113-8.

Li, J., Meng, Y., Li, Y., Cui, Q., Yang, X., Tao, C., Wang, Z., Li, L., Zhang, W., 2022. Accurate water extraction using remote sensing imagery based on normalized difference water index and unsupervised deep learning. *Journal of Hydrology*, 612, 128202. https://doi.org/10.1016/j.jhydrol.2022.128202.

Luo, J., Dong, C., Lin, K., Chen, X., Zhao, L., Menzel, L., 2022. Mapping snow cover in forests using optical remote sensing, machine learning and time-lapse photography. *Remote Sensing of Environment*, 275, 113017. https://doi.org/10.1016/j.rse.2022.113017.

Ma, S., Zhou, Y., Gowda, P. H., Dong, J., Zhang, G., Kakani, V. G., Wagle, P., Chen, L., Flynn, K. C., Jiang, W., 2019. Application of the water-related spectral reflectance indices: A review. *Ecological indicators*, 98, 68–79. https://doi.org/10.1016/j.ecolind.2018.10.049.

Major, D., Baret, F., Guyot, G., 1990. A ratio vegetation index adjusted for soil brightness. *International journal of remote sensing*, 11(5), 727–740. http://dx.doi.org/10.1080/01431169008955053.

Martinuzzi, F., Mahecha, M. D., Camps-Valls, G., Montero, D., Williams, T., Mora, K., 2023. Learning Extreme Vegetation Response to Climate Forcing: A Comparison of Recurrent Neural Network Architectures. *EGUsphere*, 2023, 1–32. https://doi.org/10.5194/egusphere-2023-2368.

Martinuzzi, F., Rackauckas, C., Abdelrehim, A., Mahecha, M. D., Mora, K., 2022. ReservoirComputing. jl: An efficient and modular library for reservoir computing models. *Journal of Machine Learning Research*, 23(288), 1–8. https://www.jmlr.org/papers/v23/22-0611.html.

Montero, D., Aybar, C., Mahecha, M. D., Martinuzzi, F., Söchting, M., Wieneke, S., 2023. A standardized catalogue of spectral indices to advance the use of remote sensing in Earth system research. *Scientific Data*, 10(1), 1–20. https://doi.org/10.1038/s41597-023-02096-0.

Montero, D., Aybar, C., Mahecha, M. D., Wieneke, S., 2022. spectral: awesome spectral indices deployed via the Google Earth engine JavaScript API. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 48, 301–306. https://doi.org/10.5194/isprs-archives-XLVIII-4-W1-2022-301-2022.

Montero, D., Mahecha, M. D., Martinuzzi, F., Aybar, C., Klosterhalfen, A., Knohl, A., Koebsch, F., Anaya, J., Wieneke, S., 2024. Recurrent Neural Networks for Modelling Gross Primary Production. *arXiv preprint arXiv:2404.12745*. https://arxiv.org/abs/2404.12745.

Pabon-Moreno, D. E., Migliavacca, M., Reichstein, M., Mahecha, M. D., 2022. On the potential of Sentinel-2 for estimating Gross Primary Production. *IEEE Transactions on Geoscience and Remote Sensing*, 60, 1–12. https://doi.org/10.1109/TGRS.2022.3152272.

Pinter Jr, P. J., Hatfield, J. L., Schepers, J. S., Barnes, E. M., Moran, M. S., Daughtry, C. S., Upchurch, D. R., 2003. Remote sensing for crop management. *Photogrammetric Engineering & Remote Sensing*, 69(6), 647–664. https://doi.org/10.14358/PERS.69.6.647.

Qi, J., Chehbouni, A., Huete, A. R., Kerr, Y. H., Sorooshian, S., 1994. A modified soil adjusted vegetation index. *Remote sensing of environment*, 48(2), 119–126. http://dx.doi.org/10.1016/0034-4257(94)90134-1.

Quinn, J., Anthoff, D., Arakaki, T., Kamiński, B., Bouchet-Valat, M., Papp, T. K., Arslan, A., ExpandingMan, Schouten, R., Robinson, N., Ferris, A., Arkoniak, Sharma, A., Davies, E., Kalybek, E., Lin, I., Dunn, J., Revels, J., Day, J., Calderón, J. B. S., TagBot, J., Samuel, O., Vertechi, P., Zwitch, R., StatisticalMice, strickek, 2021. Juliadata/tables.jl: v1.3.1.

Ramadhan, A., Wagner, G. L., Hill, C., Campin, J.-M., Churavy, V., Besard, T., Souza, A., Edelman, A., Ferrari, R., Marshall, J., 2020. Oceananigans.jl: Fast and friendly geophysical fluid dynamics on GPUs. *Journal of Open Source Software*, 5(53), 2018. https://doi.org/10.21105/joss.02018.

Reinhardt, M., Mora, K., Brandt, G., Harish, T. M., Montero, D., Ji, C., Kattenborn, T., Martinuzzi, F., Mosig, C., Mahecha, M. D., 2024. DeepFeatures: Remote sensing beyond spectral indices. http://dx.doi.org/10.5194/egusphere-egu24-18852.

Rikimaru, A., Roy, P., Miyatake, S. et al., 2002. Tropical forest cover density mapping. *Tropical ecology*, 43(1), 39–47.

Roujean, J.-L., Breon, F.-M., 1995. Estimating PAR absorbed by vegetation from bidirectional reflectance measurements. *Remote sensing of Environment*, 51(3), 375–384. http://dx.doi.org/10.1016/0034-4257(94)00114-3.

Rouse, J. W., Haas, R. H., Schell, J. A., Deering, D. W. et al., 1974. Monitoring vegetation systems in the Great Plains with ERTS. *NASA Spec. Publ*, 351(1), 309. https://ntrs.nasa.gov/citations/19740022614.

Salomonson, V. V., Appel, I., 2004. Estimating fractional snow cover from MODIS using the normalized difference snow index. *Remote Sensing of Environment*, 89(3), 351–360. https://doi.org/10.1016/j.rse.2003.10.016.

Savitzky, A., Golay, M. J., 1964. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8), 1627–1639. http://dx.doi.org/10.1021/ac60214a047.

Schouten, R., Rasters Development Team, 2019. Rasters.jl. `https://github.com/rafaqz/Rasters.jl`.

Sripada, R. P., Heiniger, R. W., White, J. G., Weisz, R., 2005. Aerial color infrared photography for determining late-season nitrogen requirements in corn. *Agronomy journal*, 97(5), 1443–1451. http://dx.doi.org/10.2134/agronj2004.0314.

van Leeuwen, D., NamedArrays Development Team, 2013. NamedArrays.jl. `https://github.com/davidavdav/NamedArrays.jl`.

Vermote, E., 2015. MOD09A1 MODIS/Terra Surface Reflectance 8-Day L3 Global 500m SIN Grid V006. https://lpdaac.usgs.gov/products/mod09a1v006/.

Wang, Y., Köhler, P., Braghiere, R. K., Longo, M., Doughty, R., Bloom, A. A., Frankenberg, C., 2022. GriddingMachine, a database and software for Earth system modeling at global and regional scales. *Scientific Data*, 9(1), 258. https://doi.org/10.1038/s41597-022-01346-x.

White, F., NamedDims Development Team, 2019. NamedDims.jl. `https://github.com/invenia/NamedDims.jl`.

Woebbecke, D. M., Meyer, G. E., Von Bargen, K., Mortensen, D. A., 1995. Color indices for weed identification under various soil, residue, and lighting conditions. *Transactions of the ASAE*, 38(1), 259–269. http://dx.doi.org/10.13031/2013.27838.

Wu, W., 2014. The generalized difference vegetation index (GDVI) for dryland characterization. *Remote Sensing*, 6(2), 1211–1233. http://dx.doi.org/10.3390/rs6021211.

Zeng, Y., Hao, D., Huete, A., Dechant, B., Berry, J., Chen, J. M., Joiner, J., Frankenberg, C., Bond-Lamberty, B., Ryu, Y. et al., 2022. Optical vegetation indices for monitoring terrestrial ecosystems globally. *Nature Reviews Earth & Environment*, 3(7), 477–493. https://doi.org/10.1038/s43017-022-00298-5.

Zha, Y., Gao, J., Ni, S., 2003. Use of normalized difference built-up index in automatically mapping urban areas from TM imagery. *International Journal of Remote Sensing*, 24(3), 583–594. https://doi.org/10.1080/01431160304987.

Appendix

Index	Bands	Constants	Formula	Reference
NDVI	N,R	/	$\frac{N-R}{N+R}$	(Rouse et al., 1974)
DVI	N,R	/	$N-R$	(Roujean and Breon, 1995)
GDVI	N,R	nexp	$\frac{N^{nexp}-R^{nexp}}{N^{nexp}+R^{nexp}}$	(Wu, 2014)
GNDVI	N,G	/	$\frac{N-G}{N+G}$	(Gitelson et al., 1996)
SAVI	N,R,L	/	$(1.0+L)*\frac{N-R}{N+R+L}$	(Huete, 1988)
SAVI2	N,R	slb, sla	$\frac{N}{R+slb/sla}$	(Major et al., 1990)
GSAVI	L,N,G	/	$(1.0+L)*\frac{N-G}{N+G+L}$	(Sripada et al., 2005)
MSAVI	N,R	/	$1/2(2N+1-\sqrt{(2N+1)^2-8(N-R)})$	(Qi et al., 1994)
NIRv	N,R	/	$\frac{N-R}{N+R}N$	(Badgley et al., 2017)
NLI	N,R	/	$\frac{N^2-R}{N^2+R}$	(Goel and Qin, 1994)
EBI	R,G,B	epsilon	$\frac{R+G+B}{(\frac{G}{B})\cdot(R-B+\epsilon)}$	(Chen et al., 2019)
ExG	R,G,B	/	$2G-R-B$	(Woebbecke et al., 1995)
GRVI	N,G	/	N/G	(Sripada et al., 2005)
MNLI	L,N,R	/	$(1+L)\cdot\frac{N^2-R}{N^2+R+L}$	(Gong et al., 2003)
SI	R,G,B	/	$((1.0-B)(1.0-G)(1.0-R))^{\frac{1}{3}}$	(Rikimaru et al., 2002)
TGI	R,G,B	/	$-0.5\cdot(190\cdot(R-G)-120\cdot(R-B))$	(Hunt et al., 2013)

Table 1. Vegetation indices used in the case study. The indices follow the naming standard from Awesome Spectral Indices: B for the blue band, G for the green band, R for the red band, and N for the near-infrared band. Additionally, the constants are index specific. For the experiments in the case study nexp is set to 2, slb is set to 0, sla is set to 1, and epsilon is set to 1