Upscaling Plant Biodiversity to Predict Animal Taxon Biodiversity and the Impact of Land Use on Mt. Kilimanjaro

Konstantin Engelmayer
Universitaet Marburg

September 29, 2024

Abstract

This study investigates the impact of human land use on biodiversity on Mt. Kilimanjaro by upscaling plant and animal biodiversity data. Utilizing the KiLi Research Unit's extensive datasets, remote sensing and machine learning techniques were applied to create a comprehensive biodiversity map. The objectives were to evaluate the use of plant biodiversity as a proxy for animal biodiversity, identify biodiversity hotspots, and analyze the influence of various land use systems. The VSURF (Variable Selection Using Random Forests) method was employed for predictor selection, and Partial Least Squares Regression models were used for predictions, validated with cross-validation and Area of Applicability assessments. Results indicated elevation as key predictor. Contrary to expectations, plant biodiversity did not reliably predict animal biodiversity, highlighting the importance of other factors. Although the study's results are limited in reliability due to the flaws in the study design, which involved a limited number of replicates for each land use type, biodiversity maps were used as a general indicator for biodiversity. These maps revealed higher species richness at lower elevations, especially in home gardens, savannas, and lower montane forests, while higher elevations showed lower species richness, aligning with other studies. Given the methodological constraints, further research is necessary to refine these findings and provide a more comprehensive understanding of the relationships between land use and biodiversity.

Contents

1 Introduction

Understanding the impact of human land use on biodiversity is crucial for effective nature conservation management and sustainable development strategies [1]. Biodiversity mapping is essential to identify biodiversity hotspots and to discern the links between land use and biodiversity [2]. However, the challenge lies in the time-consuming and resource-intensive process of sampling biodiversity data of different animal taxa and plants over extensive areas. To overcome this challenge, upscaling methods are necessary to provide a comprehensive picture of biodiversity patterns and their relationship with human activities on a landscape scale [3].

Upscaling involves taking detailed, local data and extrapolating it to broader spatial scales. The challenge in this process is the necessity to use data that is available on larger scales, which often limits the data to remote sensing sources. This especially makes the prediction of animal diversity difficult because animals are generally not visible in remote sensing data. However, plants are visible, making their biodiversity easier to predict. Additionally, it is well known that plant and animal biodiversity are deeply connected, with animals playing a crucial role in seed dispersal and plants serving as essential food sources for various animal species [4, 5, 6, 7]. Thus, this project intends to use this link and aims to upscale plant biodiversity data to investigate whether it can be effectively used to predict the biodiversity of different animal taxa. The goal is to produce a coherent dataset of plant and animal biodiversity on a landscape scale, identify biodiversity hotspots, and understand the links between land use and biodiversity.

This study is situated within the framework of the KiLi Research Unit, established on Mt. Kilimanjaro. The KiLi Project, a collaborative effort between Tanzanian and German scientists funded by the German Science Foundation (DFG), has extensively studied the influence of climate and land-use change on biodiversity and multiple ecosystem processes on Mt. Kilimanjaro. Researchers from various disciplines, including climate and soil sciences, botany, and zoology, have conducted studies on 60 research plots across 12 different land use types from savanna to alpine habitats [8].

The KiLi Project has primarily focused on how biodiversity influences ecosystem services and emphasizes the necessity of maintaining biodiversity to sustain nature's contributions to people. However, this approach may lead to an oversight of species and ecosystems that don't seem to contribute to mankind, as it only values biodiversity that directly benefits humans [9]. This narrow focus risks ignoring the potential benefits of biodiversity that we don't yet understand. By overlooking these unknown contributions, we may fail to protect essential components of biodiversity crucial for maintaining the balance and health of ecosystems [10]. Additionally, this concept overlooks the intrinsic value of biodiversity, which exists independently of its contribution to people [11]. This is why this study aims to shift the focus away from the anthropocentric concept of ecosystem services and instead emphasize the intrinsic value of biodiversity. By concentrating on the effect of land use on biodiversity, this project aims to reveal how human activities impact biodiversity, independent of the benefits provided to humans. This approach aligns with a more holistic view of nature, recognizing its inherent value beyond human utility.

The primary goals of this study are:

1. **To determine if upscaled plant biodiversity can be used as a proxy to predict the biodiversity of different animal taxa.**

2. **To identify biodiversity hotspots by upscaling plant and animal data to create a comprehensive biodiversity map.**

3. **To analyze the impact of various land use systems on these biodiversity hotspots.**

By achieving these goals, this study aims to provide a deeper understanding of the relationships between plant and animal biodiversity and the effects of human land use on these relationships. The findings will contribute to more informed conservation strategies and highlight the need to protect biodiversity for its intrinsic value.

2 Methods

2.1 Research Area

Mt. Kilimanjaro provides an ideal setting for global change research due to its unique altitudinal range of over 5000 meters, encompassing diverse climate and vegetation zones from tropical savannas to afroalpine grasslands. The mountain also features various land-use systems, ranging from intensive monocultures to traditional agroforestry systems that maintain a semi-natural forest structure. This diverse setting allows for a comprehensive study of how different land use systems impact biodiversity. The study area is situated on the southern slope of Mt. Kilimanjaro. In this area, 60 plots, each measuring 50m x 50m, were established along an elevation gradient from 800m to 4400m. These plots span 12 distinct land cover zones, with 5 replicates per zone [12].

2.2 Response variables

The response variables in this study represent the species richness (SR) of plants and various animal taxa. All SR data available

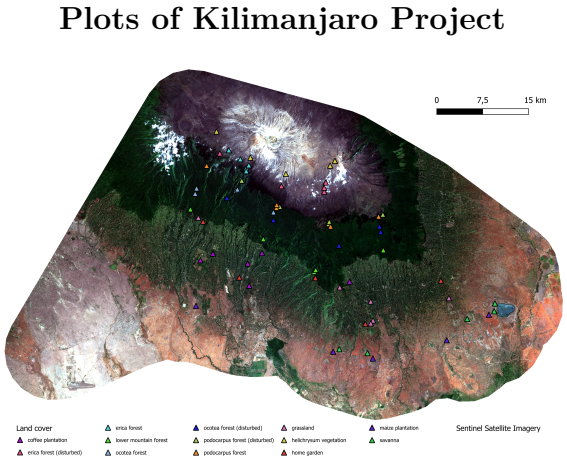


Figure 1: Map of Kilimanjaro area and plots

for the Kilimanjaro region was used to create the most accurate biodiversity map. Table 1 shows an overview of the different animal taxa, their sampling methods, and the specifics involved in calculating SR.

2.3 Predictor selection

To develop an accurate biodiversity map for the Kilimanjaro region, machine learning algorithms were leveraged to predict the SR of both plant species and various animal taxa. Selecting the right predictors was crucial to avoid overfitting and to maintain a model that was both simple and effective. Initially, a comprehensive set of predictors was employed for both plant and animal SR predictions (Appendix table 5).

To compile these predictor variables, all available remote sensing layers were collected in a brute-force manner. Based on the spectral variance hypothesis (SVH), which states that higher variance in spectral data can be an indicator of higher biodiversity [14], the standard deviation (SD) was calculated for each layer.This method was chosen due to the highly diverse needs of different animal taxa in nature. The initial dataset included WorldClim data, modeled pH data, soil data, land-cover data, Sentinel data with various calculated indices, topographic data, and human influence data,

Table 1: Sampling methods and SR calculation for animal various taxa (Table taken from Ziegler et al. [13]) and slightly adjusted.

Taxon	**General Sampling Method**	**Sampling Specifics**	**SR Calculation**
ants	plastic tubes with diverse set of resource baits on the ground	2 h at times of peak ant activity	
bats	acoustic monitoring (point stop method)	every corner of plot visited for 5 min	averaging SR values of single surveys
bees	pan traps in different vegetation heights	48 h with 3 sampling rounds	
birds	audiovisual point counts	15 min before sunrise, completed before 9 am	
dung beetles	baited pitfall trap	72 h sampling time	
grasshoppers	sightings and two rounds of sweep net sampling	non-forested areas: repeatedly 1.5 h walking on parallel tracks, forested areas: 1.5 h shaking understory vegetation	sampling effort per site was adjusted to measure asymptotic SR
hoverflies	pan traps in different vegetation heights	48 h with 3 sampling rounds	total (cumulative) number of SR with varying number of samples among study sites
large mammals	camera trap, analysis of dung remains	5 camera traps per site, 70 trap-days per site	number of all non-domestic mammal species recorded on study site
millipedes	pitfall traps and sightings	sightings: 2 h	
other beetles	pitfall traps	7 days sampling time with a total of 5 sampling rounds	
other wasps	pan traps in different vegetation heights	48 h with 3 sampling rounds	
parasitoid wasps	pan traps in different vegetation heights	48 h with 3 sampling rounds	
snails	sightings (large taxa) and collection of leaf litter (small taxa)	sightings: four rounds of fixed time surveys of 30 min, collection: 1 litre leaf litter	
spiders	pitfall traps	7 days sampling time	
springtails	pitfall traps	7 days sampling time	
true bugs	sweep net sampling	100 sweeps along two 50 m transects	

ending up with 150 different predictor variables.

For animal biodiversity, this set was further enhanced by integrating the predicted and upscaled plant biodiversity data, allowing for an assessment of the influence of plant biodiversity as a predictor for animal biodiversity.

For each individual taxon, the extensive predictor set was refined to include only the most relevant variables using the VSURF (Variable Selection Using Random Forests) method [15]. This method effectively identified key variables through a three-step process:

1. **Variable Screening:** This initial phase involved selecting important variables based on importance scores from a random forest (RF) model, discarding those with low importance scores.

2. **Interpretation Step:** The remaining variables were further analyzed to eliminate redundancy, ensuring that only the most informative variables were retained.

3. **Prediction Step:** The final step focused on selecting the smallest subset of variables that provided the best predictive performance, optimizing both the model's accuracy and efficiency.

The well established VSURF method, identified as one of the most effective methods by Speiser et al. [16], excelled in handling large and complex datasets, making it a valuable tool for ecological research and biodiversity prediction.

2.4 Training, Validation and Testing

Given the diversity of land use types and the limited number of repetitions per type, a reliable training and testing strategy is essential. In this project parts of the robust tuning strategy of Ziegler et al. was adopted [13], which employs two separate 20-fold stratified cross-validation (CV) cycles (Figure 2).

The outer CV cycle randomly selects one plot per land cover type for testing, while the remaining plots are used for training in each fold. This ensures that each land cover type is adequately represented in both training and testing sets. Using several different folds helped with choosing the most stable set of training and testing data for the final model.

Zieglers et al. [13] inner CV is embedded within the Partial Least Squares Regression (PLSR) machine learning approach, following the same method of leaving out one plot per land cover type in each resample. This inner CV, which includes forward feature selection (FFS), is dedicated to model tuning and variable selection. PLSR is known for its ability to handle multicollinearity and its effectiveness in reducing dimensionality while retaining the essential information for prediction. Most importantly for this project, the PLSR is very robust and particularly well-suited for small sample sizes, which is crucial given the limited repetitions per land use type in the project's dataset. The robustness of PLSR ensures that the model can still perform reliably and provide meaningful insights despite the constrained data availability [17].

This inner CV, however, could not be used in this project for several reasons. Firstly, the FFS exceeded available computation resources, leading to the choice of the VSRUF method for this project. Even though the VSURF method was designed for variable selection for the RF algorithm, it led to better results than a customized approach adapted to the PLSR algorithm. Secondly, the inner CV of Ziegler et al. led to plots being chosen for both training and testing in different folds, preventing the calculation of the Area of Applicability (AOA). The AOA, introduced by Meyer and Pebesma [18], defines the range within a model's predictions are reliable based on

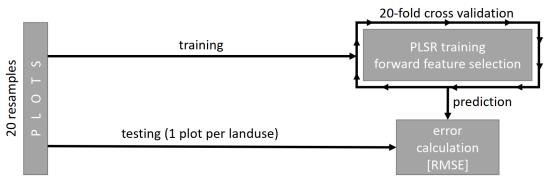


Figure 2: The model training (upper right loop) employs partial least squares regression (PLSR) combined with FFS , utilizing 20-fold CV. Validation involves predicting values for the testing plots. The division of testing and training plots (outer loop) uses a repeated stratified sampling method, randomly selecting one plot per land cover for testing and using the remaining plots for training [13].

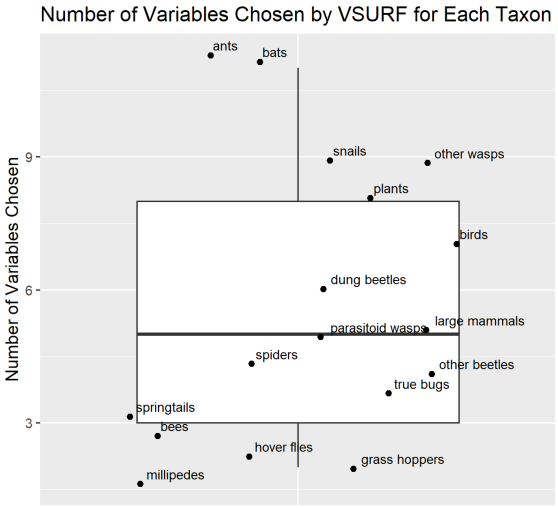


Figure 3: Boxplot of selected predictors for each taxon.

the training data and is essential in this project. To make the AOA calculation possible simple 10-fold CV's were performed in the training process.

Out of the 20 different models resulting from the outer CV, the best model was selected based on the lowest RMSE values for both training and testing. To make the RMSE values comparable between all taxa, they were normalized to the mean of SR of the taxon. After the model training, each model was used to upscale the biodiversity of this taxon. Like this, maps of biodiversity of all sampled taxon for the whole Kilimanjaro region were created, identifying hotspots and the effect of land use on biodiversity.

3 Results

3.1 Selected Variables

A mean of 5.588235 different predictors were chosen for each taxon. Ants and bats had the most predictors with 11, while millipedes had the least with 2 (figure 3. For moths, the VSURF method wasn't able to select variables, so moths were excluded from the project to keep the results comparable.

Variables from all different available sources were selected as predictors, with climate variables being the most frequently chosen, appearing 43 times. Among these, 27 were temperature-related and 19 were precipitation-related, as shown in Table 2. The most frequently selected climate variables were BIO17 (Precipitation of Driest Quarter) and BIO4 (Temperature Seasonality), each occurring 5 times.

Indices derived from Sentinel imagery were chosen 20 times, with the Bare Soil Index (BSI) appearing most frequently at 4 occurrences. The vegetation-related indices, ARVI (Atmospherically Resistant Vegetation Index), GNDVI (Green Normalized Difference Vegetation Index), and WDRVI (Wide Dynamic Range Vegetation Index), were each selected twice.

Topographic variables were selected 11 times. The vertical distance to the nearest stream was the most frequently selected topographic predictor, appearing 6 times, while the Euclidean distance to the nearest stream was selected 2 times. Sentinel bands were chosen 9 times, and soil-related layers were selected 5 times.

The predicted plant SR was never chosen as predictor for the different taxa. (See table 6 in Appendix for all selected predictors for each taxon.)

Table 2: Counts for selected variables, which were selected more than one time

Variable	Count	Source
vertical_distance_to_stream	6	topographic
wc2.1_30s_bio_17	5	climate
wc2.1_30s_bio_4	5	climate
BSI	4	indicie
wc2.1_30s_bio_3	4	climate
wc2.1_30s_bio_8	4	climate
wc2.1_30s_bio_7	3	climate
AOT	2	sentinel
ARVI	2	indicie
distance_to_stream	2	topographic
GNDVI	2	indicie
hii_landuse_driver	2	climate
soil_grids	2	soil
wc2.1_30s_bio_10	2	climate
wc2.1_30s_bio_16	2	climate
wc2.1_30s_bio_19	2	climate
wc2.1_30s_bio_2	2	climate
wc2.1_30s_bio_9	2	climate
WDRVI	2	indicie

3.2 Prediction Validation

For each taxon the model was chosen based on low RSME values for training and testing, and a small difference between both (Appendix 5). The performance of the predictive models for various taxa was evaluated using the normalized RMSE and correlation coefficient (R) for both training and test datasets. SD and mean values were also calculated to provide a comprehensive evaluation. The results are summarized in table 3. The normalized RMSE values were obtained by dividing the RMSE by the mean, allowing comparison across taxa with very different individual counts .

The model for bats showed the best performance, with normalized RMSE values of 0.23 for training and 0.25 for testing. This means that the predicted values differed from the actual SR values by about 23% of the mean SR values for the training data and by about 25% for the testing data. In other words, the model's predictions were, on average, within 23% to 25% of the actual mean SR values for bats. They also showed very high R values of 0.87 for training and 0.89 for testing.

Grasshoppers had the highest test normalized RMSE at 1.32 and millipedes the highest train normalzed RSME at 1.30. There also were some really low R values for other beetles and spiders being smaller than 0.1.

The model for plants had one of the lowest normalized RMSE values, with 0.28 for training and 0.30 for testing, and moderate correlation coefficients of 0.52 for training and 0.55 for testing.

Over all taxa the RSME of training and testing only differed slightly mostly staying under 0.05 difference. Grasshoppers had the biggest difference of 0.2795836.

Table 3: RSME and R-sqared for training and testing runs, SD and mean of SR of each Response variable, relative RSME to the mean of SR.

Taxon	**RMSE Train**	**RMSE Test**	**R Train**	**R Test**	**SD**	**Mean**	**Relative RMSE Train**	**Relative RMSE Test**
ants	2.063051	1.903497	0.8319484	0.81110434	3.356249	2.700000	0.7640928	0.7049989
bats	1.224542	1.322105	0.8706912	0.89036260	3.321253	5.375000	0.2278217	0.2459731
bees	4.422648	4.565636	0.4868290	0.42402345	5.918562	6.566667	0.6734998	0.6952746
birds	4.369286	4.977378	0.7215610	0.71599743	8.668470	16.100000	0.2713842	0.3091539
dung beetles	5.315462	5.311206	0.5398856	0.47765506	7.360628	4.707692	1.1291014	1.1281974
grasshoppers	10.302813	13.066031	0.4429446	0.21935050	13.603454	9.883333	1.0424432	1.3220268
hoverflies	2.712127	2.715689	0.2952258	0.35880584	3.274409	3.416667	0.7937934	0.7948359
large mammals	1.449550	1.757279	0.4820361	0.52610752	1.747907	2.223881	0.6518112	0.7901857
millipedes	1.537036	1.588697	0.3648274	0.19594032	1.717636	1.180328	1.3022107	1.3459794
other beetles	4.407666	4.440843	0.2808776	0.02846282	4.896712	8.474576	0.5201046	0.5240195
other wasps	2.202698	2.311666	0.7849297	0.64746250	3.955223	3.183333	0.6919471	0.7261777
parasitoid wasps	9.155373	11.071037	0.6323623	0.63498536	13.628816	15.866667	0.5770193	0.6977544
plants	14.742975	15.752919	0.5160170	0.54984963	20.651638	51.901639	0.2840561	0.3035149
snails	3.952801	4.102049	0.6542708	0.48725356	5.616235	6.683333	0.5914416	0.6137729
spiders	2.028644	2.041353	0.3571189	0.08658692	2.188988	5.000000	0.4057288	0.4082707
springtails	1.476286	1.516282	0.7114562	0.70200378	2.416109	4.530612	0.3258469	0.3346749
true bugs	1.965560	2.363758	0.3342719	0.13383209	2.303421	2.066667	0.9510772	1.1437537

3.3 Upscaling

Figure 4 shows the predicted and upscaled SR for all animal taxa and plants subtracted by the Areas where the models were not applicable. It also shows the final biodiversity map which only includes areas where all models were applicable. The model for spider prediction showed the smallest AOA, while the AOA of the models for millipede and springtail prediction nearly spanned the whole research area. The model for plant, ant, bee, bird, dung beetle, and snail prediction was not applicable in large areas at the peak of Kilimanjaro. Most taxa show higher SR in lower elevation than in higher elevation. The area where all the taxa's models were applicable is limited resulting in very fragmented final biodiversity map. In addition to the peak of the mountain, the areas of the maize fields, the lower montane forests and the home gardens have been narrowed down under consideration of the AOA.

Some of the predictions clearly show from which variables they were driven the most. This is especially the case for predictions where only a few layers were used. The prediction of the millipedes represents the landuse driver layer quite well, while the true bugs predictions looks similar to the vertical distance to the next stream and both had only two layers used for prediction (table 4). Furthermore taxa with a lower amount of predictor variables tend to have higher relative RMSE values (table 3, 6).

Table 4 shows the SR for each landcover class. The table and the biodiversity maps show the highest SR in home gardens (58.72), maize plantations (59.90), savanna (60.27) and lower montane forests (61.13), all being located at the lowest elevation. Also, overgrown and cleared areas (57.65) and forest plantations (56.72) showed high SR values. The lowest SR was predicted to be on bare rock (32.08) and in helichrysum vegetation (35.79). But also grassland (40.17), podocarpus forests (disturbed 41.87 and undisturbed 45.64), swamps (45.62), and Erica forest (46.62) showed quite low SR.

Table 4: Mean SR for each Landcover Class

Landcover Class	**Mean SR**	**Mean elevation**
Bare rock	32.08	4260.173
Helichrysum vegetation	35.79	4028.882
Grassland	40.17	3167.727
Podocarpus forest (disturbed)	41.87	3095.403
Swamp	45.62	1116.116
Podocarpus forest	45.64	2733.638
Erica forest (disturbed and undisturbed)	46.62	3142.951
Ocotea forest	50.93	2538.471
Coffee plantation	51.50	1144.851
Riverine	55.80	1063.843
Forest plantation	56.72	1716.760
overgrown/clearing	57.65	1620.752
Home garden	58.72	1045.151
Maize plantation	59.90	1228.137
savanna	60.27	1063.844
lower montane forest	61.13	1999.214

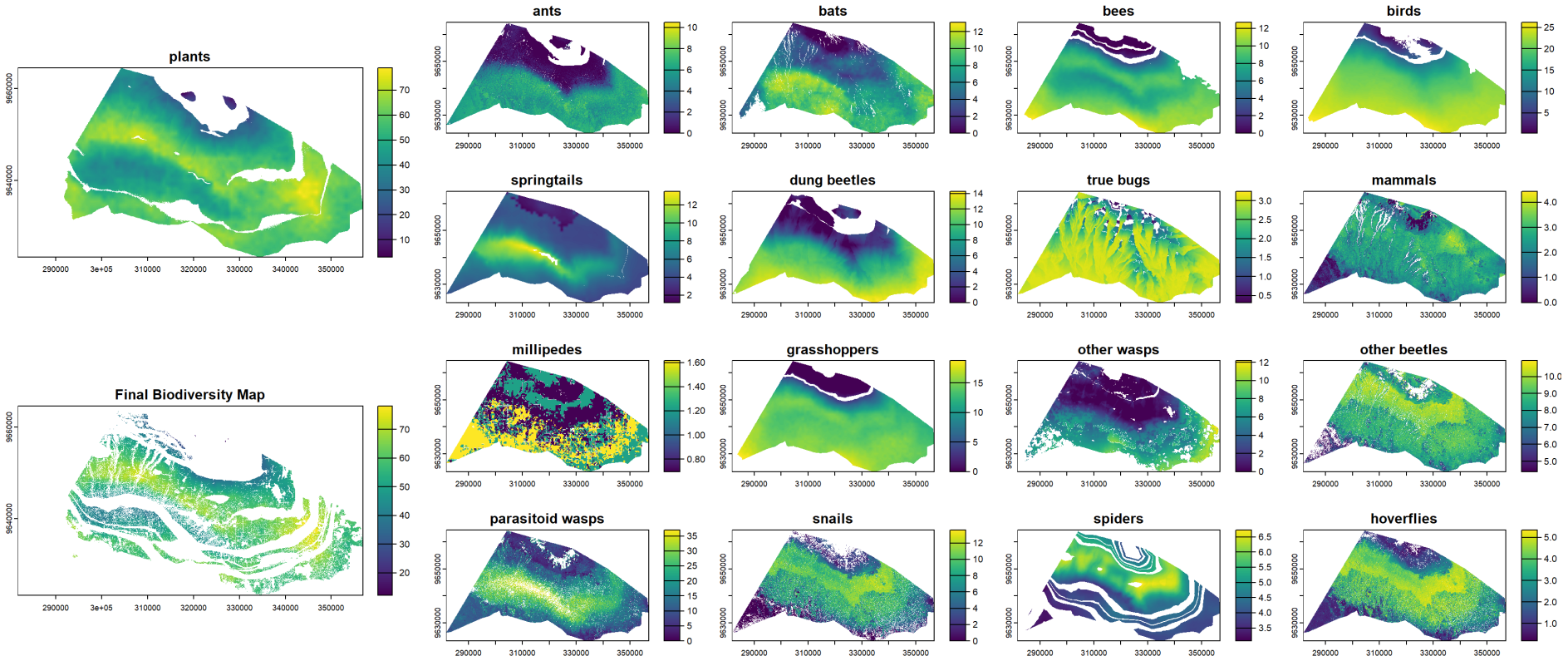


Figure 4: Upscaled biodiversity for all animal taxa and plant SR. All areas where the models were not applicable according to AOA [18] were removed from the map. Final biodiversity map includes the SR of plants and all animal taxa and only areas where all the models were applicable according to AOA.

4 Discussion

4.1 Variable Selection

Due to limited computational resources for variable selection, the VSURF method was employed instead of the FFS method used by Ziegler et al. [13]. Despite this change, the use of VSURF resulted in models with very small differences in RMSE values between training and testing datasets, indicating strong generalization capabilities (table 3). The VSURF method proved effective in identifying the most important variables while maintaining low correlations between them. This approach successfully avoided overfitting, even though the training data was limited and the number of potential predictor variables was high. Consequently, VSURF enabled the creation of robust models that could reliably predict biodiversity patterns across different taxa, provided there were appropriate predictor variables available for the specific taxon. While the VSURF method excels in selecting the best variables, it cannot compensate for the absence of suitable predictors.

The first goal of the study was to determine if upscaled plant biodiversity can be used as a proxy for animal SR prediction. Surprisingly, plant biodiversity was never selected as a predictor in the models, precluding confirmation that plant SR drives animal SR (table 6). Despite numerous studies suggesting a strong correlation between plant diversity and animal diversity [4], this project did not support this relationship, indicating that other factors might play a more significant role in driving animal SR in the ecosystems examined.

The variable selected the most was the vertical distance to stream (table 2). Based on the fact that water flows down hill, the vertical distance to stream can be used as an indicator for general soil moisture [19]. Pixels being higher up than the next water flow path in their watershed are likely to be drier than pixels which are roughly on the same height as the water flow path of their watershed. This variable was successfully used before to predict bryophytes SR [20] and proved to be a valuable variable in this study too. Many other studies found a positive correlation between soil moisture and different biodiversity indices, based on the fact the water availability can drastically change plant community compositions and distribution [21, 22, 23]. 5 out of 6 times the vertical distance to stream layer was used in combination with the precipitation in the driest quater layer of WorldClim (BIO17) (table 2). Also BIO17 was only chosen in combination with the vertical distance to stream, which underlines the connection between those to layers. Together they provide a robust measure for soil moisture variability, especially for critical conditions in dry month.

Despite the vertical distance to stream being selected most frequently, the most important variable proved to be elevation. Although the elevation layer (DEM) was never chosen directly (table 6), elevation significantly influences temperature. As elevation increases, temperature decreases, resulting in a strong correlation between the DEM and all 11 temperature-related WorldClim layers with the DEM being the driving factor. These temperature layers were present in every single prediction (table 6), highlighting the critical importance of the DEM. When considering its influence on temperature, the DEM emerges as the most important layer by far.

The calculated standard deviation layers were infrequently selected (table 6), and when included, they contributed minimally to the model's performance. Consequently, the SVH could not be confirmed.

4.2 Prediction Validation

The predictive models developed in this study demonstrated varying degrees of reliability across different taxa. Models for bats and plants exhibited the lowest nor-

malized RMSE values, indicating high predictive accuracy. In contrast, models for taxa such as grasshoppers, millipedes, and other beetles displayed higher RMSE values (table 3), suggesting less reliable predictions. The small differences between training and testing RMSE values for most taxa indicate that the models generalized well and did not overfit the data, reflecting the robust variable selection and cross-validation strategies employed.

However, significant variability was observed among the 20 models calculated for each taxon in the outer CV loop (figure 5), resulting from the different data splits created for each model. While the training RMSE values remained relatively consistent across different models, the testing RMSE values fluctuated considerably. This shows the impact different plots in testing and training can have on the validation results. This problem occurred because of the limited amount of data which resulted in 48 training data points and a even more limited testing data set never exceeding 12 data points. Consequently, the validation process became extremely unstable and unreliable. No validation strategy can fully rectify this flaw of to few data points.

Despite the models showing small differences between training and testing RMSE values (table 3), these good RMSE values of testing can be misleading. They indicate good model performance for the tested 12 data points; however, these points are unlikely to encompass the full range of possible values needed to evaluate the model's true performance. Consequently, the selected model may perform well within the value ranges of the testing data but could perform poorly on other value ranges, which cannot be detected due to their absence in the limited testing dataset. This represents a significant flaw in the study design which has to deal with the limitations in data sampling capacity, especially for a large area like Kilimanjaro.

Additionally, the challenges in prediction and upscaling are exacerbated by the diversity of land use types. Twelve different land use types were sampled, with a maximum of five repetitions per type, leaving only one plot per land use type for testing. Moreover, predictor variables influence biodiversity differently across land use types, complicating the identification of driving variables. For instance, precipitation might be a driving factor in undisturbed Podocarpus forest but its impact is probably negligible on maize and coffee plantations where irrigation is possible. This could also explain why the DEM emerged as the most significant factor, as land use types change primarily with elevation, while other variables influence biodiversity in other ways on different land use types leaving their impact undetectable.

The final issue with evaluating the model's performance is that the R-squared value, which is usually an important measure of how much variance the model explains in the data, may not be helpful in this case. This is because the variance and range of values in the training, validation, and testing datasets can differ significantly depending which data points are chosen for testing and training. Consequently, the R-squared value can become less reliable due to the way it is calculated. Even if a model can explain the variance of the data quite well, this might not be apparent if the training and testing datasets have substantially different variances. Therefore in this study mostly the RSME was taken into account, when validating model performance.

4.3 Upscaling and Land Use

Due to previously mentioned limitations, the upscaled data lacks precision and should not be considered definitive. However, the generated biodiversity maps serve as approximate indicators of actual biodiversity (figure 4). The lowest SR in the biodiversity maps was observed in areas characterized by bare rock, Helichrysum vegetation,

grassland and disturbed Podocarpus forest (table 4). These land cover types are found at the highest elevations, ranging from 3000 to 4000 meters. Such high altitudes are associated with extreme temperatures and harsh weather conditions, including significant snowfall and ice, which limit the number of species that can survive in these environments [24].

This phenomenon, known as environmental filtering, has also been documented by Galván-Cisneros et al. on Kilimanjaro and other mountains [25]. Additionally, other factors also contribute to the reduced SR at high elevations, such as the limited area available on peaks [26], seasonal time constraints [27], and geological time constraints [28]. These factors collectively influence biodiversity patterns in mountainous regions.

The highest SR in the biodiversity maps was observed in the lower montane forest and savanna, both exceeding a mean SR of 60 (table 4). This finding aligns with the research of Hemp et al., who identified the lower montane forest and savanna as areas with high species richness on Mount Kilimanjaro [29]. In contrast, forest plantations and clearings exhibited lower SR compared to the lower montane forests, indicating that intensive human land use significantly reduces species richness in these regions. This observation is consistent with Hemp et al.'s findings, which noted a gap in species richness between two elevation peaks at 1000m and 1800m due to strong human impact over the last centuries [30]. The high SR at 1000m is largely attributed to the biodiversity within home gardens [30, 31], corroborating the findings of a mean SR of 58.72 in home gardens (table 4). In this context, human impact appears to positively affect biodiversity in home gardens. However, in other cases, human land use generally leads to reduced biodiversity, as seen in the Podocarpus forests, where disturbed forests exhibited lower mean SR compared to undisturbed forests (41.87 vs. 45.64) (table 4).

The analysis revealed an unexpectedly high SR in maize plantations, contrary to other studies that have reported significantly lower SR in such areas. Peters et al. found that maize plantations exhibited 50% lower plant diversity and 30% lower animal diversity compared to the savanna from which they are often converted [32]. This discrepancy could be attributed to the similar abiotic conditions, such as elevation and climate, between the original savanna and the converted maize plantations. In the models, these abiotic factors were the primary predictors of SR, whereas land use was rarely selected as a predictor. This could have led to an overestimation of SR in maize plantations in this study.

Conversely, according to Peters et al. coffee plantations, which are typically established on cleared rainforest areas, showed smaller decreases in biodiversity compared to the original rainforest[32]. He states that the climatic conditions of an area converted to agriculture is the main driver of biodiversity changes due to land use, with dry conditions resulting in a more drastic change in biodiverity after conversion to agriculture. The relatively small impact on SR observed following the conversion of rainforest to coffee plantations in this study aligns with findings from previous research [32]. However, a potential methodological issue similar to that identified in the maize plantation analysis may have influenced these results. Specifically, the models predominantly selected abiotic factors as predictors, which remain consistent despite changes in land use. This reliance on abiotic variables could have led to an underestimation of the true impact of land use changes on biodiversity.

5 Conclusion

Contrary to expectations, the VSURF method did not select plant biodiversity as

a predictor for animal biodiversity, emphasizing the complexity of factors that influence biodiversity in the Kilimanjaro region. Elevation emerged as the most critical predictor, underscoring its role in shaping biodiversity patterns.

However, the projects methodological limitations, particularly the small sample size and the reliance on abiotic factors, necessitate cautious interpretation of the results of this study. The upscaled data, while providing a broad overview, lacks precision and should be viewed as an indicator rather than an exact representation of biodiversity. Still comparisons with other studies show consistency in the trends identified, such as the higher biodiversity in natural vegetation compared to disturbed areas (home gardens excluded), reaffirming the negative impact of human land use on biodiversity. However, the overestimation of species richness in maize plantations highlights the difficulty in modeling biodiversity across varied land use types, where using mixed land use training data can obscure the true impact of human activities specific to each type.

This study contributes to the understanding of biodiversity on Kilimanjaro, aligning with the project's aim to emphasize the intrinsic value of biodiversity, as discussed in the introduction. It also highlights the necessity for future research to refine these findings, particularly through more detailed analyses of individual taxa and a deeper investigation into the effects of land use on biodiversity. This approach will be crucial in developing more accurate and comprehensive conservation strategies that recognize both the intrinsic and instrumental values of biodiversity.

References

[1] Eduardo Sonnewend Brondízio et al. "Global assessment report on biodiversity and ecosystem services of the Intergovernmental Science-Policy Platform on Biodiversity and Ecosystem Services". In: (2019).

[2] W. Jetz et al. "Essential biodiversity variables for mapping and monitoring species populations". In: *Nature Ecology Evolution* 3 (2019), pp. 539–551. DOI: 10.1038/s41559-019-0826-1.

[3] Solen Le Clec'h et al. "Mapping ecosystem services at the regional scale: the validity of an upscaling approach". In: *International Journal of Geographical Information Science* 32.8 (2018), pp. 1593–1610. DOI: 10.1080/13658816.2018.1445256.

[4] B. Castagneyrol and H. Jactel. "Unraveling plant-animal diversity relationships: a meta-regression analysis." In: *Ecology* 93 9 (2012), pp. 2115–24. DOI: 10.1890/11-1300.1.

[5] Jordi Bascompte and Pedro Jordano. "Plant-Animal Mutualistic Networks: The Architecture of Biodiversity". In: *Annual Review of Ecology, Evolution, and Systematics* 38 (2007), pp. 567–593. DOI: 10.2307/30033872.

[6] Y.M. Rosas, P.L. Peri, M.V. Lencinas, et al. "Improving the knowledge of plant potential biodiversity-ecosystem services links using maps at the regional level in Southern Patagonia". In: *Ecological Processes* 10 (2021), p. 53. DOI: 10.1186/s13717-021-00326-0.

[7] Clara I. Nicholls and Miguel A. Altieri. "Plant biodiversity enhances bees and other insect pollinators in agroecosystems. A review". In: *Agronomy for Sustainable Development* 33 (2013), pp. 257–274. DOI: 10.1007/s13593-012-0092-y.

[8] Claudia Hemp et al. *The KiLi Project: Kilimanjaro ecosystems under global change: Linking biodiversity, biotic interactions and biogeo-*

chemical ecosystem processes. Sept. 2018. ISBN: 978-3-929907-96-4.

[9] E. Gómez-Baggethun et al. "The history of ecosystem services in economic theory and practice: From early notions to markets and payment schemes". In: *Ecological Economics* 69 (2010), pp. 1209–1218. DOI: 10.1016/J.ECOLECON.2009.11.007.

[10] Gopi Upreti. "Valuation of Biodiversity, Ecosystem Services, and Natural Capital". In: *Ecosociocentrism: The Earth First Paradigm for Sustainable Living*. Cham: Springer Nature Switzerland, 2023, pp. 163–188. ISBN: 978-3-031-41754-2. DOI: 10.1007/978-3-031-41754-2_8. URL: https://doi.org/10.1007/978-3-031-41754-2_8.

[11] M. Davidson. "On the relation between ecosystem services, intrinsic value, existence value and economic valuation". In: *Ecological Economics* 95 (2013), pp. 171–177. DOI: 10.1016/J.ECOLECON.2013.09.002.

[12] Senckenberg Gesellschaft für Naturforschung. *Study Region - Kili-SES Project*. Accessed: 2024-07-27. 2024. URL: https://kili-ses.senckenberg.de/en/project/study-region/.

[13] Alice Ziegler et al. "Potential of Airborne LiDAR Derived Vegetation Structure for the Prediction of Animal Species Richness at Mount Kilimanjaro". In: *Remote Sensing* 14.3 (2022). ISSN: 2072-4292. DOI: 10.3390/rs14030786. URL: https://www.mdpi.com/2072-4292/14/3/786.

[14] Michael W. Palmer et al. "Opportunities for Long-Term Ecological Research at the Tallgrass Prairie Preserve, Oklahoma". In: 2000. URL: https://api.semanticscholar.org/CorpusID:198411980.

[15] Robin Genuer, Jean-Michel Poggi, and Christine Tuleau- Malot. "VSURF: An R Package for Variable Selection Using Random Forests". In: *The R Journal* 7.2 (2015), pp. 19–33. DOI: 10.32614/RJ-2015-018. URL: https://doi.org/10.32614/RJ-2015-018.

[16] Jaime Lynn Speiser et al. "A comparison of random forest variable selection methods for classification prediction modeling". In: *Expert Systems with Applications* 134 (2019), pp. 93–101. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2019.05.028. URL: https://www.sciencedirect.com/science/article/pii/S0957417419303574.

[17] Paul Geladi and Bruce R. Kowalski. "Partial least-squares regression: a tutorial". In: *Analytica Chimica Acta* 185 (1986), pp. 1–17. ISSN: 0003-2670. DOI: https://doi.org/10.1016/0003-2670(86)80028-9. URL: https://www.sciencedirect.com/science/article/pii/0003267086800289.

[18] H. Meyer and E. Pebesma. "Predicting into unknown space? Estimating the area of applicability of spatial prediction models". In: *Methods in Ecology and Evolution* 12 (2020), pp. 1620–1633. DOI: 10.1111/2041-210X.13650.

[19] Gabriel S. Oltean, Philip G. Comeau, and Barry White. "Linking the Depth-to-Water Topographic Index to Soil Moisture on Boreal Forest Sites in Alberta". In: *Forest Science* 62.2 (Jan. 2016), pp. 154–165. ISSN: 0015-749X. DOI: 10.5849/forsci.15-054. eprint: https://academic.oup.com/forestscience/article-pdf/62/2/154/23177419/2878.pdf. URL: https://doi.org/10.5849/forsci.15-054.

[20] Samuel Bartels et al. "Relating Bryophyte Assemblages to a Remotely Sensed Depth-to-Water Index in Boreal Forests". In: *Frontiers in Plant Science* 9 (June 2018). DOI: 10.3389/fpls.2018.00858.

[21] Chaturvedi Rk and R. As. "Effect of Soil Moisture on Composition and Diversity of Trees in Tropical dry Forest". In: 3 (2018). DOI: 10.15406/MOJES.2018.03.00059.

[22] Gao-Lin Wu et al. "Interactions of soil water content heterogeneity and species diversity patterns in semi-arid steppes on the Loess Plateau of China". In: *Journal of Hydrology* 519 (2014), pp. 1362–1367.

[23] Xin-Feng ZHAO et al. "Effects of nutrient and water additions on plant community structure and species diversity in desert grasslands". In: *Chinese Journal of Plant Ecology* 38.2 (2014), p. 167.

[24] Ram Dani, Pradeep Divakar, and Chitra Baniya. *Diversity and Composition Plants Species Along Elevational Gradient: Research Trends*. Nov. 2022. DOI: 10.21203/rs.3.rs-2268968/v1.

[25] Carlos Galván-Cisneros et al. "Altitude as environmental filtering influencing phylogenetic diversity and species richness of plants in tropical mountains". In: *Journal of Mountain Science* 20 (Feb. 2023), pp. 285–298. DOI: 10.1007/s11629-022-7687-9.

[26] G Grabherr et al. "Patterns and current changes in alpine plant diversity". In: *Arctic and alpine biodiversity: patterns, causes and ecosystem consequences* (1995), pp. 167–181.

[27] Caroline M Williams et al. "Understanding Evolutionary Impacts of Seasonality: An Introduction to the Symposium". In: *Integrative and Comparative Biology* 57.5 (Oct. 2017), pp. 921–933. ISSN: 1540-7063. DOI: 10.1093/icb/icx122. eprint: https://academic.oup.com/icb/article-pdf/57/5/921/24324956/icx122.pdf. URL: https://doi.org/10.1093/icb/icx122.

[28] Erich Tasser, Mirijam Mader, and Ulrike Tappeiner. "Effects of land use in alpine grasslands on the probability of landslides". In: *Basic and Applied Ecology* 4.3 (2003), pp. 271–280. ISSN: 1439-1791. DOI: https://doi.org/10.1078/1439-1791-00153. URL: https://www.sciencedirect.com/science/article/pii/S1439179104701218.

[29] A. Hemp. "Ecology of the Pteridophytes on the Southern Slopes of Mt. Kilimanjaro. Part II: Habitat Selection". In: *Plant Biology* 3.5 (2001), pp. 493–523. DOI: https://doi.org/10.1055/s-2001-17729. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1055/s-2001-17729. URL: https://onlinelibrary.wiley.com/doi/abs/10.1055/s-2001-17729.

[30] Hemp Andreas. "Altitudinal zonation and diversity patterns in the forests of Mount Kilimanjaro, Tanzania". In: *Tropical Montane Cloud Forests: Science for Conservation and Management* (Jan. 2011), pp. 134–141. DOI: 10.1017/CBO9780511778384.014.

[31] E.C.M. Fernandes, A. Oktingati, and J. Maghembe. "The Chagga home gardens: A multi-storeyed agroforestry cropping system on Mt. Kilimanjaro, Northern Tanzania". In: *Food and Nutrition Bulletin* 7.3 (1985), pp. 1–8. DOI: 10.1007/BF00131267.

[32] Marcell K Peters et al. "Climate–land-use interactions shape tropical mountain biodiversity and ecosystem functions". In: *Nature* 568.7750 (2019), pp. 88–92.

[33] European Space Agency. *Sentinel-2 MSI: MultiSpectral Instrument, Level-2A*. Dataset. Accessed on July 28, 2024. 2024. URL: https://scihub.copernicus.eu/dhus.

[34] Wildlife Conservation Society (WCS) and Center for International Earth Science Information Network (CIESIN)/Columbia University. *Last of the Wild Project, Version 2, 2005 (LWP-2): Global Human Influence Index (HII) Dataset (Geographic)*. Version 2.00. Accessed: 2024-07-28. 2005. DOI: 10.7927/H4BP00QC. URL: https://doi.org/10.7927/H4BP00QC.

[35] Stephen E. Fick and Robert J. Hijmans. "WorldClim 2: new 1km spatial resolution climate surfaces for global land areas". In: *International Journal of Climatology* 37.12 (2017), pp. 4302–4315.

[36] Tomislav Hengl et al. "SoilGrids250m: Global gridded soil information based on machine learning". In: *PLoS one* 12.2 (2017), e0169748.

Appendix

Additional tables

Table 5: Predictor variables, their sources, and variable classes. Sentinel-2 and Human Influence Index data obtained from Google Earth Engine [33, 34], World clim data obtained from here, soil grid obtained from here [35], soil groups and pH data obtained from here [36].

Variable	Source	Variable Class
slope	DEM	Topographical
aspect	DEM	Topographical
TPI (Topographic Position Index)	DEM	Topographical
TRI (Topographic Ruggedness Index)	DEM	Topographical
distance to stream	DEM	Topographical
vertical distance to stream	DEM/Watershed analysis	Topographical
DEM (Digital Elevation Model)	DEM	Topographical
TWI (Topographic Wetness Index)	DEM	Topographical
HLI (Heat Load Index)	DEM	Topographical
Visible Sky	DEM	Topographical
B1-B12	Sentinel-2	Sentinel
AOT (Aerosol Optical Thickness)	Sentinel-2	Sentinel
WVP (Water Vapor Pressure)	Sentinel-2	Sentinel
SCL (Scene Classification Layer)	Sentinel-2	Sentinel
TCI (True Color Image)	Sentinel-2	Sentinel
NDVI (Normalized Difference Vegetation Index)	Sentinel-2	Indices
EVI (Enhanced Vegetation Index)	Sentinel-2	Indices
NDWI (Normalized Difference Water Index)	Sentinel-2	Indices

Continued on next page

Table 5 – continued from previous page

Variable	Source	Variable Class
SAVI (Soil Adjusted Vegetation Index)	Sentinel-2	Indices
NBR (Normalized Burn Ratio)	Sentinel-2	Indices
GNDVI (Green Normalized Difference Vegetation Index)	Sentinel-2	Indices
RENDVI (Red Edge Normalized Difference Vegetation Index)	Sentinel-2	Indices
MSAVI (Modified Soil Adjusted Vegetation Index)	Sentinel-2	Indices
ARVI (Atmospherically Resistant Vegetation Index)	Sentinel-2	Indices
NDBI (Normalized Difference Built-up Index)	Sentinel-2	Indices
BSI (Bare Soil Index)	Sentinel-2	Indices
NDRE (Normalized Difference Red Edge)	Sentinel-2	Indices
GSAVI (Green Soil Adjusted Vegetation Index)	Sentinel-2	Indices
RDVI (Renormalized Difference Vegetation Index)	Sentinel-2	Indices
WDRVI (Wide Dynamic Range Vegetation Index)	Sentinel-2	Indices
copernicus landcover	Copernicus	Sentinel
pH predicted (0-20 cm)	SoilGrids	Soil
pH predicted (20-50 cm)	SoilGrids	Soil
pH standard deviation (0-20 cm)	SoilGrids	Soil
pH standard deviation (20-50 cm)	SoilGrids	Soil
soil groups	SoilGrids	Soil
wc2.1 (bioclimatic variables 1-19)	WorldClim	Climate

Continued on next page

Table 5 – continued from previous page

Variable	Source	Variable Class
hii (human impact index)	NASA	Human Impact Index
hii distance to roads	NASA	Human Impact Index
hii landuse driver	NASA	Human Impact Index
hii population density driver	NASA	Human Impact Index
hii power driver	NASA	Human Impact Index

Table 6: Variables selected for different taxa

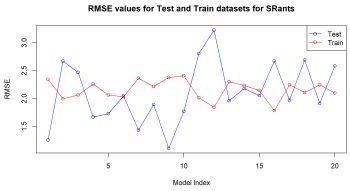
Taxon	Variable
ants	vertical_distance_to_stream_mean, wc2.1_30s_bio_8_mean, wc2.1_30s_bio_4_mean, wc2.1_30s_bio_3_mean, hii_landuse_driver_mean, wc2.1_30s_bio_17_mean, B6_mean, wc2.1_30s_bio_15_sd, B7_mean, B8A_sd, wc2.1_30s_bio_16_sd
bats	vertical_distance_to_stream_mean, wc2.1_30s_bio_9_mean, soil_grids_mean, wc2.1_30s_bio_7_mean, wc2.1_30s_bio_17_mean, wc2.1_30s_bio_2_mean, HLI_mean, ARVI_mean, distance_to_stream_mean, wc2.1_30s_bio_13_mean, wc2.1_30s_bio_14_sd
bees	wc2.1_30s_bio_4_mean, wc2.1_30s_bio_8_mean, wc2.1_30s_bio_16_mean
birds	wc2.1_30s_bio_3_sd, wc2.1_30s_bio_3_mean, wc2.1_30s_bio_4_mean, pH_standard_deviation_20_50_cm_mean, wc2.1_30s_bio_8_mean, pH_predicted_mean_20_50_cm_sd, SCL_sd
dung beetles	vertical_distance_to_stream_mean, wc2.1_30s_bio_7_mean, wc2.1_30s_bio_10_mean, wc2.1_30s_bio_9_mean, wc2.1_30s_bio_3_mean, wc2.1_30s_bio_2_sd
grasshoppers	WDRVI_mean, wc2.1_30s_bio_4_mean
hoverflies	BSI_mean, ARVI_mean
large mammals	Visible_Sky_mean, GNDVI_mean, B12_mean, DEM_UTM37S_sd, distance_to_stream_mean
millipedes	BSI_mean, hii_landuse_driver_mean
other beetles	wc2.1_30s_bio_17_sd, B7_sd, BSI_mean, B11_mean
other wasps	vertical_distance_to_stream_mean, wc2.1_30s_bio_17_mean, wc2.1_30s_bio_10_mean, soil_grids_mean, wc2.1_30s_bio_4_mean, wc2.1_30s_bio_8_mean, AOT_mean, HLI_sd, wc2.1_30s_bio_18_sd

Continued on next page

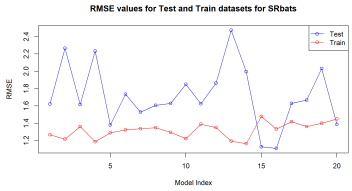
Table 6 – continued from previous page

Taxon	**Variable**
parasitoid wasps	EVI_mean, B8_mean, wc2.1_30s_bio_16_mean, RDVI_sd, Visible_Sky_sd
plants	WDRVI_mean, vertical_distance_to_stream_mean, wc2.1_30s_bio_17_mean, wc2.1_30s_bio_3_mean, wc2.1_30s_bio_7_mean, wc2.1_30s_bio_19_mean, wc2.1_30s_bio_19_sd, wc2.1_30s_bio_14_mean
snails	SAVI_mean, NDBI_mean, wc2.1_30s_bio_1_mean, MSAVI_mean, NDVI_mean, BSI_mean, GNDVI_mean, wc2.1_30s_bio_8_sd, pH_standard_deviation_20_50_cm_sd
spiders	AOT_mean, RENDVI_sd, wc2.1_30s_bio_19_mean, wc2.1_30s_bio_2_mean
springtails	wc2.1_30s_bio_12_mean, NBR_mean, wc2.1_30s_bio_18_mean
true bugs	wc2.1_30s_bio_17_mean, vertical_distance_to_stream_mean, wc2.1_30s_bio_15_mean, wc2.1_30s_bio_4_sd

Additional Figures

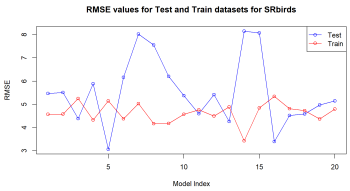


Ants

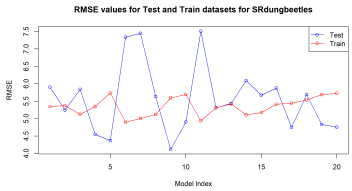


Bats

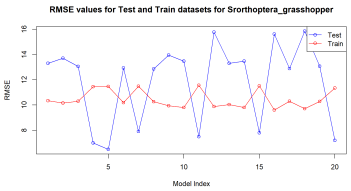
Bees



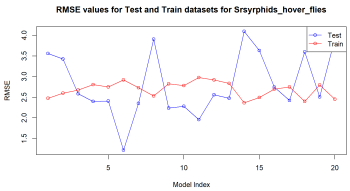
Birds



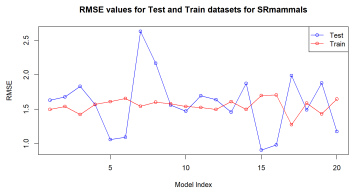
Dung Beetles



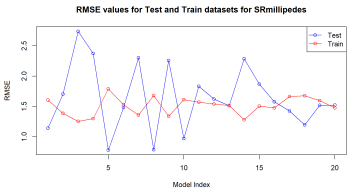
Grasshoppers



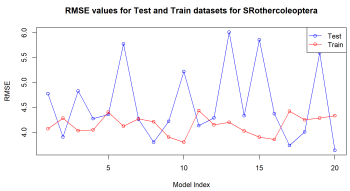
Hover Flies



Mammals

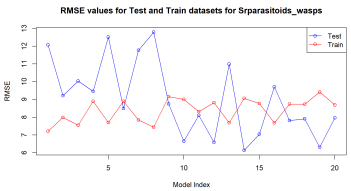


Millipedes

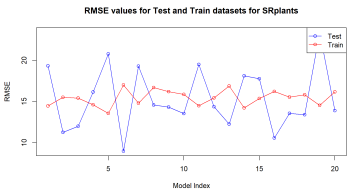


Other Coleoptera

Other Wasps

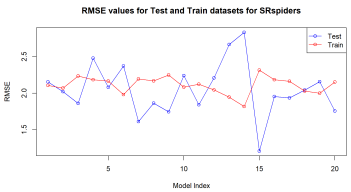


Parasitoid Wasps

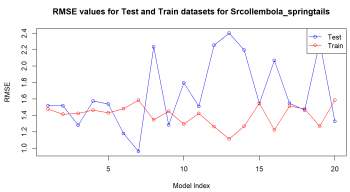


Plants

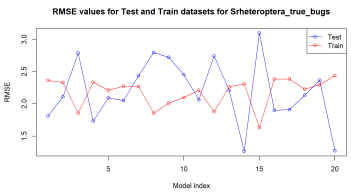
Snails



Spiders



Springtails



True Bugs

Figure 5: RMSE of models for different taxa from the outer cross-validation (CV) analysis.

R and Python Code

Data preparation

```r
# This script processes Sentinel-2 imagery to calculate various vegetation indices, merges the results,
# and saves the output to a file. The script involves reading a shapefile for the study area,
# reading and merging Sentinel-2 raster data, calculating indices such as NDVI, EVI, NDWI, SAVI, and others,
# and finally plotting and saving the computed indices.

library(terra)
library(sf)

polygons <- st_read("data/study_area/VegAug1_KILI_SES.shp")
merged_polygon <- st_union(polygons)
plot(merged_polygon)

sentinel <- rast("data/sentinel/new_merged_sentinel.tif")
sentinel_path <- list.files("data/sentinel/raw", full.names = TRUE)
sentinel_list <- lapply(sentinel_path, rast)
merged_sentinel <- do.call(merge, sentinel_list)
writeRaster(merged_sentinel, "data/sentinel/new_merged_sentinel_full.tif")

names(sentinel)

# Calculate NDVI
ndvi <- (sentinel$B8 - sentinel$B4) / (sentinel$B8 + sentinel$B4)
names(ndvi) <- "NDVI"

# Calculate EVI
evi <- 2.5 * (sentinel$B8 - sentinel$B4) / (sentinel$B8 + 6 * sentinel$B4 - 7.5 * sentinel$B2 + 1)
names(evi) <- "EVI"

# Calculate NDWI
ndwi <- (sentinel$B3 - sentinel$B8) / (sentinel$B3 + sentinel$B8)
names(ndwi) <- "NDWI"

# Calculate SAVI
L <- 0.5
savi <- (sentinel$B8 - sentinel$B4) / (sentinel$B8 + sentinel$B4 + L) * (1 + L)
names(savi) <- "SAVI"

# Calculate NBR
nbr <- (sentinel$B8 - sentinel$B12) / (sentinel$B8 + sentinel$B12)
names(nbr) <- "NBR"

# Calculate GNDVI
gndvi <- (sentinel$B8 - sentinel$B3) / (sentinel$B8 + sentinel$B3)
names(gndvi) <- "GNDVI"

# Calculate RENDVI
rendvi <- (sentinel$B8A - sentinel$B5) / (sentinel$B8A + sentinel$B5)
names(rendvi) <- "RENDVI"

```

```r
# Calculate MSAVI
msavi <- (2 * sentinel$B8 + 1 - sqrt((2 * sentinel$B8 + 1)^2 - 8 * (sentinel$B8 - sentinel$B4))) / 2
names(msavi) <- "MSAVI"

# Calculate ARVI
arvi <- (sentinel$B8 - (2 * sentinel$B4 - sentinel$B2)) / (sentinel$B8 + (2 * sentinel$B4 - sentinel$B2))
names(arvi) <- "ARVI"

# Calculate NDBI
ndbi <- (sentinel$B11 - sentinel$B8) / (sentinel$B11 + sentinel$B8)
names(ndbi) <- "NDBI"

# Calculate BSI
bsi <- ((sentinel$B11 + sentinel$B4) - (sentinel$B8 + sentinel$B2)) / ((sentinel$B11 + sentinel$B4) + (sentinel$B8 + sentinel$B2))
names(bsi) <- "BSI"

# Calculate NDRE
ndre <- (sentinel$B8A - sentinel$B5) / (sentinel$B8A + sentinel$B5)
names(ndre) <- "NDRE"

# Calculate GSAVI
gsavi <- (sentinel$B8 - sentinel$B3) / (sentinel$B8 + sentinel$B3 + L) * (1 + L)
names(gsavi) <- "GSAVI"

# Calculate RDVI
rdvi <- (sentinel$B8 - sentinel$B4) / sqrt(sentinel$B8 + sentinel$B4)
names(rdvi) <- "RDVI"

# Calculate WDRVI
a <- 0.2
wdrvi <- (a * sentinel$B8 - sentinel$B4) / (a * sentinel$B8 + sentinel$B4)
names(wdrvi) <- "WDRVI"

# Stack all indices
indices2 <- c(ndvi, evi, ndwi, savi, nbr, gndvi, rendvi, msavi, arvi, ndbi, bsi, ndre, gsavi, rdvi, wdrvi)

# Plot the indices
plot(indices2, main = c("NDVI", "EVI", "NDWI", "SAVI", "NBR", "GNDVI", "RENDVI", "MSAVI", "ARVI",
                        "NDBI", "BSI", "NDRE", "GSAVI", "RDVI", "WDRVI"))

# Save the indices to a file
writeRaster(indices2, "data/sentinel/_new_sentinel_and_indices.tif", overwrite = TRUE)

# Plot each index with its name
lapply(names(indices2), function(name) {
  plot(indices2[[name]], main = name)
})
```

Listing 1: R code for Sentinel processing

```r
# This script processes a Digital Elevation Model (DEM) to derive various topographic metrics using terra and whitebox packages.
```

```r
# The script includes calculating terrain attributes (slope, aspect, TPI, TRI), filling depressions, calculating flow direction
# and accumulation, identifying stream networks, calculating the Topographic Wetness Index (TWI), and other derived indices.
# The results are saved to files and plotted.

library(terra)
library(whitebox)

dem <- rast("data/topographic/DEM_UTM37S.tif")
topographics <- terrain(dem, c("slope", "aspect", "TPI", "TRI"), unit = "radians")

# Write DEM to a temporary file for whitebox processing
writeRaster(dem, "temp/temp_dem.tif", overwrite = TRUE)

# Fill sinks using whitebox
wbt_fill_depressions("temp/temp_dem.tif", "temp/filled_dem.tif")
filled_dem <- rast("temp/filled_dem.tif")

# Calculate flow direction using whitebox
wbt_d8_pointer("temp/filled_dem.tif", "temp/flowdir.tif")
flowdir <- rast("temp/flowdir.tif")

# Calculate flow accumulation using whitebox
wbt_d8_flow_accumulation("temp/filled_dem.tif", "temp/flowacc.tif")
flowacc <- rast("temp/flowacc.tif")

# Calculate TWI using the formula: TWI = ln(a / tan(beta))
twi <- log(flowacc / tan(topographics[[1]]))

# Define a threshold for streams (e.g., cells with flow accumulation greater than a threshold are streams)
threshold <- 10000
streams <- flowacc > threshold
plot(streams)
writeRaster(streams, "temp/streams.tif", overwrite = TRUE)

# Calculate euclidean distance to streams
wbt_euclidean_distance("temp/streams.tif", "temp/distance_to_stream.tif")
euclidean_distance <- rast("temp/distance_to_stream.tif")
vertical_distance_to_stream <- rast("data/topographic/relative_height.tif")

# Calculate Heat Load Index (HLI)
hli <- 0.339 + 0.808 * cos(topographics$aspect - 0.785) * sin(topographics$slope) +
  0.196 * sin(topographics$aspect - 0.785) * sin(topographics$slope) -
  0.482 * cos(topographics$slope)

# Process visible sky layer from saga
visible_sky <- rast("data/topographic/visible_sky.tif")
plot(visible_sky)
visible_sky <- crop(visible_sky, ext(hli))
visible_sky <- resample(visible_sky, hli, method = "bilinear")

# Merge everything
topographics <- c(topographics, euclidean_distance, vertical_distance_to_stream, dem, twi, hli, visible_sky)
```

```r

names(topographics)[6] <- "vertical_distance_to_stream"
names(topographics)[8] <- "TWI"
names(topographics)[9] <- "HLI"
topographics[[5]][!is.finite(topographics[[5]])] <- NA
plot(topographics)
writeRaster(topographics, "data/topographic/topographics.tif", overwrite = TRUE)
names(topographics)
```

Listing 2: Calculation of Topographical Variables

```r
# This script calculates the distance to roads and railways from given raster data representing human impact.
# It involves reading the road and railway raster files, identifying non-road and non-railway cells, calculating distances to the nearest roads and railways,
# and saving the distance rasters to files.

library(terra)
library(whitebox)

roads_driver <- rast("data/human_impact/Roads_Driver.tif")
railway_driver <- rast("data/human_impact/Railways_Driver.tif")

roads <- roads_driver != 800
railways <- railway_driver != 800

distance_to_roads <- distance(roads, target = TRUE, exclude = NA)
distance_to_railways <- distance(railways, target = TRUE, exclude = NA)

writeRaster(distance_to_roads, "data/human_impact/distance_to_roads.tif", overwrite = TRUE)
writeRaster(distance_to_railways, "data/human_impact/distance_to_railways.tif", overwrite = TRUE)
```

Listing 3: Calculation of Distance to Roads and Railways

```r
# This script calculates watersheds for given points using a Digital Elevation Model (DEM) and flow accumulation data.
# It involves defining file paths, initializing whitebox tools, calculating watersheds, and converting the resulting raster to vector polygons.
# The results are loaded and plotted for verification.

library(whitebox)
library(terra)

# Define file paths for the DEM and flow accumulation file
dem_file <- "data/topographic/DEM_UTM37S.tif"
flow_accum_file <- "temp/flowacc.tif"
points_file <- "data/data_lev2/merge_points.shp"
output_watersheds <- "temp/output_watersheds.tif"
flow_dir <- "temp/flowdir.tif"
output_vector_file <- "temp/watersheds.shp"

# Ensure whitebox tools are installed
whitebox::wbt_init()

# Calculate watersheds for each point
```

```r
wbt_watershed(
  d8_pntr = flow_dir,              # Input D8 pointer file (can be DEM if
    pointer not available)
  pour_pts = points_file,          # Shapefile containing points
  output = output_watersheds       # Output raster file for watersheds
)

# Load the output to verify
watersheds <- rast(output_watersheds)
plot(watersheds)

# Convert the watershed raster to vector polygons
wbt_raster_to_vector_polygons(
  input = output_watersheds,  # Input raster file
  output = output_vector_file      # Output vector shapefile
)
```

Listing 4: Calculation of watersheds for vertical distance to stream caluclation

```r
# This script processes spatial data to extract mean and standard deviation
    values of predictors for each plot,
# combines these with biodiversity data, and saves the resulting dataset to
    a CSV file.
# This data resulting data is used for model training

library(terra)
library(corrplot)
library(dplyr)

# Load plot polygons and predictor rasters
plots <- vect("data/study_area/BPolygon.shp")
predictors <- rast("data/upscalling_data/predictors.tif")

# Extract the mean values of the predictors for each polygon
model_data_mean <- extract(predictors, plots, fun = mean, na.rm = TRUE)
model_data_mean <- model_data_mean[,-(c(30:34, 78))]
colnames(model_data_mean) <- paste0(colnames(model_data_mean), "_mean")

# Extract the standard deviation values of the predictors for each polygon
model_data_sd <- extract(predictors, plots, fun = sd, na.rm = TRUE)
model_data_sd <- model_data_sd[,-(c(1, 30:34, 78))]
colnames(model_data_sd) <- paste0(colnames(model_data_sd), "_sd")

# Combine mean and standard deviation data
model_data <- cbind(model_data_mean, model_data_sd)
model_data$ID_mean <- plots$PlotID
colnames(model_data)[1] <- "plotID"

# Load biodiversity data and merge with the model data
SR <- read.table("data/model_data/Biodiversity_Data_Marcel.csv", sep = ";",
    header = TRUE, dec = ",")
SR <- SR[,c(1, 2, 14:31)]
model_data <- purrr::reduce(list(model_data, SR), dplyr::left_join, by = '
    plotID')

# Save the combined data to a CSV file
write.csv(model_data, "data/model_data/new_model_data_all_master.csv", row.
```

```
    names = FALSE)
```

Listing 5: Calculation of Model Data

```
# This script processes various raster datasets including Sentinel imagery, topographic data, climate, soil, land cover data, and human impact data.
# The data is cropped and resampled to match the study area and combined into a single raster stack for further analysis.
# The final combined raster is saved to a file.
# This resulting data is used for upscaling.

library(terra)
library(sf)

# Load and transform study area
study_area <- st_read("data/study_area/VegAug1_KILI_SES.shp")
study_area <- sf::st_transform(study_area, 32737)
study_area <- vect(study_area)

# Load and crop Sentinel data to study area
sentinel <- rast("data/sentinel/new_sentinel_and_indices.tif")
sentinel <- crop(sentinel, study_area, mask = TRUE)

# Load, resample, and crop topographic data to match Sentinel data and study area
topographics <- rast("data/topographic/topographics.tif")
topographics <- resample(topographics, sentinel[[1]])
topographics <- crop(topographics, study_area, mask = TRUE)

# Load, project, resample, and crop climate, soil, and land cover data to study area
climate_ph_soil_lc <- lapply(list.files("data/other", full.names = TRUE, pattern = "\\.tif$"), rast)
climate_ph_soil_lc[[4]] <- project(climate_ph_soil_lc[[4]], sentinel[[1]])
climate_ph_soil_lc[[1]] <- resample(climate_ph_soil_lc[[1]], sentinel[[2]])
climate_ph_soil_lc[[2]] <- resample(climate_ph_soil_lc[[2]], sentinel[[2]])
climate_ph_soil_lc[[3]] <- resample(climate_ph_soil_lc[[3]], sentinel[[2]])
climate_ph_soil_lc[[4]] <- resample(climate_ph_soil_lc[[4]], sentinel[[2]])
climate_ph_soil_lc <- lapply(climate_ph_soil_lc, crop, y = study_area, mask = TRUE)
climate_ph_soil_lc <- rast(climate_ph_soil_lc)

# Load, resample, and crop human impact data to match Sentinel data and study area
human_impact <- rast(list.files("data/human_impact", full.names = TRUE, pattern = "\\.tif$"))
names(human_impact)[c(1, 2)] <- c("hii_distance_to_railways", "hii_distance_to_roads")
human_impact <- resample(human_impact, sentinel[[1]])
human_impact <- crop(human_impact, study_area, mask = TRUE)

# Combine all datasets into a single raster stack
final <- c(topographics, sentinel, climate_ph_soil_lc, human_impact)

# Save the combined raster stack
writeRaster(final, "data/upscalling_data/new_predictors.tif")

# Load and rename the combined raster stack
```

```r
upscaling_data <- rast("data/upscalling_data/new_predictors.tif")
names(upscaling_data) <- paste0(names(upscaling_data), "_mean")
writeRaster(upscaling_data, "data/upscalling_data/predictors_new.tif",
    overwrite = TRUE)
```

Listing 6: Calculation of Final Biodiversity Upscaling Stack

```python
# This script calculates the relative height of pixels in a DEM relative to
    the nearest river pixels within watershed areas.
# It involves loading DEM and river rasters, identifying watersheds, and
    computing the height difference for each pixel
# relative to the nearest river pixel. The resulting relative heights are
    saved as a new raster file.

import rasterio
import numpy as np
from scipy.spatial import cKDTree
import fiona
from shapely.geometry import shape
from rasterio.features import geometry_mask

def calculate_relative_height(dem_data, river_data, watershed_masks):
    relative_heights = np.full(dem_data.shape, np.nan, dtype=np.float32)

    for watershed_mask in watershed_masks:
        # Mask the DEM and river data for the current watershed
        watershed_dem = np.where(watershed_mask, dem_data, np.nan)
        watershed_river = np.where(watershed_mask, river_data, 0)

        # Identify river pixels and their DEM values
        river_indices = np.where(watershed_river == 1)
        river_dem_values = watershed_dem[river_indices]

        if river_indices[0].size == 0:
            continue

        # Convert river pixel indices to spatial coordinates
        river_points = np.column_stack((river_indices[0], river_indices[1])
    )

        # Build a spatial index (k-d tree) for efficient nearest neighbor
    search
        tree = cKDTree(river_points)

        # Prepare all pixel coordinates within the watershed
        watershed_indices = np.where(watershed_mask)
        all_pixel_indices = np.column_stack((watershed_indices[0],
    watershed_indices[1]))

        # Perform batch query to find nearest river pixel for all pixels in
     the watershed
        distances, nearest_river_indices = tree.query(all_pixel_indices, k
    =1)

        # Calculate relative heights for valid pixels
        valid_pixel_indices = (watershed_indices[0], watershed_indices[1])
        flat_dem_data = watershed_dem[valid_pixel_indices]
        nearest_river_dem_values = river_dem_values[nearest_river_indices]

```

```
        relative_heights[valid_pixel_indices] = flat_dem_data - nearest_river_dem_values

    return relative_heights

def load_raster(file_path):
    with rasterio.open(file_path) as src:
        data = src.read(1)  # Read the first band
        profile = src.profile
        nodata = src.nodata
        if nodata is not None:
            data = data.astype(np.float32)  # Ensure the data is in float format to handle NaNs
            data[data == nodata] = np.nan
    return data, profile

def save_raster(output_path, data, reference_raster_profile):
    profile = reference_raster_profile.copy()
    profile.update(dtype=rasterio.float32, count=1, nodata=np.nan)
    with rasterio.open(output_path, 'w', **profile) as dst:
        dst.write(data.astype(rasterio.float32), 1)

def load_watersheds(watershed_file, dem_shape, dem_transform):
    with fiona.open(watershed_file, 'r') as src:
        watersheds = [shape(feature['geometry']) for feature in src]

    watershed_masks = []
    for watershed in watersheds:
        mask = geometry_mask([watershed], transform=dem_transform, invert=True, out_shape=dem_shape)
        watershed_masks.append(mask)

    return watershed_masks

# Paths to your DEM, river raster, and watershed shapefile
dem_path = 'C:/Users/konst/Documents/Master/bip/berchtes_gaden/data/data_lev1/DEM.tif'
river_path = 'C:/Users/konst/Documents/Master/upscalling_biodiversity/final_proj/temp/streams.tif'
watershed_file = 'C:/Users/konst/Documents/Master/upscalling_biodiversity/final_proj/temp/watersheds.shp'
output_path = 'C:/Users/konst/Documents/Master/relative_height.tif'

try:
    # Load DEM and river rasters
    dem_data, dem_profile = load_raster(dem_path)
    river_data, _ = load_raster(river_path)
    river_data = river_data.astype(bool)  # Ensure river_data is boolean

    # Load watershed masks
    watershed_masks = load_watersheds(watershed_file, dem_data.shape, dem_profile['transform'])

    # Calculate relative height
    relative_heights = calculate_relative_height(dem_data, river_data, watershed_masks)

    # Save the result
```

```
    save_raster(output_path, relative_heights, dem_profile)

    print(f'Relative heights raster saved to: {output_path}')
except Exception as e:
    print(f'An error occurred: {e}')
```

Model training

```
# This script trains a PLSR model using VSURF and hyperparameter tuning.
# It performs outer cross-validation with 20 splits, evaluates model performance,
# and saves the results and best model to files.

library(CAST)
library(caret)
library(randomForest)
library(vip)
library(foreach)
library(doParallel)
library(progress)
library(dplyr)
source("src/functions/variable_selection.R")
source("src/functions/outer_cv_index.R")

# Load and preprocess the model data
model_data <- read.csv("data/model_data/model_data_all_master.csv")
model_data <- na.omit(model_data[, c(1:150, 168)])

set.seed(123) # For reproducibility

# Create stratified splits for outer cross-validation
splits <- lapply(1:20, function(x) create_stratified_split(model_data, stratify_by = "cat"))

# Initialize lists to store models and performances
models_list <- list()
performances_list <- list()

# Register parallel backend
cl <- makeCluster(detectCores() - 1) # Use one less than the total number of cores
registerDoParallel(cl)

pb <- progress_bar$new(
  format = "  Progress [:bar] :percent in :elapsed (ETA: :eta)",
  total = 20, # total number of tasks
  width = 60
)

# Define final variables for the model
final_vars <- c("DEM_UTM37S_mean", "hii_landuse_driver_mean", "hii_mean", "B4_mean", "hii_popdens_driver_mean",
                "vertical_distance_to_stream_mean", "wc2.1_30s_bio_16_mean", "hii_power_driver_mean",
                "copernicus_landcover_mean")

# Loop through each split for outer cross-validation
```

```r
results <- foreach(i = 1:20, .packages = c("CAST", "caret", "randomForest"), .combine = 'c') %dopar% {
  # Create training and testing datasets
  featuresTrain <- model_data[splits[[i]]$train, ]
  featuresTest <- model_data[splits[[i]]$test, ]

  set.seed(234)

  # Define train control for LOOCV
  trControl <- trainControl(method = "cv", number = 10)

  # Define the tuning grid for number of components
  tune_grid <- expand.grid(ncomp = seq(1, 6, by = 1))

  # Train the model
  ffs_model <- train(
    featuresTrain[, final_vars], # Select relevant feature columns
    as.numeric(featuresTrain$SRallplants), # Response variable
    method = "pls", # Random Forest
    metric = "RMSE", # Evaluation metric
    verbose = TRUE, # Verbose output
    tuneGrid = tune_grid, # Hyperparameter tuning grid
    trControl = trControl # Train control with LOOCV
  )

  # Predict on the test set
  featuresTest$pred_ffs <- predict(ffs_model, newdata = featuresTest[, final_vars])

  # Evaluate the model performance
  performance <- postResample(pred = featuresTest$pred_ffs, obs = featuresTest$SRallplants)

  # Update progress bar
  pb$tick()
  message(i)

  list(model = ffs_model, performance = performance)
}

# Stop parallel backend
stopCluster(cl)

split_list_by_index <- function(input_list) {
  odd_index_list <- list()
  even_index_list <- list()

  for (i in seq_along(input_list)) {
    if (i %% 2 == 1) {
      odd_index_list[[length(odd_index_list) + 1]] <- input_list[[i]]
    } else {
      even_index_list[[length(even_index_list) + 1]] <- input_list[[i]]
    }
  }

  return(list(odd_index_list = odd_index_list, even_index_list = even_index_list))
}
```

```
99
100 # Split results into models and performances
101 split_results <- split_list_by_index(results)
102 models_list <- split_results$odd_index_list
103 performances_list <- split_results$even_index_list
104
105 # Save models and performances
106 save(models_list, file = "data/models/plant_models_list_pls_sd_added.RData"
    )
107 save(performances_list, file = "data/models/
    plant_performances_list_pls_sd_added.RData")
108
109 load("~/Master/upscalling_biodiversity/final_proj/data/models/
    plant_models_list_pls.RData")
110 load("~/Master/upscalling_biodiversity/final_proj/data/models/
    plant_performances_list_pls.RData")
111 performances_list_mean <- performances_list
112 models_list_mean <- models_list
113
114 load("~/Master/upscalling_biodiversity/final_proj/data/models/
    plant_models_list_pls_sd_added.RData")
115 load("~/Master/upscalling_biodiversity/final_proj/data/models/
    plant_performances_list_pls_sd_added.RData")
116 performances_list_sd_added <- performances_list
117 models_list_sd_added <- models_list
118
119 # Convert list to data frame
120 performances_df_mean <- as.data.frame(do.call(rbind, performances_list_mean
    ))
121 performances_df_sd_added <- as.data.frame(do.call(rbind,
    performances_list_sd_added))
122
123 # Extract the best model performances from each model
124 best_model_performances_mean <- lapply(models_list_mean, function(model) {
125   best_model <- which.min(model$results$RMSE)
126   c(RMSE = model$results$RMSE[best_model], Rsquared =
    model$results$Rsquared[best_model], MAE = model$results$MAE[best_model])
127 })
128 best_model_performances_sd_added <- lapply(models_list_sd_added, function(
    model) {
129   best_model <- which.min(model$results$RMSE)
130   c(RMSE = model$results$RMSE[best_model], Rsquared =
    model$results$Rsquared[best_model], MAE = model$results$MAE[best_model])
131 })
132
133 # Convert the best model performances to a dataframe
134 best_model_df_mean <- as.data.frame(do.call(rbind,
    best_model_performances_mean))
135 best_model_df_sd_added <- as.data.frame(do.call(rbind,
    best_model_performances_sd_added))
136
137 # Append the best model performance to the existing performances dataframe
138 performances_df_mean <- cbind(performances_df_mean, best_model_df_mean)
139 performances_df_sd_added <- cbind(performances_df_sd_added,
    best_model_df_sd_added)
140
141 # Rename columns
142 colnames(performances_df_mean)[1:3] <- paste0(colnames(performances_df_mean
```

```
    )[1:3], "_test")
colnames(performances_df_mean)[4:6] <- paste0(colnames(performances_df_mean
    )[4:6], "_train")
colnames(performances_df_sd_added)[1:3] <- paste0(colnames(
    performances_df_sd_added)[1:3], "_test")
colnames(performances_df_sd_added)[4:6] <- paste0(colnames(
    performances_df_sd_added)[4:6], "_train")

# Extract R-squared and RMSE values for test and train datasets
rsquared_test_mean <- performances_df_mean$Rsquared_test
rsquared_train_mean <- performances_df_mean$Rsquared_train
rsquared_test_sd_added <- performances_df_sd_added$Rsquared_test
rsquared_train_sd_added <- performances_df_sd_added$Rsquared_train

rmse_test_mean <- performances_df_mean$RMSE_test
rmse_train_mean <- performances_df_mean$RMSE_train
rmse_test_sd_added <- performances_df_sd_added$RMSE_test
rmse_train_sd_added <- performances_df_sd_added$RMSE_train

# Plotting R-squared values
par(mfrow = c(1, 1))  # Set up the plotting area to have 1 row and 1 column

plot(rsquared_test_mean, type = "o", col = "blue", ylim = range(c(
    rsquared_test_mean, rsquared_train_mean)),
     ylab = "R-squared", xlab = "Model Index", main = "R-squared values for
     Test and Train datasets (Mean vs SD Added)")
points(rsquared_train_mean, type = "o", col = "red")
points(rsquared_test_sd_added, type = "o", col = "darkblue")
points(rsquared_train_sd_added, type = "o", col = "darkred")

# Add legend
legend("bottomleft", legend = c("Test Mean", "Train Mean", "Test SD Added",
     "Train SD Added"), col = c("blue", "red", "darkblue", "darkred"), lty =
     1, pch = 1)

# Plotting RMSE values
plot(rmse_test_mean, type = "o", col = "blue", ylim = range(c(
    rmse_test_mean, rmse_train_mean)),
     ylab = "RMSE", xlab = "Model Index", main = "RMSE values for Test and
    Train datasets (Mean vs SD Added)")
points(rmse_train_mean, type = "o", col = "red")
points(rmse_test_sd_added, type = "o", col = "darkblue")
points(rmse_train_sd_added, type = "o", col = "darkred")

# Add legend
legend("topright", legend = c("Test Mean", "Train Mean", "Test SD Added", "
    Train SD Added"), col = c("blue", "red", "darkblue", "darkred"), lty =
    1, pch = 1)

# Calculate absolute differences
performances_df_mean$RMSE_diff_mean <- abs(performances_df_mean$RMSE_test -
     performances_df_mean$RMSE_train)
performances_df_mean$Rsquared_diff <- abs(
    performances_df_mean$Rsquared_test - performances_df_mean$Rsquared_train
    )
performances_df_mean$MAE_diff  <- abs(performances_df_mean$MAE_test -
    performances_df_mean$MAE_train)

```

```r
# Sum the differences
performances_df_mean$total_diff <- performances_df_mean$RMSE_diff +
    performances_df_mean$Rsquared_diff + performances_df_mean$MAE_diff

# Identify the model with the smallest total difference
best_model_index <- which.min(performances_df_mean$total_diff)
best_model <- performances_df_mean[best_model_index, ]

# Print the details of the best model
print(best_model)
final_model <- models_list[[as.integer(row.names(best_model))]]

# Plot variable importance
plot(varImp(final_model))
saveRDS(final_model, "data/results/plants/allPlants_ffs_rf_cv_20_10.rds")
```

Listing 7: Model Training of Plant Prediciton

```r
# This script trains a Partial Least Squares (PLS) model using forward
    feature selection and hyperparameter tuning.
# It performs outer cross-validation with 20 splits, evaluates model
    performance,
# and saves the results and best model to files.

library(CAST)
library(terra)
library(caret)
library(randomForest)
library(vip)
library(foreach)
library(doParallel)
library(progress)
library(dplyr)
library(VSURF)
source("src/functions/variable_selection.R")
source("src/functions/outer_cv_index.R")

# Load and preprocess the model data
model_data <- read.csv("data/model_data/new_model_data_all_master.csv")

plant_sr <- rast("data/results/plants/prediction_pls_ffs.tif")

plots <- vect("data/study_area/BPolygon.shp")

plant_sr_extract <- extract(plant_sr, plots, FUN = mean)

model_data <- cbind(model_data, plant_sr_extract)

# Variable for the response column index
response_col_index <- 161

# Store the original column names
response_name <- names(model_data[, c(1, response_col_index)])[2]

# Perform NA omission and remove specified column
# here 168 is the plant SR
model_data <- na.omit(model_data[, c(1:150, 168, response_col_index)])
# after this step plant SR is in column 151

```

```r
model_data <- model_data[, !names(model_data) %in% "hii_distance_to_railways_mean"]
# after this step plant SR is at position 150

# Update response column index based on the column name
response_col_index <- which(names(model_data) == response_name)

set.seed(123) # For reproducibility

# Create stratified splits for outer cross-validation
splits <- lapply(1:20, function(x) create_stratified_split(model_data, stratify_by = "cat"))

# Initialize lists to store models and performances
models_list <- list()
performances_list <- list()

# Register parallel backend
cl <- makeCluster(detectCores() - 1) # Use one less than the total number of cores
registerDoParallel(cl)

pb <- progress_bar$new(
  format = "  Progress [:bar] :percent in :elapsed (ETA: :eta)",
  total = 20, # total number of tasks
  width = 60
)

set.seed(345)

# Perform variable selection using VSURF
# excluded column 1 and 149 (plotId and cat) for VSURF
VSRUF_resu <- VSURF(x = model_data[c(2:148, 150)], y = model_data[, response_col_index], nmin = 0.01, nsd = 0.01, nmj = 0.01)
final_vars <- colnames(model_data[, VSRUF_resu$varselect.pred])
final_vars

# Loop through each split for outer cross-validation
results <- foreach(i = 1:20, .packages = c("CAST", "caret", "randomForest"), .combine = 'c') %dopar% {
  # Create training and testing datasets
  featuresTrain <- model_data[splits[[i]]$train, ]
  featuresTest <- model_data[splits[[i]]$test, ]

  set.seed(234)

  # Define train control for LOOCV
  trControl <- trainControl(method = "cv", number = 10)

  # Define the tuning grid for number of components
  tune_grid <- expand.grid(ncomp = seq(1, length(final_vars) - 1, by = 1))

  # Train the model with forward feature selection and hyperparameter tuning
  ffs_model <- train(
    featuresTrain[, final_vars], # Select relevant feature columns
    as.numeric(featuresTrain[, response_col_index]), # Response variable
    method = "pls", # Partial Least Squares
```

```
    metric = "RMSE", # Evaluation metric
    verbose = TRUE, # Verbose output
    tuneGrid = tune_grid, # Hyperparameter tuning grid
    trControl = trControl # Train control with LOOCV
  )
  ffs_model
  # Predict on the test set
  featuresTest$pred_ffs <- predict(ffs_model, newdata = featuresTest[,
    final_vars])

  # Evaluate the model performance
  performance <- postResample(pred = featuresTest$pred_ffs, obs =
    featuresTest[, response_col_index])

  # Update progress bar
  pb$tick()
  message(i)

  list(model = ffs_model, performance = performance)
}

# Stop parallel backend
stopCluster(cl)

split_list_by_index <- function(input_list) {
  odd_index_list <- list()
  even_index_list <- list()

  for (i in seq_along(input_list)) {
    if (i %% 2 == 1) {
      odd_index_list[[length(odd_index_list) + 1]] <- input_list[[i]]
    } else {
      even_index_list[[length(even_index_list) + 1]] <- input_list[[i]]
    }
  }

  return(list(odd_index_list = odd_index_list, even_index_list =
    even_index_list))
}

# Split results into models and performances
split_results <- split_list_by_index(results)
models_list <- split_results$odd_index_list
performances_list <- split_results$even_index_list

performances_list_mean <- performances_list
models_list_mean <- models_list

# Convert list to data frame
performances_df_mean <- as.data.frame(do.call(rbind, performances_list_mean
    ))

# Extract the best model performances from each model
best_model_performances_mean <- lapply(models_list_mean, function(model) {
  best_model <- which.min(model$results$RMSE)
  c(RMSE = model$results$RMSE[best_model], Rsquared =
    model$results$Rsquared[best_model], MAE = model$results$MAE[best_model])
})
```

```r

# Convert the best model performances to a dataframe
best_model_df_mean <- as.data.frame(do.call(rbind,
    best_model_performances_mean))

# Append the best model performance to the existing performances dataframe
performances_df_mean <- cbind(performances_df_mean, best_model_df_mean)

# Rename columns
colnames(performances_df_mean)[1:3] <- paste0(colnames(performances_df_mean
    )[1:3], "_test")
colnames(performances_df_mean)[4:6] <- paste0(colnames(performances_df_mean
    )[4:6], "_train")

# Extract R-squared and RMSE values for test and train datasets
rsquared_test_mean <- performances_df_mean$Rsquared_test
rsquared_train_mean <- performances_df_mean$Rsquared_train

rmse_test_mean <- performances_df_mean$RMSE_test
rmse_train_mean <- performances_df_mean$RMSE_train

# Plotting R-squared values
par(mfrow = c(1, 1))  # Set up the plotting area to have 1 row and 1 column

plot(rsquared_test_mean, type = "o", col = "blue", ylim = range(c(
    rsquared_test_mean, rsquared_train_mean)),
     ylab = "R-squared", xlab = "Model Index", main = "R-squared values for
     Test and Train datasets (Mean vs SD Added)")
points(rsquared_train_mean, type = "o", col = "red")

# Plotting RMSE values
plot(rmse_test_mean, type = "o", col = "blue", ylim = range(c(
    rmse_test_mean, rmse_train_mean)),
     ylab = "RMSE", xlab = "Model Index", main = "RMSE values for Test and
    Train datasets (Mean vs SD Added)")
points(rmse_train_mean, type = "o", col = "red")

# Calculate absolute differences
performances_df_mean$RMSE_diff_mean <- abs(performances_df_mean$RMSE_test -
     performances_df_mean$RMSE_train)
performances_df_mean$Rsquared_diff <- abs(
    performances_df_mean$Rsquared_test - performances_df_mean$Rsquared_train
    )
performances_df_mean$MAE_diff  <- abs(performances_df_mean$MAE_test -
    performances_df_mean$MAE_train)

# Sum the differences
performances_df_mean$total_diff <- performances_df_mean$RMSE_diff_mean +
    performances_df_mean$Rsquared_diff + performances_df_mean$MAE_diff

# Identify the model with the smallest total difference
best_model_index <- which.min(performances_df_mean$total_diff)
best_model <- performances_df_mean[best_model_index, ]

# Print the details of the best model
print(best_model)
final_model <- models_list[[as.integer(row.names(best_model))]]

```

```r
# Plot variable importance
plot(varImp(final_model))

# Save the final model in a directory named after the column name of the animal type
dir.create(file.path("data/results", response_name), showWarnings = FALSE)
saveRDS(final_model, file.path("data/results", response_name, paste0(response_name, "_ffs_pls_cv_20_10.rds")))
saveRDS(best_model, file.path("data/results", response_name, paste0(response_name, "_pls_model_performance.rds")))
source("src/upscalling/animals_upscaling_pls.R")
```

Listing 8: Model Training for all Animal Taxa

Upscaling

```r
# This script performs predictions using a trained PLS model on upscaling data, calculates the Area of Applicability (AOA),
# and trains the Discrepancy Index (DI). The results are saved as raster files.

# Load necessary libraries
library(terra)
library(CAST)
library(caret)
library(doParallel)
library(parallel)

# Load upscaling data
upscaling_data <- rast("data/upscalling_data/predictors.tif")
names(upscaling_data) <- paste0(names(upscaling_data), "_mean")

# Load the trained model
final_model <- readRDS("data/models/plants/allPlants_ffs_pls_cv_20_10.rds")

# Subset the upscaling data to only include selected variables from the final model
upscaling_data <- upscaling_data[[colnames(final_model$trainingData[,-length(colnames(final_model$trainingData))])]]

# Predict using the final model
prediction_plant_pls <- predict(upscaling_data, final_model, na.rm = TRUE)

# Write the prediction results to a file
writeRaster(prediction_plant_pls, "data/results/plants/prediction_pls_ffs.tif", overwrite = TRUE)

# Set up parallel processing
cl <- makeCluster(detectCores() - 1)
registerDoParallel(cl)

# Calculate AOA
AOA <- aoa(upscaling_data, model = final_model)

# Stop parallel processing
stopCluster(cl)
registerDoSEQ()
```

```r

# Train Discrepancy Index
train_DI <- trainDI(model = final_model)

# Plot the Discrepancy Index (DI)
plot(AOA$DI)

# Plot the Area of Applicability (AOA)
plot(AOA$AOA)

# Save the DI and AOA rasters
writeRaster(AOA$DI, "data/results/plants/DI_prediction_pls_ffs.tif")
writeRaster(AOA$AOA, "data/results/plants/AOA_prediction_pls_ffs.tif")
```

Listing 9: Plant SR Upscaling

```r
# This script performs predictions using a trained PLS model on upscaling data, calculates the Area of Applicability (AOA),
# and trains the Discrepancy Index (DI). The results are saved as raster files.
# Also missing SD layers are calculated here, because they were not all calculated before

# Load necessary libraries
library(terra)
library(CAST)
library(caret)
library(doParallel)
library(parallel)
library(future)
library(future.apply)

# Load upscaling data
upscaling_data <- rast("data/upscalling_data/predictors.tif")
dem <- upscaling_data$pH_standard_deviation_20_50_cm_mean

# Define a function to calculate SD within a moving window
moving_window_sd <- function(x) {
  if (all(is.na(x))) {
    return(NA)
  } else {
    return(sd(x, na.rm = TRUE))
  }
}

# Apply the focal function with a 10-pixel radius window
sd_raster <- focal(dem, w = 5, fun = moving_window_sd)
plot(sd_raster)
names(sd_raster) <- "pH_standard_deviation_20_50_cm_sd"

# Load and prepare upscaling data
names(upscaling_data) <- paste0(names(upscaling_data), "_mean")
upscaling_data <- c(upscaling_data, sd_raster)

# Load the trained model
final_model <- readRDS("data/models/plants/allPlants_ffs_pls_cv_20_10.rds")

# Subset the upscaling data to only include selected variables from the final model
```

```r
upscaling_data <- upscaling_data[[colnames(final_model$trainingData[,-length(colnames(final_model$trainingData))])]]

# Predict using the final model
prediction_plant_pls <- predict(upscaling_data, final_model, na.rm = TRUE)
plot(prediction_plant_pls)

# Write the prediction results to a file
writeRaster(prediction_plant_pls, file.path("data/results", "plants", "prediction_pls_ffs.tif"), overwrite = TRUE)

# Set up parallel processing
cl <- makeCluster(detectCores() - 1)
registerDoParallel(cl)

# Calculate AOA
AOA <- aoa(upscaling_data, model = final_model)

# Stop parallel processing
stopCluster(cl)
registerDoSEQ()

# Plot the Discrepancy Index (DI)
plot(AOA$DI)

# Plot the Area of Applicability (AOA)
plot(AOA$AOA)

# Save the DI and AOA rasters
writeRaster(AOA$DI, file.path("data/results", "plants", "DI_prediction_pls_ffs.tif"), overwrite = TRUE)
writeRaster(AOA$AOA, file.path("data/results", "plants", "AOA_prediction_pls_ffs.tif"), overwrite = TRUE)
```

Listing 10: Animal Taxa Upcaling

Data analysis

```r
# This script processes multiple biodiversity prediction raster files, applies a mask using AOA raster files,
# sums the predictions, and saves the final biodiversity map. It also plots the results.

library(terra)

# Get list of all prediction_pls_ffs.tif files and AOA files in subfolders
tif_files <- list.files(path = "data/results", pattern = "^prediction_pls_ffs\\.tif$", full.names = TRUE, recursive = TRUE)
aoa_files <- list.files(path = "data/results", pattern = "AOA_prediction_pls_ffs\\.tif$", full.names = TRUE, recursive = TRUE)

# Read and stack all AOA raster files
aoa_rasters <- lapply(aoa_files[1:17], rast)
aoa_rasters <- rast(aoa_rasters)

# Read and stack all prediction raster files
rasters <- lapply(tif_files[1:17], rast)
```

```r
predictions <- rast(rasters)
predictions[predictions < 0] <- 0

# Apply mask to predictions using AOA rasters
final <- mask(predictions, aoa_rasters, maskvalues = 0)

# Mapping of Latin names to English names
lyr_names <- c("plants", "ants", "bats", "bees", "birds", "springtails", "dung beetles", "true bugs", "mammals",
               "millipedes", "grasshoppers", "other wasps", "other beetles", "parasitoid wasps",
               "snails", "spiders", "hoverflies")

# Assign layer names to the final raster stack
names(final) <- lyr_names

# Sum all prediction rasters
sum_raster <- Reduce("+", predictions)

# Create a mask based on AOA rasters
result <- app(aoa_rasters, fun = function(x) all(x != 0))
final2 <- mask(sum_raster, result, maskvalues = 0)

# Correct layout matrix for 4 rows and 5 columns
layout_matrix <- matrix(c(1, 3, 4, 5, 6,
                          1, 7, 8, 9, 10,
                          2, 11, 12, 13, 14,
                          2, 15, 16, 17, 18), nrow = 4, byrow = TRUE)

# Apply the layout with adjusted column widths
layout(layout_matrix, widths = c(1.5, 1, 1, 1, 1))

# Plot the first raster spanning two rows and the first column
plot(final[[1]], main = names(final[[1]]))

# Plot the second raster spanning two rows and the second column
plot(final2, main = "Final Biodiversity Map")

# Plot the remaining rasters in a 4x4 grid
for(i in 2:17){
  plot(final[[i]], main = names(final[[i]]))
}

# Save the final biodiversity map to a file
writeRaster(final2, "data/results/final_biodiversity_map.tif")
```

Listing 11: Creation of Biodiversity Maps

```r
# This script processes model performance data, calculates relative RMSE, and visualizes the number of variables chosen for each taxon using a boxplot.

# Load necessary libraries
library(terra)
library(ggplot2)

# Get performance data files
performances_path <- list.files(path = "data/results", pattern = "_pls_model_performance.rds", full.names = TRUE, recursive = TRUE)
```

```r
performances_list <- lapply(performances_path, function(x) {
  df <- readRDS(x)
  return(df[, 1:6])
})

# Extract taxon names from file names
taxon_names <- c("plants", "ants", "bats", "bees", "birds", "springtails",
    "dung beetles", "true bugs", "large mammals",
                 "millipedes", "grass hoppers", "other wasps", "other
    beetles", "parasitoid wasps", "snails", "spiders",
                 "hover flies")

# Combine all performance data into a dataframe and add taxon names
performances_df <- do.call(rbind, performances_list)
performances_df <- cbind(Taxon = taxon_names, performances_df)

# Load model data and calculate SD and mean values
model_data <- read.csv("data/model_data/new_model_data_all_master.csv")
sd_values <- sapply(model_data[, 151:168], function(x) sd(x, na.rm = TRUE))
mean_values <- sapply(model_data[, 151:168], function(x) mean(x, na.rm =
    TRUE))

# Mapping of Latin names to column names
taxon_sd_mapping <- c(
  "plants" = "SRallplants",
  "ants" = "SRants",
  "bats" = "SRbats",
  "bees" = "SRbees",
  "birds" = "SRbirds",
  "springtails" = "Srcollembola_springtails",
  "dung beetles" = "SRdungbeetles",
  "true bugs" = "Srheteroptera_true_bugs",
  "large mammals" = "SRmammals",
  "millipedes" = "SRmillipedes",
  "grass hoppers" = "Srorthoptera_grasshopper",
  "other wasps" = "Srotheraculeata_other_wasps",
  "other beetles" = "SRothercoleoptera",
  "parasitoid wasps" = "Srparasitoids_wasps",
  "snails" = "SRsnails",
  "spiders" = "SRspiders",
  "hover flies" = "Srsyrphids_hover_flies"
)

# Add the standard deviation and mean values to the dataframe
performances_df$SD <- sapply(performances_df$Taxon, function(x) sd_values[
    taxon_sd_mapping[x]])
performances_df$Mean <- sapply(performances_df$Taxon, function(x)
    mean_values[taxon_sd_mapping[x]])

# Relative RMSE Normalization
performances_df$Relative_RMSE_test <- performances_df$RMSE_test /
    performances_df$Mean
performances_df$Relative_RMSE_train <- performances_df$RMSE_train /
    performances_df$Mean

# Select and reorder columns
performances_df <- performances_df[, c("Taxon", "RMSE_train", "RMSE_test",
    "Rsquared_train", "Rsquared_test", "SD", "Mean", "Relative_RMSE_train",
```

```
    "Relative_RMSE_test")]

# Sort the dataframe by the Taxon column alphabetically
performances_df <- performances_df[order(performances_df$Taxon), ]

# Print the updated dataframe
print(performances_df)

# Data for number of variables chosen by VSURF for each taxon
variables_chosen <- c(8, 11, 11, 3, 7, 3, 6, 4, 5, 2, 2, 9, 4, 5, 9, 4, 2)
taxa <- c("plants", "ants", "bats", "bees", "birds", "springtails", "dung beetles", "true bugs", "large mammals",
          "millipedes", "grass hoppers", "other wasps", "other beetles", "parasitoid wasps", "snails", "spiders",
          "hover flies")

# Create a data frame for the variables chosen data
data <- data.frame(taxa, variables_chosen)

# Create the boxplot with ggplot2
ggplot(data, aes(x = "", y = variables_chosen)) +
  geom_boxplot() +
  geom_point(position = position_jitter(width = 0.4, seed = 13)) +
  geom_text(aes(label = taxa),
            position = position_jitter(width = 0.4, seed = 13),
            hjust = -0.1,
            vjust = -0.5,
            size = 3) +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank()) +
  labs(title = "Number of Variables Chosen by VSURF for Each Taxon",
       x = "",
       y = "Number of Variables Chosen") +
  theme(legend.position = "none")  # Remove the legend if not needed
```

Listing 12: Model performance analysis

```
# This script loads trained models, extracts chosen variables, counts their occurrences,
# maps them to their types, and prints a sorted dataframe of the results.

# Load necessary libraries
library(terra)

# Load models
models_path <- list.files(path = "data/results", pattern = "_ffs_pls_cv_20_10.rds", full.names = TRUE, recursive = TRUE)
models <- lapply(models_path, readRDS)

# Extract taxon names from file names
taxon_names <- basename(models_path)
taxon_names <- sub("_ffs_pls_cv_20_10.rds", "", taxon_names)

# Initialize an empty list to store the results
results <- list()

# Loop through each model and extract the taxon and chosen variables
for (i in seq_along(models)) {
  taxon <- taxon_names[i]
```

```
  variables <- colnames(models[[i]]$ptype)
  results[[i]] <- data.frame(Taxon = taxon, Variables = I(list(variables)))
}

# Combine all results into a single dataframe
results_df <- do.call(rbind, results)

# Flatten the list of variables
all_variables <- unlist(results_df$Variables)

# Count the occurrences of each variable
variable_counts <- table(all_variables)

# Convert the table to a dataframe for easier viewing
variable_counts_df <- as.data.frame(variable_counts)
colnames(variable_counts_df) <- c("Variable", "Count")

# Remove "_mean" suffix from variable names
variable_counts_df$Variable <- gsub("_mean", "",
    variable_counts_df$Variable)

# Sort the dataframe by the number of counts in descending order
variable_counts_df <- variable_counts_df[order(-variable_counts_df$Count),
    ]

# Print the sorted dataframe
print(variable_counts_df)

# Define the mapping of each variable to its type
variable_types <- c(
  "AOT" = "sentinel",
  "ARVI" = "indicie",
  "BSI" = "indicie",
  "B11" = "sentinel",
  "B12" = "sentinel",
  "B6" = "sentinel",
  "B7" = "sentinel",
  "B7_sd" = "sentinel",
  "B8" = "sentinel",
  "B8A_sd" = "sentinel",
  "DEM_UTM37S_sd" = "topographic",
  "distance_to_stream" = "topographic",
  "EVI" = "indicie",
  "GNDVI" = "indicie",
  "hii_landuse_driver" = "climate",
  "HLI" = "topographic",
  "HLI_sd" = "topographic",
  "MSAVI" = "indicie",
  "NBR" = "indicie",
  "NDBI" = "indicie",
  "NDVI" = "indicie",
  "pH_predicted_20_50_cm_sd" = "soil",
  "pH_standard_deviation_20_50_cm" = "soil",
  "pH_standard_deviation_20_50_cm_sd" = "soil",
  "RDVI_sd" = "indicie",
  "RENDVI_sd" = "indicie",
  "SAVI" = "indicie",
  "SCL_sd" = "sentinel",
```

```
  "soil_grids" = "soil",
  "vertical_distance_to_stream" = "topographic",
  "Visible_Sky" = "topographic",
  "Visible_Sky_sd" = "topographic",
  "WDRVI" = "indicie",
  "wc2.1_30s_bio_1" = "climate",
  "wc2.1_30s_bio_2" = "climate",
  "wc2.1_30s_bio_2_sd" = "climate",
  "wc2.1_30s_bio_3" = "climate",
  "wc2.1_30s_bio_3_sd" = "climate",
  "wc2.1_30s_bio_4" = "climate",
  "wc2.1_30s_bio_4_sd" = "climate",
  "wc2.1_30s_bio_7" = "climate",
  "wc2.1_30s_bio_8" = "climate",
  "wc2.1_30s_bio_8_sd" = "climate",
  "wc2.1_30s_bio_9" = "climate",
  "wc2.1_30s_bio_10" = "climate",
  "wc2.1_30s_bio_12" = "climate",
  "wc2.1_30s_bio_13" = "climate",
  "wc2.1_30s_bio_14" = "climate",
  "wc2.1_30s_bio_14_sd" = "climate",
  "wc2.1_30s_bio_15" = "climate",
  "wc2.1_30s_bio_15_sd" = "climate",
  "wc2.1_30s_bio_16" = "climate",
  "wc2.1_30s_bio_16_sd" = "climate",
  "wc2.1_30s_bio_17" = "climate",
  "wc2.1_30s_bio_17_sd" = "climate",
  "wc2.1_30s_bio_18" = "climate",
  "wc2.1_30s_bio_18_sd" = "climate",
  "wc2.1_30s_bio_19" = "climate",
  "wc2.1_30s_bio_19_sd" = "climate"
)

# Convert the mapping to a dataframe
variable_types_df <- data.frame(
  Variable = names(variable_types),
  Source = variable_types,
  stringsAsFactors = FALSE
)

# Merge the variable counts dataframe with the variable types dataframe
variable_counts_df <- merge(variable_counts_df, variable_types_df, by = "
    Variable", all.x = TRUE)

# Sort the dataframe by the number of counts in descending order
variable_counts_df <- variable_counts_df[order(-variable_counts_df$Count),
    ]

# Print the sorted dataframe with the added Type column
print(variable_counts_df)

# Print the chosen variables for each taxon
results_df$Variables
```

Listing 13: Variable analysis

```
# This script extracts raster values for each polygon in a shapefile,
    calculates mean and standard deviation
```

```r
# of the extracted values, merges them with landcover classes, and creates a LaTeX table of the results.

# Load necessary libraries
library(terra)
library(sf)
library(dplyr)
library(exactextractr)
library(knitr)

# Load the raster
raster_file <- "data/results/final_biodiversity_map.tif"
r <- rast(raster_file)

# Load the shapefile
shp_file <- "data/study_area/VegAug1_KILI_SES.shp"
shp <- st_read(shp_file)

# Ensure CRS match
shp <- st_transform(shp, crs(r))

# Extract raster values for each polygon
extracted_values <- exact_extract(r, shp, c("mean", "stdev"))

# Combine extracted values with shapefile attributes
shp$mean_SR <- extracted_values$mean
shp$sd_SR <- extracted_values$stdev

# Landcover classes lookup table
landcover_classes <- data.frame(
  grid_code = c(0, 1, 2, 3, 4, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19),
  landcover_class = c("snow, glacier", "Agriculture (MAI)", "savannah (SAV)", "swamp",
                      "overgrown/clearing", "forest plantation", "riverine",
                      "upper montane Erica excelsa forest (FPO Podocarpus disturbed)",
                      "subalpine Erica trimera bushland (FED, including small patches of FER)",
                      "FPO", "subalpine tussock grassland", "HOM", "HEL", "FOC",
                      "bare rock", "FLM", "COF")
)

# Merge the lookup table with the result dataframe
shp <- shp %>%
  left_join(landcover_classes, by = "grid_code")

# Calculate mean SR and SD for each landcover class
summary_table <- shp %>%
  group_by(landcover_class) %>%
  summarize(mean_SR = mean(mean_SR, na.rm = TRUE),
            sd_SR = mean(sd_SR, na.rm = TRUE)) %>%
  arrange(mean_SR)

summary_table2 <- as.data.frame(summary_table)[, c(1, 2)]

```

```
# Create LaTeX table
latex_table <- summary_table2 %>%
  kable("latex", booktabs = TRUE, col.names = c("Landcover Class", "Mean SR
    "))

# Print LaTeX table code
print(latex_table)

# Optionally, export LaTeX table to a file
cat(latex_table, file = "mean_sd_sr_table.tex")

# Plot mean_SR from two summary tables (if needed for comparison)
#merged <- merge(x = summary_table, y = summary_table2, by = "
    landcover_class")
#plot(x = merged$mean_SR.x, y = merged$mean_SR.y)
```

Listing 14: Calculation and Creation of Landcover table