



Deliverable 3.1 (D3.1)

Application software implementing remote sensing, distributional down- and biodiversity up-scaling

M30

Project acronym: EU BON
 Project name: EU BON: Building the European Biodiversity Observation Network
 Call: ENV.2012.6.2-2
 Grant agreement: 308454
 Project duration: 01/12/2012 – 31/05/2017 (54 months)
 Co-ordinator: MfN, Museum für Naturkunde - Leibniz Institute for Research on
 Evolution and Biodiversity, Germany

Delivery date from Annex I: M30 (May 2015)

Actual delivery date: M31 (June 2015)

Lead beneficiary: University of Leeds

Authors: Gavish, Y. (UniLeeds, University of Leeds, School of Biology, UK)
 Marsh, C.J. (UniLeeds, University of Leeds, School of Biology, UK)
 Kunin, W.E. (UniLeeds, University of Leeds, School of Biology, UK)
 Garzon-Lopez, C.X. (FEM, Fondazione Edmund Mach, Italy)
 Rocchini, D. (FEM, Fondazione Edmund Mach, Italy)
 Schirrmeister, F. (UFZ, Helmholtz Centre for Environmental Research, Germany)
 Pe'er, G. (UFZ, Helmholtz Centre for Environmental Research, Germany)
 Henle, K. (UFZ, Helmholtz Centre for Environmental Research, Germany)

This project is supported by funding from the specific programme 'Cooperation', theme 'Environment (including Climate Change)' under the 7th Research Framework Programme of the European Union

Dissemination Level

PU	Public	✓
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

This project has received funding from the European Union's Seventh Programme for research, technological development and demonstration under grant agreement No 308454.

All intellectual property rights are owned by the EU BON consortium members and protected by the applicable laws. Except where otherwise specified, all document contents are: “© EU BON project“. This document is published in open access and distributed under the terms of the Creative Commons Attribution License 3.0 (CC-BY), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.



Executive Summary

Introduction

This deliverable provides background information and implementation code for the analytic tools developed in *task 3.1* and *task 3.2* of EU BON WP3. After a general introduction to WP3 and its tasks, the deliverable first clarifies the need for each tool we developed. Then we describe the conceptual rationale of the tool. This is followed by a more detailed explanation of the content of each tool along with a clear description of the additional information provided in the appendices. Executable files for each tool, where appropriate, can be found here: <http://www.eubon.eu/documents/1/>.

Progress towards objectives

The main objective of WP3 is to develop and improve tools and methods for biodiversity data analyses, covering habitat classification tools (*task 3.1*), scaling issues (*task 3.2*), enhanced species distribution models (*task 3.3*) and data mining (*task 3.4*), with the first two being the focus of this deliverable. The second main objective of WP3 is to make these tools more accessible to both professionals and the general public. To meet the first objective we developed a diverse set of tools for each task, including *Hierarchical RandomForest*, *Fourier Transforms* and *Land Surface Temperature* tools for *task 3.1* and *UpScaling Diversity*, *DownScaling Occupancy* and *Virtual Ecologist* tools for *task 3.2*. For the second objective, the tools reported here have been developed in open access platforms such as *R* and *GrassGIS*, and much emphasis was given to providing clear supporting documentation, either as help files, vignettes or scientific publications.

Achievements and current status

In *task 3.1* (Advanced tools for interpreting satellite or aerial imagery using environmental datasets and machine learning methods) we aimed to develop enhanced tools that use fine-scale multi-spectral remotely sensed imagery along with other earth observation data to map habitats, land-covers or land-uses and to monitor their change. These tools include:

- **Hierarchical RandomForest** – A habitat classification tool that incorporate the pre-defined hierarchical structure of a habitat classification scheme into a machine-learning classification algorithm based on randomForest (*R* package).
- **Fourier Transforms** – Detecting multi-temporal landscape fragmentation by remote sensing (*GrassGIS* and *R* commands).
- **Land Surface Temperature (LST)** – A method to construct high-resolution LST maps using remote sensing data (*GrassGIS* commands).

In *task 3.2* (Techniques for up-scaling and downscaling biotic datasets) we aimed to develop tools that use readily available data sources at certain scales to predict hard-to-measure properties of biotic datasets at other scales. These tools include:

- **UpScaling Diversity** – Tools that use a limited number of fine-resolution samples of assemblages to predict the number of species in a larger spatial extent (*R* package).
- **DownScaling Occupancy** – Tools that uses widely available atlas data at coarse resolution to predict the proportion of occupied cells and the area of occupancy at much finer resolutions (*R* package).
- **Virtual Ecologist** – a modelling framework which samples from the outcomes of individual-based models to assess the capacity of alternative sampling designs to downscale from species distribution and abundances to local patterns of (observed) occurrence and abundances (*R* package).

Future developments

Future development/use of the tools could involve

1. Dissemination of the tools for implementation in WP4, testing in WP5 and dissemination to the general public in WP8.
2. Link the tools to EU BON portal.
3. Submission of the *R* packages to *R* CRAN and their continuous maintenance.
4. Publication of papers in R-oriented journals (e.g. Journal of Statistical Software- <http://www.jstatsoft.org/>).
5. Codification of new methods and/or additional supplementary methods into existing packages and software.
6. Publication of high impact case-study manuscripts.

Table of Contents

1. GENERAL INTRODUCTION.....	7
2. HIERARCHICAL RANDOM FOREST	9
2.1. Introduction.....	10
2.2. The Hierarchical randomForest Approach.....	11
2.3. Main features of the ‘HieRanFor’ package.....	15
2.4. ‘HieRanFor’ package - Tutorial.....	17
3. FOURIER TRANSFORMS	29
3.1. Introduction.....	29
3.2. Approach.....	29
3.3. Main features	30
3.4. Tutorial and examples.....	30
4. LAND SURFACE TEMPERATURE	33
4.1. Introduction.....	33
4.2. Approach.....	33
4.3. Main features	33
4.4. Tutorial and examples.....	34
5. UPSCALING DIVERSITY.....	35
5.1. Introduction.....	35
5.2. Simplified Maximum Entropy	36
5.3. True and Sampled SOD	38
5.4. Total-Species (T-S) Curve	40
5.5. Nested SAC models	41
5.6. Main features of the ‘UpScaling’ package.....	45
5.7. ‘UpScaling’ package – Tutorial	47
6. DOWNSCALING OCCUPANCY.....	65
6.1. Introduction.....	65
6.2. The ‘downscale’ package structure	65
6.3. Installing and using the ‘downscale’ package.....	68
7. VIRTUAL ECOLOGIST	76
7.1. Introduction.....	76
7.2. Model Concepts and structure.....	77
7.2.1. Purpose.....	77
7.2.2. Elements, entities, scales and variables.....	78
7.2.3. Processes	78

7.3. Design concepts	79
7.4. Implementation	79
7.5. Outputs	81
7.6. Additional notes	81
8. REFERENCES.....	82
9. ACKNOWLEDGMENT	85
10. APPENDICES	86

1. GENERAL INTRODUCTION

Effective management of natural systems at various scales and levels of organization requires a detailed understanding of the states and trends of populations, species, communities and ecosystems. Such understanding cannot be achieved without accumulation and mobilization of biodiversity data. Thus, one of the main objectives of EU BON is to integrate and harmonize various data sources and increase their accessibility to various stakeholders (Hoffmann et al. 2014). However, an effective management scheme cannot rely on raw data alone. In fact, the three main pillars of EU BON -- data sources & infrastructure, science & application, and policy & dialogue -- follow the transformation of various types of raw data through statistical analysis to a clearer understanding of states or trends that can then be translated to policy. The role of WP3 in this general framework is in the science & application pillar- our aim is to develop new analytical tools and/or improve existing analytical tools for analyses of biodiversity data. The tools developed in WP3 will undergo testing and validation in WP5 and will be used in WP4 to identify biodiversity status and trends. Furthermore, the tools will be disseminated to the wider scientific community, policy makers, and other stakeholders by WP8. Finally, the integration of the tools within EU BON's biodiversity portal will be explored by WP2.

WP3 covers four of the many aspects of biodiversity, including habitat classification tools (*task 3.1*), scaling issues (*task 3.2*), enhanced species distribution models (*task 3.3*) and data mining (*task 3.4*). In *task 3.1* we aim to develop enhanced tools that use fine-scale multi-spectral remotely sensed imagery along with other earth observation data to map habitats, land-covers or land-uses and to monitor their change. In *task 3.2* we aim to develop tools that use readily available data sources at certain scales to predict hard-to-measure properties of biotic datasets such as occupancy or richness at other scales. In *task 3.3* we aim to develop enhanced methods for species distribution models for both data-limited and data-rich species. Finally, in *task 3.4* we aim to develop a tool to semi-automatically extract user defined information from published data. *Task 3.3* and *task 3.4* will be reported in deliverables D3.2 (M40) and D3.3 (M48), respectively, although considerable information on the various tools developed in these tasks can be found in EU BON's progress reports. In this deliverable we will focus on the six different tools developed in *task 3.1* and *task 3.2*, led by *UnivLeeds*, *FEM* and *UFZ*. However, we note here that many additional partners (*EBCC*, *EURAC*, *INPA*, *MRAC*, *NBIC*, *NHM*, *SGN*, *UCPH*, and *Vizzuality*) contributed considerably to tool development through their suggestions and critiques during web conferences and annual meetings.

In *task 3.1* we have developed several tools that use multi-spectral remotely sensed data and other earth observation data to map habitats (hierarchical randomForest), to monitor the change in the distribution of habitat (Fourier transforms) or to provide fine scale information on climatic conditions that can increase the accuracy of species distribution models (Land Surface Temperature). These tools are more thoroughly described in sections 2-4 of this deliverable. In *section 2* we focus on the new R package 'HieRanFor' developed by *UnivLeeds*. This R package allows the application of the

hierarchical randomForest algorithm – a new machine-learning-based habitat classification tool that accounts for the pre-defined hierarchical structure of habitat classification schemes. In *section 3* we focus on the *Fourier transforms* methods developed by *FEM*, covered in Rocchini et al. (2013) and implemented in GRASSGIS 7.0 (Neteler et al. 2012)– a free and open-source geographical information system platform. This method detects change in landscape cover in a continuous manner, thereby avoiding the loss of information associated with transforming continuous information (such as reflectance) to categories (such as habitat classes). Finally, in *section 4* we focus on the *Land Surface Temperature* (LST) method developed by *FEM*, covered in Metz et al. (2014) and implemented in GRASSGIS. This method can reconstruct surface temperature from remotely sensed images over wide spatial extents at an unprecedented resolution of 250 m and at a monthly return rates.

The general unifying concept of the various tools developed in *task 3.2* is to use readily available data sources at certain scales to predict hard-to-measure properties of biotic datasets at other scales. For some aspects of biodiversity (e.g. number of species) the available information is in fine resolution while the management scale is usually at coarser resolutions. For other biodiversity aspects (e.g., real or modelled species distributions), accurate data is usually available at coarser resolution than the management scale. Thus in *task 3.2* we focused on 3 different tools that involve upscaling or downscaling of biodiversity data. In *section 5* we focus on the new R package ‘UpScaling’ developed by *UnivLeeds*. This R package brings together several advanced models that predict the number of species in a large extent based on information from scattered finer resolution samples from within the focal area; these include one new and three published models. In *section 6* we focus on another new R package ‘downscale’ developed by *UnivLeeds*. This R package contains ten different published models that predict the proportion of occupied cells at fine resolution from coarse scale atlas data on species’ distributions, as well as an ensemble method. The package also incorporates advanced tools for “upgraining” data to coarser resolution while accounting for changing spatial extents. Finally, in *section 7* we focus on the ‘*Virtual Ecologist*’ tool developed by *UFZ*. This tool provides a modelling framework, which samples from the outcomes of individual-based models to downscale from species distribution and abundances, as well as connectivity, to local patterns of (observed) occurrence and abundance, while accounting for various aspects of sampling, including collector expertise and sampling bias.

Applying the new and published tools covered in *sections 2-7* from scratch is not straightforward without considerable experience and expertise. Some of the tools involve complex mathematical models that are beyond the scope of most ecologists, while others require consecutive steps and advanced programming abilities. In fact, the ecological literature is full of complex analytical tools that may contribute considerably to our understanding of the states and trends of biodiversity. However, many of these tools are only used by a handful of expert that understand them well enough to apply them from scratch. Therefore, one of the main objectives of WP3 was to develop these tools in a format that will increase their accessibility to both professionals and the general

public. Therefore, the tools covered in this deliverable were implemented where possible in open source and widely used platform such as R (R Development Core Team 2011) and GrassGIS (Neteler et al. 2012). In addition, much emphasis has been given to providing clear supporting documentation, either as help files, vignettes or scientific publications.

For the three R packages (sections 2, 5 and 6) this report starts with a general introduction that explains the need for each tool and the main conceptual framework on which the tool is based. This is followed by a more through description of the main functions of the package along with an installation guide. Finally for each R package we include a tutorial, to supplement the help files or vignettes that can be downloaded at <http://www.eubon.eu/documents/1/> (see **Table 1.1**). Sections 3 and 4 (the two GrassGIS packages) follow a similar structure, starting with a general introduction and presentation of the conceptual approach, followed by the main features of the tools and a tutorial. We note that for these two tools a more detailed report can be found in relevant published manuscripts (Rocchini et al. 2013 for the Fourier transforms, Metz et al. 2014 for the land surface temperature). For the virtual ecologist tool (section 7), the introduction and presentation of the concept is followed by model description based on the ODD (Overview, Design concepts, Details) protocol (Grimm et al. 2006, Grimm et al. 2010). We provide additional information on the design of the R package, its implementation and output package, along with presentation of the three main functions.

Table 1.1: List of tools developed in each task, their contact person and relevant supporting files that can be accessed at <http://www.eubon.eu/documents/1/>.

	Tool	Contact person	Additional information	
			File name	Description
Task 3.1	Hierarchical randomForest	Yoni Gavish gavishyoni@gmail.com	SI2.1-HieRanFor.pdf	Help files
			HieRanFor_1.0.tar.gz	Installation file from within the R console
			HieRanFor_1.0.zip	Zipped installation file
	Fourier transforms	Duccio Rocchini ducciorocchini@gmail.com	http://gis.cri.fmach.it/rocchini/	
	Land Surface Temperature	Duccio Rocchini ducciorocchini@gmail.com	http://gis.cri.fmach.it/rocchini/	
Task 3.2	Upscaling diversity	Yoni Gavish gavishyoni@gmail.com	SI5.1-UpScaling.pdf	Help files
			SI5.2-Azaele_2015_PCF.pdf	Mathematica code - pair correlation function
			UpScaling_1.0.tar.gz	Installation file from within the R console
			UpScaling_1.0.zip	Zipped installation file
	Downscaling occupancy	Charles J Marsh c.marsh@leeds.ac.uk	SI6.1-downscale.pdf	Help files
			SI6.2-Downscaling_tutorial.pdf	Downscaling species occupancy- an introduction and tutorial
			SI6.3-Upgraining_tutorial.pdf	Upgraining atlas data for downscaling + threshold selection using upgrain.threshold
			downscale_1.0.tar.gz	Installation file from within the R console
			downscale_1.0.zip	Zipped installation file
	Virtual ecologist	Guy Pe'er guy.peer@ufz.de	https://www.ufz.de/index.php?en=15885	

2. HIERARCHICAL RANDOM FOREST

2.1. Introduction

High quality habitat maps are required for effective management, land-cover change detection and nature conservation. In many cases, the habitat categories are pre-defined by national or international classification schemes, to allow a common language of communication between scientists, management agents and policy makers. Most habitat classification schemes (e.g., EUNIS) have a hierarchical, tree-like structure, in which fine categories are nested within more general categories. For example, the British National Vegetation Classification (NVC) consists of twelve major categories (e.g. woodland), each containing 12-42 communities (e.g. *Fagus sylvatica* – *Mercurialis perennis* woodland) for a total of 286 communities, while communities can be further classified into sub-communities. The EUNIS classification scheme is of similar tree-like structure, only with up to eight hierarchical levels.

The hierarchical structure of habitat classification schemes plays an important role in knowledge-based classification (classification methods that incorporate expert knowledge). Most knowledge - based classification methods follow a top-down approach, in which experts first provide rules that separate general categories from one another, and then move down the hierarchy while providing more specific rules for finer categories (e.g., Lucas et al. 2007, Lucas et al. 2011). The main advantage of rule-based classifications is in the more process-based understanding they provide and in the ability to apply the model without ground truth data. However, expert knowledge is not always available for all systems and fine-tuning the rules is time consuming. In addition, rules are not easily transferable from one system to another, even if the two systems share most of their habitats. Finally, the rules provide a static model that cannot automatically update when new input data becomes available or old data becomes unavailable.

Alternatively, in recent years there is a growing usage of machine-learning algorithms (e.g., Support Vector Machine, Random-Forest) for habitat classification (e.g., Bradter et al. 2011). However, most machine learning classification methods follow a flat classification approach (sensu Silla and Freitas 2011). In flat classification (**Figure 2.1A**), all terminal nodes are classified simultaneously in a ‘one-against-all’ approach which may cause reduced accuracy when the number of nodes is high. In addition, no usage is made of the pre-defined habitat structure even though this information is available for the ground-truth data. Furthermore, variable importance provide information on the contribution of the variable to the overall accuracy, but not on the actual classes that the variable can tease apart. Finally, accuracy assessment usually treats all misclassification the same, ignoring the distance between categories along the hierarchical tree structure. In most applications, misclassifying a *Fagus sylvatica*- *Mercurialis perennis* woodland as a *Fagus sylvatica*-*rubus fruticosus* woodland would be a less serious mistake than misclassifying it as a *Pinus sylvestris* woodland, or indeed as a grassland or saltmarsh.

Therefore, in task 3.1 we developed a novel machine-learning based hierarchical classification method (**Figure 2.1B**), which accounts for the pre-defined hierarchical structure of the habitat classification scheme, while avoiding the need of expert knowledge. We are aware of only a few published manuscripts that focused on hierarchical, machine-learning based classification methods in the remote-sensing literature. However these have either used a very simple local classification algorithm (Thoonen et al. 2013) or did not follow an external pre-defined habitat classification scheme (Chen and Breiman 2004, Melgani and Bruzzone 2004). None of the above provided an executable file that allows others to explore or apply their tool.

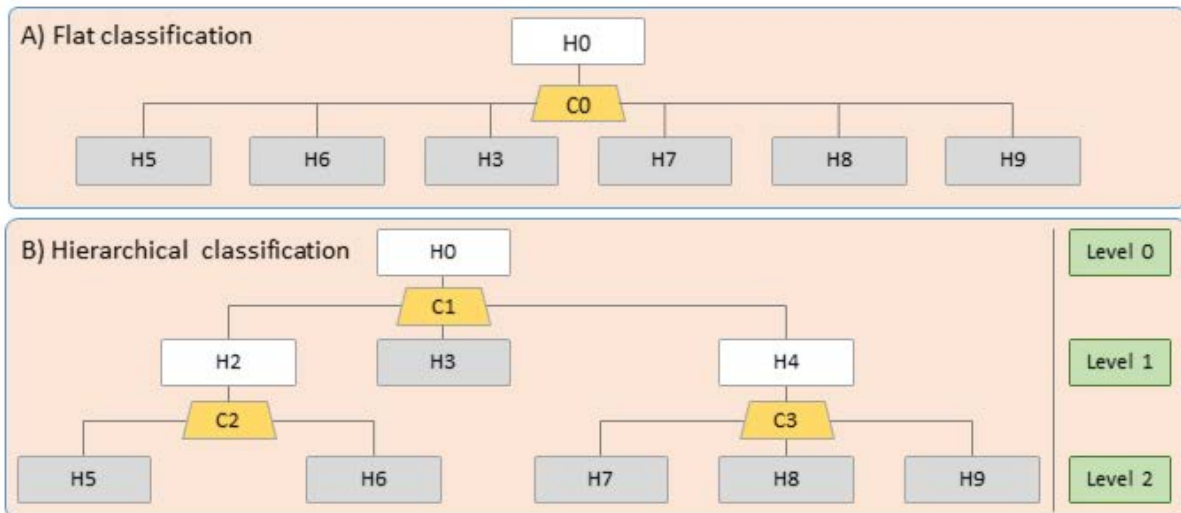


Figure 2.1: (A) In flat classification, all habitats (terminal nodes, grey filled rectangles) are classified in a single local randomForest model, ignoring the class hierarchy. (B) In hierarchical classification, a flat randomForest is used as a local classifier (yellow trapezoid) along the class hierarchy (in this case 2 levels). Local classifiers are built for every internal node (white filled rectangles) that has more than one child node. Training cases for H5 are used along with those of H6 to represent H2 in local classifier C1, in addition to their role in local classifier C2.

2.2. The Hierarchical randomForest Approach

RandomForest (Breiman 2001) is a machine learning method designed to produce accurate classification of multiple cases to predefined classes (see appendix A2.1 for definitions). Each application of randomForest takes as input a set of training cases (e.g., grid-cells containing certain habitat types), representing a predefined and finite set of classes, along with relevant explanatory variables. The algorithm then ‘learns’ the rules by which the explanatory variables can distinguish between the different classes (see **Figure 2.2A,B** for more details). The learning procedure is based on fitting a ‘forest’ of classification trees (**Figure 2.2B**), with each tree being based on a different subset of the original training cases (“in-bag” data) and each branching in any classification tree using only a small and random subset of the available list of explanatory variable (**Figure 2.2A**). Therefore, each classification tree is unique, and can provide independent prediction for each case that was not included in tree growing (i.e., “out-of-bag” data – OOB). Thus the output of randomForest is a list containing the predicted class for each OOB training case according to each tree (known as a vote). The results are then usually translated into a soft classification output, by estimating for each case, the

proportion of OOB votes that assigned the case to any of the classes. The reliance on OOB votes (i.e., only trees in which the case was not in the in-bag subset) results with a method robust to over-fitting. In addition, as accuracy is always based on OOB votes, there is no need for an external validation set.

The hierarchical RandomForest (HRF) takes similar input as flat randomForest along with additional information on the class hierarchy (**Figure 2.2C**). Then, randomForest is used as the local classifier in every internal node that has at least two children nodes. The training set for each randomForest consist of the training cases of all descendent of the local classifier parent node. For example, in **Figure 2.2C**, training cases representing the H5 and H6 habitat classes are used in classifier C1 to represent habitat H2. The same cases are also used in local classifier C2 to represent habitats H5 and H6. However, the H5 and H6 training cells are not used in classifier C3 which aims to separate between habitats H7, H8 and H9.

After training all the local randomForests, a specially designed predict function runs each case in each local classifier, while ensuring the usage of OOB trees. As a case is never predicted by a classification tree in which it was within the in-bag subset, the overall HRF retains the robustness of randomForest to over-fitting. Similarly, as accuracy is always based on OOB votes, there is no need for an external validation set. None-the-less, similar to flat randomForest, it is possible to run additional cases through the HRF model for map production or for additional assessment of accuracy and performance if the ‘true’ class is known.

Both flat and hierarchical randomForest produce for each model the proportion of OOB votes for a focal case. To assess the performance (accuracy) of the model and to produce maps, a single class needs to be selected according to the proportion of OOB votes (i.e., translating the ‘soft’ probabilities to a ‘crisp’ class). For flat randomForest, the flat majority rule is usually applied by selecting for a focal case the class that received the highest proportion of OOB votes. For HRF, there are three different majority rule options – stepwise majority, multiplicative majority and multiplicative permutations (**Figure 2.3**).

In stepwise majority rule, the flat majority rule is applied in each local classifier, and starting from the tree root, the selected class is followed until a terminal node is reached. In multiplicative majority rule, the proportion of OOB votes are multiplied along every path from the tree root until each terminal node, and the flat majority rule is applied on the multiplicative proportions. Interestingly, the multiplicative proportion of OOB votes are comparable to the proportion of OOB votes generated by a flat classification. The multiplicative permutation is based on the multiplicative proportion of OOB votes. However, instead of applying a majority rule, the method randomly selects a terminal node based on the multiplicative probabilities. The user defines the number of permutations and accuracy is assessed separately for each permutation. When most cases are classified to the same class in various classification trees (i.e., when the proportion of votes for one category is close to 1), the mean accuracy over all permutation should be similar to that achieved under the multiplicative

majority method. In the example in **Figure 2.3**, each permutation will choose a class as random draw with probabilities equal to the values in red.

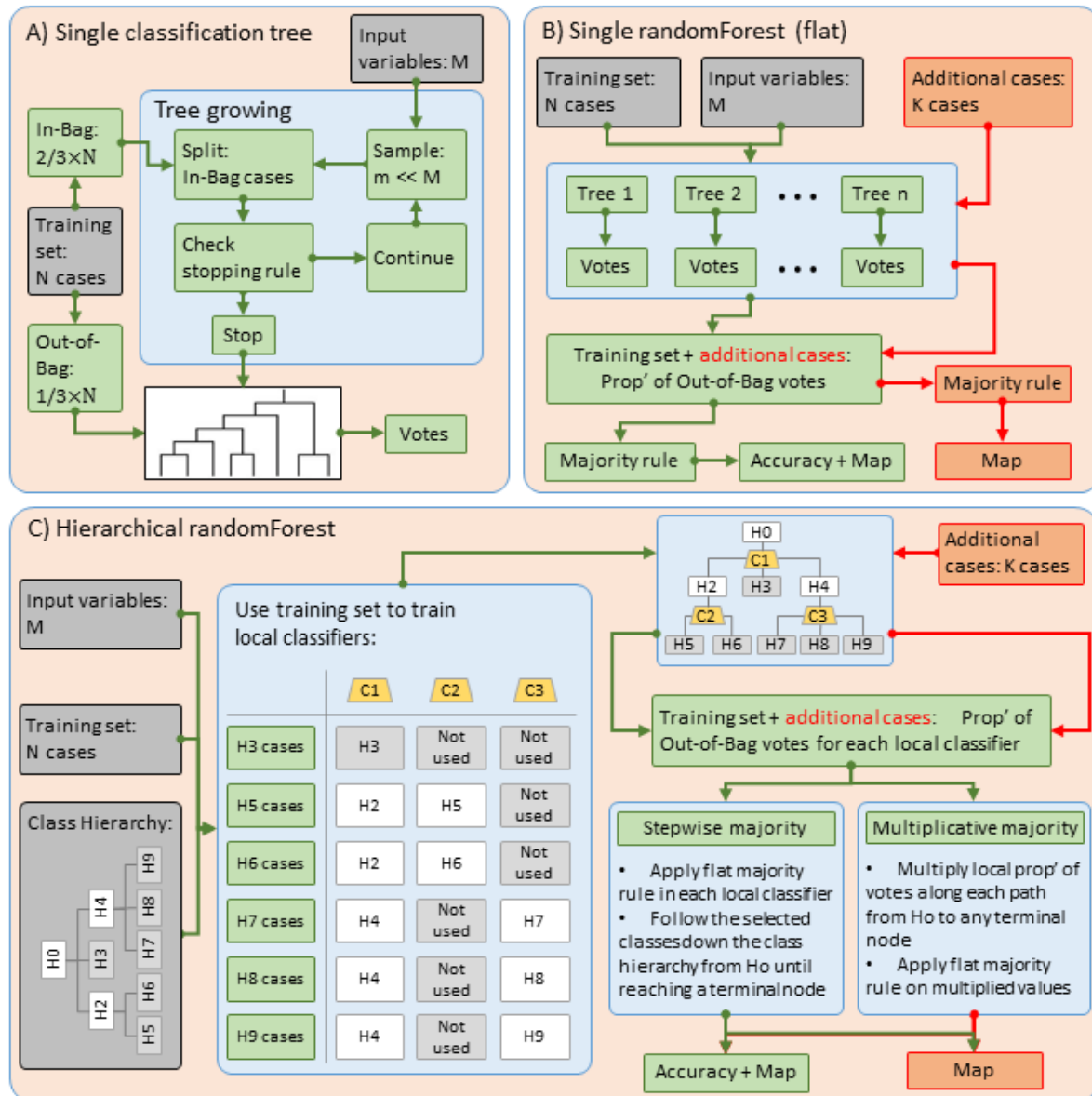


Figure 2.2: Flow chart for creating: (A) a single classification tree, (B) a randomForest model, and (C) a hierarchical randomForest model. (A) A single classification tree takes as input a training set and explanatory variables. The training set is then divided to in-bag cases that will be used to grow the tree and out-of-bag (OOB) cases that will be used to assess accuracy. The tree is grown by selecting at each branching a random and small number of input variables and using them to split the in-bag cases. This is continued until a stopping rule is achieved. The tree provides a single vote to all OOB cases. (B) A randomForest model applies multiple classification trees, each with a unique division to in-bag and OOB, and a unique and random draw of input variables at each branching node. The proportion of votes for all training cases is then predicted using only the OOB trees for any case. Similarly, additional cases can be predicted using a randomForest model. A flat majority rule is then used to assign the most probable class for each case using the proportion of OOB votes. The crisp class can be used for accuracy assessment and mapping. (C) The hierarchical randomForest fits a randomForest model as a local classifier for each node that have more than two children nodes in the class hierarchy, using relevant training data. Similar to flat randomForest, each local classifier is used to predict the proportion of OOB votes for all training (and external) cases. Unlike flat randomForest, three methods of translating the proportion of votes into a crisp classification can be employed for accuracy assessment and mapping, with the multiplicative permutations (not shown) being based on the multiplicative proportions.

Although never formally tested, we hypothesise that the stepwise option will be more accurate if the local classifier close to the tree root provides high accuracy. On the other hand, the multiplicative approach may be better if uncertainty is equally distributed along the class hierarchy. However, if the tree structure is highly unbalanced in terms of depth (i.e., some terminal nodes are located deep down the class hierarchy and some very close to the tree root) or number of siblings, the multiplicative majority rule may be biased towards classes closer to the tree root or with fewer sibling nodes. The multiplicative permutation is still experimental and its usage should be more thoroughly explored. Preliminary analyses revealed that the mean permutation based accuracy is usually lower than both the stepwise and multiplicative methods.

For both flat and hierarchical models, once the proportion of votes are translated to a single ‘crisp’ class, various performance and accuracy measures may be estimated. Usually, flat accuracy indices such as accuracy (proportion of cases correctly classified) or Cohen’s kappa are estimated. However, these indices do not account for the class hierarchy and give similar weights to all misclassifications. For example if the focal case in **Figure 2.3** is of class H6, then its misclassification as H5 by the multiplicative majority rule affect the overall performance of the model similarly to misclassifying it as H7. In the ‘*HieRanFor*’ package, performance can also be assessed using Kiritchenko et al. (2005) hierarchical precision (**hP**), hierarchical recall (**hR**) and hierarchical F (**hF**) measures that account for the class hierarchy. We developed new formulations for **hR**, **hP** and **hF** that can be calculated directly from the confusion matrix (see appendix A2.2). In addition, we developed similar hierarchical indices for single terminal node (**hP_k**, **hR_j**, and **hF_{j=k}**), that converge to the precision and recall accuracies of flat accuracy measures when the hierarchy is ignored (**Table A2.2**).

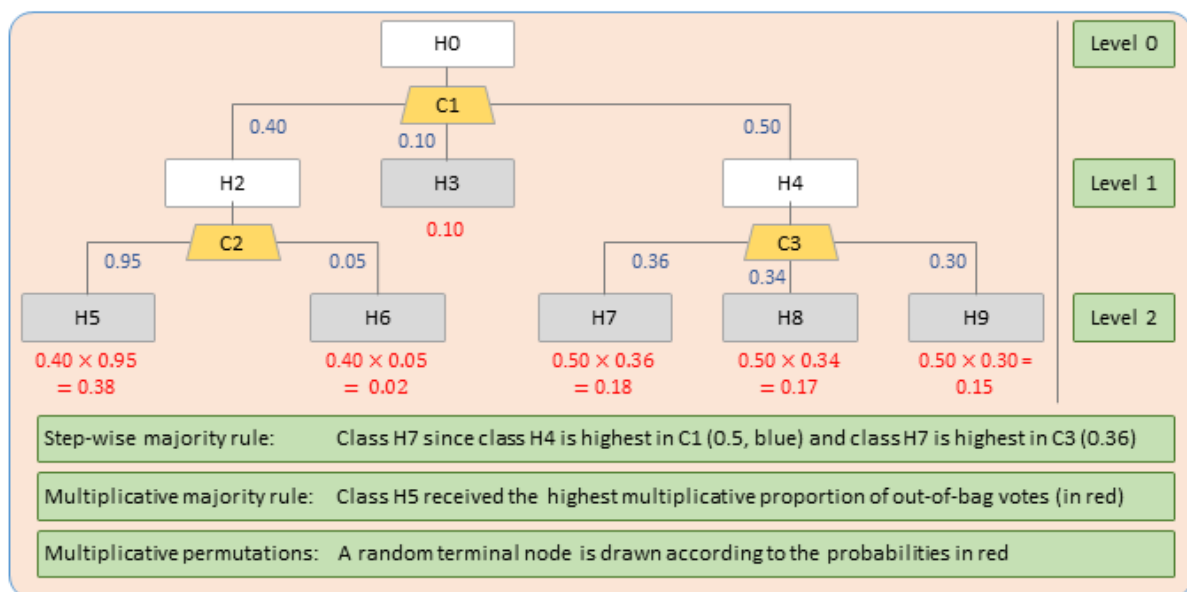


Figure 2.3: The proportion of OOB (in blue) votes that a hypothetical case received in each local classifier, and the three options to select a single, most-probable class. Note that for this hypothetical case the stepwise majority rule and multiplicative majority rule result with a different crisp class.

2.3. Main features of the ‘HieRanFor’ package

To increase the accessibility to the general public the HRF method was codified as an R package. The package is currently available as a beta version in R-Forge and will be soon submitted to CRAN. The package allows the users full functionality and include the following main function:

1. `RunHRF`

The main function of the package that identified the class hierarchy from the input training set and runs `randomForest` as the local classifier at each internal node that has more than one child node. The function returns an object of class ‘HRF’ that contains all the local `randomForest` objects along with additional information on the hierarchical structure.

2. `plot.HRF`

An S3 plot function for an object of class ‘HRF’ that takes as input an object of class ‘HRF’, and plots the structure of the class hierarchy. See **Figure 2.4** for an example.

3. `ImportanceHie`

This function extracts the importance value of each explanatory variable for each local classifier. Variable importance is estimated by the `randomForest` package as the mean decrease in accuracy due to permutation of variable’s values.

4. `PlotImportanceHie`

A plotting function for the importance values generated by `ImportanceHie`. Plots the parent node and variable name as the two axes with tile colours or bubble sizes scaled according to the importance values. See **Figures 2.5** and **2.6** for examples.

5. `predict.HRF`

An S3 predict function for an object of class ‘HRF’ that predicts the proportion of OOB votes for each case of the training data and/or of a new dataset in each local classifier. Note that unlike `PredictNewHRF` (see below), this function does not estimate the multiplicative proportion of votes and translate the proportion of votes to a crisp classification.

6. `PerformanceHRF`

This function predicts the proportion of votes for each case in each local classifier and then estimates various flat and hierarchical performance measures. The performance measures are based on translating the proportion of OOB votes to a crisp class (selecting a single class). The function offers three different method for translating the soft proportion of votes to a crisp class – stepwise majority rule, multiplicative majority rule and multiplicative permutation. See **Figure 2.3** and above.

7. `PredictNewHRF`

This function predicts the crisp class of new data (data not included in the training set) using any of the three crisp rules described above. The crisp class generated by this function may allow users to create habitat maps for larger extents than those covered by the training set.

`PerformanceNewHRF` Similar to the `PredictNewHRF`, only with the additional computation of various performance measures if the ‘true’ class is known for the new data (i.e., an external validation set).

8. `PerformanceFlatRF`

This function takes as input an object of class ‘HRF’, identifies the observed terminal node for each case of the original training data, and runs a regular flat classification on all terminal nodes. The function then applies two different methods for selecting a single terminal node (note that the stepwise majority rule cannot be applied in a flat classification). Finally, the performance is explored in relation to the observed class using flat and hierarchical performance measures.

In addition to these main functions, the package also contains a few additional functions, including a function that creates random class hierarchies and data (`RandomHRF`), a function that estimates the hierarchical F measures (`HieFMeasure`), a function that takes the proportion of OOB votes at each local classifier along with information on the class hierarchy and returns the multiplicative proportion of votes (`GetMultPropVotes`), a function that corrects a small bug in the `tuneRF` function of `randomForest` (`TuneRF2`), and a utility function that joins the factor levels of two vectors (`JoinLevels`). Finally, the package contains a modification of an R dataset which was used as an example in the package documentations (`OliveOilHie`).

The `HieRanFor` package relies on the ‘`randomForest`’ package that implement the original code of (Breiman 2001). Although the `randomForest` package is still widely used in applying `randomForest` in R, alternatives are available. Among the various alternative, the implementation of `randomForest` in the “`party`” package has the advantage of basing the forest on conditional inference trees (Hothorn et al. 2006, Strobl et al. 2008) which produce unbiased variable importance values. We chose the `randomForest` package for two main reasons. First, as conditional inference trees require permutation, they require considerably longer computational time, which is even more demanding when applying multiple local models along a pre-defined class hierarchy. Second, the `randomForest` package allows saving for each tree the identity of the in-bag and OOB cases, which is crucial for the `predict` function in the `HieRanFor` package.

A fully functional beta version of the `HieRanFor` package has been submitted to R-FORGE (<https://r-forge.r-project.org/>). To view the beta version, search for the project ‘*hie-ran-forest*’ in the R-FORGE site and navigate to the SCM repository. In the near future, the package will be submitted to CRAN and could be installed directly from the R console. For now, the package can be installed from the *HieRanFor_1.0.tar.gz* or *HieRanFor_1.0.zip* files (downloadable from <http://www.eubon.eu/documents/1/>). For further installation assistance and additional details contact Yoni Gavish (gavishyoni@gmail.com).

2.4. ‘HieRanFor’ package - Tutorial

As required by all R package, the ‘HieRanFor’ package include help files on each function along with executable examples. These help files include a general description of the aim of the function and on each of the input terms it requires. In addition, the help files contains detailed information on the output that each function generates and additional important details. All these help files are summarized in the file *SI2.1- HieRanFor.pdf* (downloadable from <http://www.eubon.eu/documents/1/>).

Here we provide a simple tutorial that does not cover all functions, but rather explains the main flow of analysis. We will base the tutorial on the OliveOilHie dataset. Although this dataset does not include habitat information, the general structure of the data is similar. This is also the dataset on which many of the examples of the help files are based on. The data-set include 572 cases, arranged in a class hierarchy with 3 levels and 9 terminal nodes. Not all terminal nodes are at the deepest hierarchy level (see **Figure 2.4**) and a total of 6 local classifiers are required to run the HRF analysis. The data can be uploaded and explore in R using:

```
# Upload the OliveOilHie data-set to the global environment
data(OliveOilHie)
names(OliveOilHie)
[1] "case.ID"      "L1"           "L2"           "L3"           "pal m i t i c"
[6] "pal m i t o l e i c" "stea r i c"    "ol e i c"      "l i n o l e i c" "l i n o l e n i c"
[11] "arachi di c"  "ei cosenoi c"

dim(OliveOilHie)
[1] 572 12

levels(OliveOilHie$L3)
[1] "Apul i a. north" "Apul i a. south" "END. PATH"      "Li guri a. east" "Li guri a. west"
```

The main function that runs the HRF analysis is `RunHRF`. This function takes as input a data frame that must include a column for case IDs and information on the class hierarchy. All case IDs should be unique to allow separating in-bags from OOB. The hierarchy structure should be specified in a set of columns that contain the name of the habitat at each level for each case. In the OliveOilHie the three levels are specified in variables L1, L2 and L3. Note that L3 also contain a factor level named “END. PATH”- a special factor levels that can be specified by the user with `end.path.name`. This factor level signifies the end of a path from a tree root, if it ands before the maximal depth of the class tree. It is the only factor levels in the hierarchy that can appear in more than 1 level. In OliveOilHie, any terminal node ending in L2 (e.g., Umbria, **Figure 2.4**) should have “END. PATH” for its L3. Note also that the user must specify the name of `end.path.name` to avoid an error and must set `internal.end.path = TRUE` if the class hierarchy contains terminal nodes at lower levels than the maximal depth. We further note that habitat names should be

syntactically valid names, i.e. names that do not start with a number or a dot followed by a number (see `?make.names` in the R console or http://www.hep.by/gnu/r-patched/r-faq/R-FAQ_65.html).

```
#### running the hierarchical randomForest analysis with RunHRF
# return an error due to: internal.end.path = FALSE

hie.RF.00 <- RunHRF(train.data = OliveOilHie, case.ID = "case.ID", hie.levels = c(2:4),

                    mtry = "tuneRF2", internal.end.path = FALSE)

error: Some enteries within unique.path are end.path.name='END.PATH', despite internal.end.path==FALSE

# set internal.end.path = TRUE
# still return an error due to: end.path.name = "Wrong.Name"

hie.RF.00 <- RunHRF(train.data = OliveOilHie, case.ID = "case.ID", hie.levels = c(2:4),

                    mtry = "tuneRF2", internal.end.path = TRUE,
                    end.path.name = "Wrong.Name")
error: could not find end.path.name='Wrong.Name' in unique.path, despite internal.end.path==TRUE

# no error message
hie.RF.00 <- RunHRF(train.data = OliveOilHie, case.ID = "case.ID", hie.levels = c(2:4),
                    mtry = "tuneRF2", internal.end.path = TRUE, end.path.name = "END.PATH")

# plotting the class hierarchy
OOH.tree <- plot(x = hie.RF.00, text.size = 9, split.text = 10)

# adding title etc. using ggplot2
OOH.tree.plot <- OOH.tree$plot

# add a title
OOH.tree.plot <- OOH.tree.plot +
  ggtitle("Class Hierarchy: internal and terminal nodes + all local classifiers")
OOH.tree.plot <- OOH.tree.plot +
  theme(plot.title = element_text(lineheight=.8, face="bold", color="red"))
# edit the axis title
OOH.tree.plot <- OOH.tree.plot +
  theme(axis.title.y = element_text(size = 20,
                                    colour="blue"))
OOH.tree.plot
```

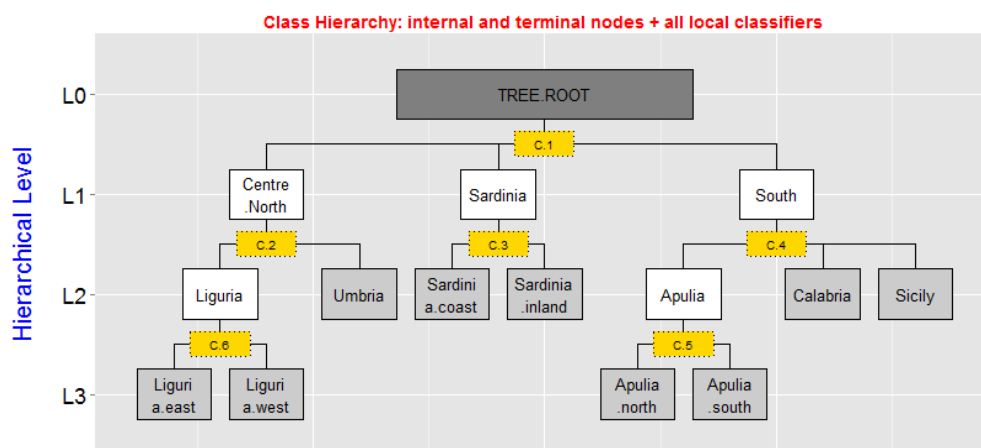


Figure 2.4: The class hierarchy of OliveOilHie as plotted using the `plot` function on a S3 object of class 'HRF'

The `RunHRF` function returns an object of class 'HRF' – a list that contains all the local randomForest classifiers, as well as information on the each local classifier, each node in the class hierarchy, each path from the tree root to any of the terminal nodes and a standardized, rearranged training data frame on which all other functions that require an object of class 'HRF' rely. The 'HRF' object is an S3 class object with well-defined methods for predict and plot. The predict option is explained below, while plot returns a figure with the class hierarchy (see **Figure 2.4**). The plot is based on the package ggplot2, and can be further edited by the user. The following codes can be used to explore and extract information from an object of class 'HRF':

```
# returns an object of class "HRF"
class(hie.RF.00)
[1] "HRF"

names(hie.RF.00)
[1] "hier.struc"      "train.data.ready" "case.ID"          "path.name"
[5] "hie.levels"      "exp.var"          "all.local.RF"     "order.local.RF"
[9] "call"

# each local classifier - the classifier ID, level and name of the parent class,
# classifier ID in which the parent class was classified, the total number of child nodes,
# and the number of terminal and internal child nodes.
lRF.info      <- hie.RF.00$hier.struc$lRF.info

# each node in the class hierarchy- the node's name, level and frequency in the data,
# the level and the name of the parent node, is the node terminal or internal, if internal
# whether it is a parent node in a local classifier + the name of this classifier, the Id
# of the classifier in which the node was classified and the number of
# levels above in which this classifier is located.
nodes.info    <- hie.RF.00$hier.struc$nodes.info

#each path from the tree root till any terminal node
unique.path   <- hie.RF.00$hier.struc$unique.path

# the re-arranged training data and the column numbers
train.data.ready <- hie.RF.00$train.data.ready
case.ID        <- hie.RF.00$case.ID
path.name      <- hie.RF.00$path.name
hie.levels     <- hie.RF.00$hie.levels
exp.var        <- hie.RF.00$exp.var

# All the information on the local classifiers is stored in the list all.local.RF.
all.local.RF   <- hie.RF.00$all.local.RF
# The order of classifier in this list is given in:
order.local.RF <- hie.RF.00$order.local.RF
# to access the classifier C.1 information:
local.lRF.list <- all.local.RF[[1]]
names(local.lRF.list)
[1] "local.lRF.info" "local.data"      "local.RF"

# the name of local classifier and all its info from lRF.info
local.lRF.info <- all.local.RF[[1]]$local.lRF.info
# A vector with all the cell.ID used as training data in this randomForest
local.data <- all.local.RF[[1]]$local.data

# the randomForest object itself
class(all.local.RF[[1]]$local.RF)
[1] "randomForest"
```

The `RunHRF` function contains additional arguments that can be controlled by the user (see the help file), among which the argument `importance` controls whether variable importance should be kept for each local classifier. If `importance` is set to the default value of `TRUE` in `RunHRF`, then the variable importance of each explanatory variable in each local classifier can be extracted using the function `ImportanceHie` and plotted using the function `PlotImportanceHie`. For the importance function the output can be arranged in a table or a 4 column format, and the plotting can be based on a heat map (**Figure 2.5**) or bubble size (**Figure 2.6**). Note for example, that the variable ‘*eicosenoic*’ is the most important variable in classifier C.1, yet it is not used to further separate classes lower down the class hierarchy.

```
# variable importance and plotting of importance
impor.hie.RF.00 <- ImportanceHie(hie.RF      = hie.RF.00,
                                format.out = c("col.4.out", "table.out"))

# The table format
imp.val.table <- impor.hie.RF.00$imp.val.table

# the plotting function uses the 4 column format - heat map plot
PlotImportanceHie(input.data = impor.hie.RF.00$imp.var.4.col,
                  X.data      = 2,
                  Y.data      = 3,
                  imp.data     = 4,
                  plot.type    = "Tile",
                  X.Title      = c("Parent node Name"),
                  Y.Title      = c("Explanatory variable"),
                  imp.title     = c("Mean \n Decrease \n in \n Accuracy"))

# bubble plot
PlotImportanceHie(input.data = impor.hie.RF.00$imp.var.4.col,
                  X.data      = 2,
                  Y.data      = 3,
                  imp.data     = 4,
                  plot.type    = "Bubble",
                  X.Title      = c("Parent node Name"),
                  Y.Title      = c("Explanatory variable"),
                  imp.title     = c("Mean \n Decrease \n in \n Accuracy"))
```

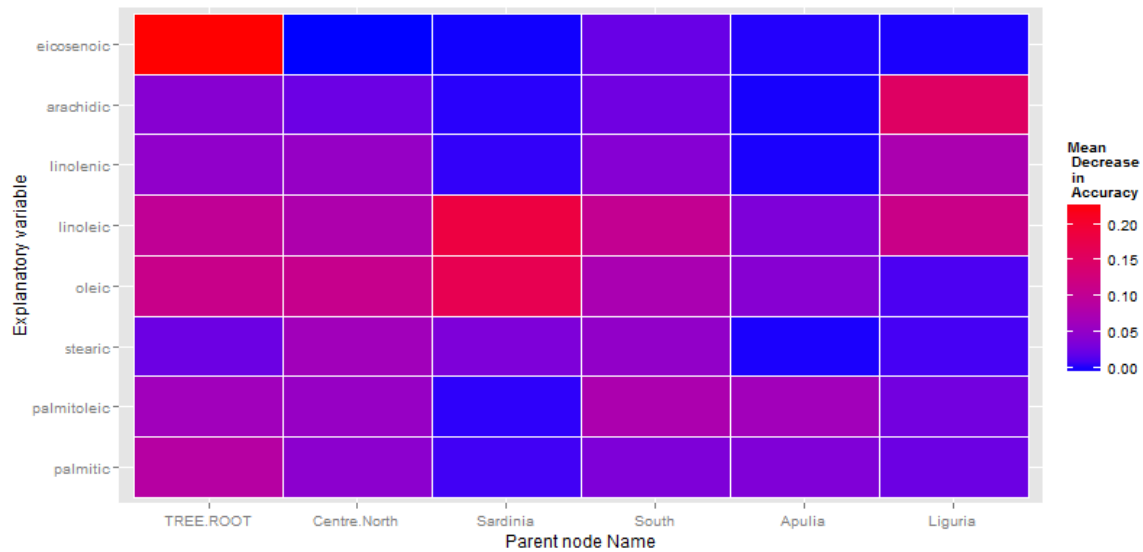


Figure 2.5: Variable importance plot at each level of the class hierarchy for the OliveOilHie data-set, using the 'PlotImportanceHie' function with the `plot.type="Tile"` option.

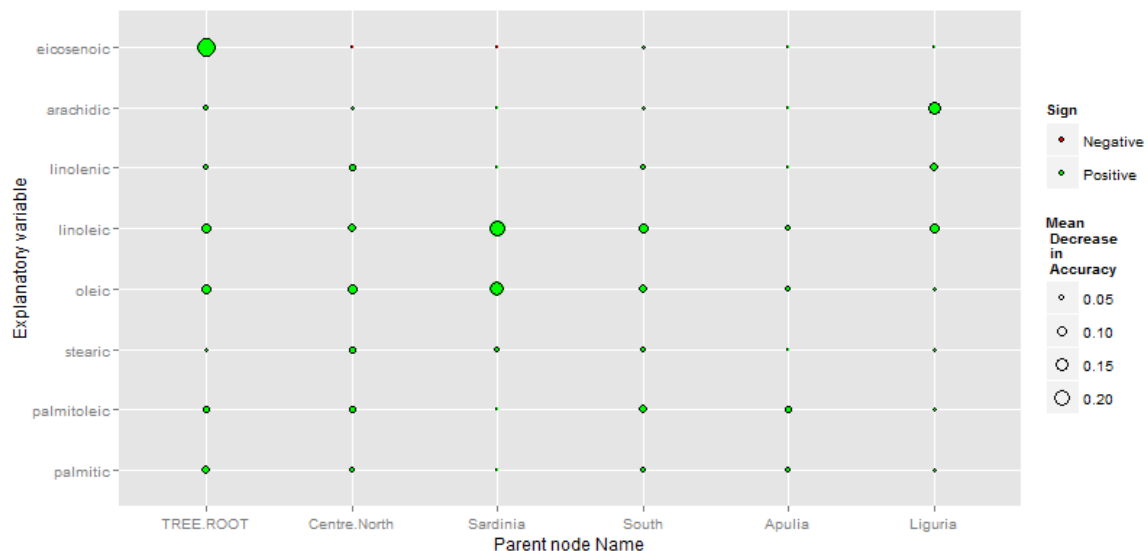


Figure 2.6: Variable importance plot at each level of the class hierarchy for the OliveOilHie data-set, using the 'PlotImportanceHie' function of 'HieRanFor' with `plot.type="Bubble"` option.

An object of class 'HRF' as created by the function `RunHRF` is used as the input in most other functions of the package. As an S3 class, it can also be used with a `predict` function, which will extract the proportion of OOB votes that each case received in each local classifier. The proportion of OOB votes can then be used to estimate the multiplicative proportion of votes using the `GetMultPropVotes` function. The multiplicative proportion of votes multiplies the row votes along every path down the class hierarchy and returns the multiplicative proportion of votes either for the entire class hierarchy or for all level (when setting `all.levels = TRUE` in the call for `GetMultPropVotes`).

```

Predict.00 <- predict(object = hie.RF.00,
                      train.predict = TRUE)

names(Predict.00)
[1] "prop. vote. train"

# dataframe with the row proportion votes for each class in each local
# classifier. Each row is for a single case.
prop.vote.train <- Predict.00$prop.vote.train

# the proportion of votes can be used to get the multiplicative proportion
# of votes for each terminal node. This can be done for the entire class
# hierarchy or a more detailed options that stops at every level.
multi.prop.votes.full <- GetMultPropVotes(prop.vote = prop.vote.train,
                                         unique.path = unique.path,
                                         all.levels = TRUE)

names(multi.prop.votes.full)
[1] "prop. multiplicative. votes. L1" "prop. multiplicative. votes. L2" "prop. multiplicative. votes. L3"

# all the nodes of level 1
multi.prop.votes.L1 <- multi.prop.votes.full[[1]]
names(multi.prop.votes.L1)
[1] "train. or. test" "case. ID" "Centre. North" "Sardinia" "South"
# same as:
nodes.info[nodes.info$node.level==1, "node.name"]
[1] "Centre. North" "Sardinia" "South"

# all the nodes of level 2 - internal (e.g. Liguria) and terminal (e.g. Umbria)
multi.prop.votes.L2 <- multi.prop.votes.full[[2]]
names(multi.prop.votes.L2)
[1] "train. or. test" "case. ID" "Liguria" "Umbria" "Sardinia. coast"
[6] "Sardinia. inland" "Apulia" "Calabria" "Sicily"
# same as:
nodes.info[nodes.info$node.level==2, "node.name"]
[1] "Liguria" "Umbria" "Sardinia. coast" "Sardinia. inland" "Apulia"
[6] "Calabria" "Sicily"

# all terminal nodes, including those ending at level 2
multi.prop.votes.L3 <- multi.prop.votes.full[[3]]
names(multi.prop.votes.L3)
[1] "train. or. test" "case. ID" "Liguria. east" "Liguria. west" "Umbria"
[6] "Sardinia. coast" "Sardinia. inland" "Apulia. north" "Apulia. south" "Calabria"
[11] "Sicily"
# same as:
nodes.info[nodes.info$term.int.node=="term.node", "node.name"]
[1] "Umbria" "Sardinia. coast" "Sardinia. inland" "Calabria" "Sicily"
[6] "Apulia. north" "Apulia. south" "Liguria. east" "Liguria. west"

# different from:
nodes.info[nodes.info$node.level==3, "node.name"]
[1] "Apulia. north" "Apulia. south" "Liguria. east" "Liguria. west"

```

The `predict.HRF` (`predict`) and `GetMultPropVotes` functions can be used on the training data as we have done here, yet they are more useful for predicting the row and multiplicative proportion of votes for new data (that that was not included in the training set- see below). For the training data set we recommend using the wrap-up function `PerformanceHRF` that returns the same output as both `predict` `predict.HRF` (`predict`) and `GetMultPropVotes` along with various flat and hierarchical performance measures. The `PerformanceHRF` function can use any of the 3 crisp rules described in section 2.2 and **Figure 2.3**, can return both flat and hierarchical performance

indices and can be applied on the overall confusion table or at the node level. Note, that the multiplicative permutations option for the crisp rule is still experimental and should be used with caution. This option usually returns lower accuracy values than the stepwise majority or multiplicative majority rules.

```
perf.HRF.00 <- PerformanceHRF(hie.RF = hie.RF.00,
                             per.index = c("flat.measures", "hie.F.measure"),
                             crisp.rule = c("stepwise.majority",
                                             "multiplicative.majority",
                                             "multiplicative.permutation"),
                             perm.num   = 10, div.print = 2)

names(perf.HRF.00)
[1] "raw.votes"      "crisp.case.class" "hie.performance"
[4] "multiplicative.prop" "nodes.measures.columns" "call"

# same output as predict
raw.votes.00 <- perf.HRF.00$raw.votes

# same output as GetMultPropVotes with all.levels = FALSE
multiplicative.prop.00 <- perf.HRF.00$multiplicative.prop

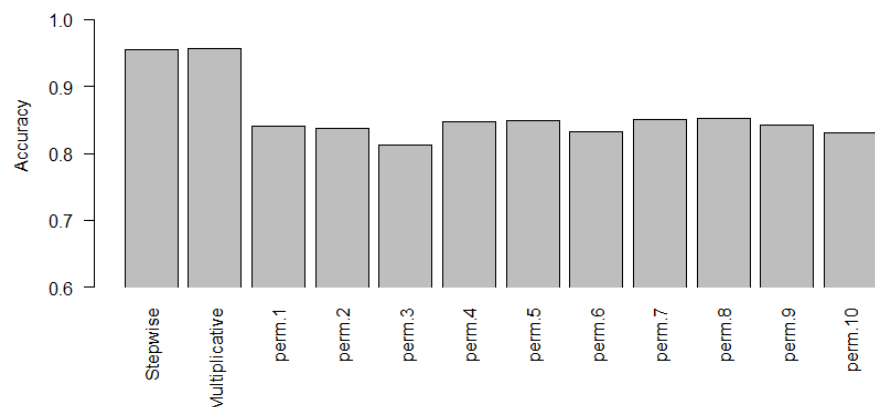
# all the crisp classes for each case
crisp.case.class.00 <- perf.HRF.00$crisp.case.class

# all the performance measures
hie.performance.00 <- perf.HRF.00$hie.performance

# note the decrease in accuracy for the permutation (Figure 2.7)
par(mar=c(6, 4, 1, 1))
names.arg <- hie.performance.00$crisp.rule
names.arg[1:2] <- c("Stepwise", "Multiplicative")

barplot(names.arg = names.arg,
        height    = hie.performance.00$Accuracy,
        ylim      = c(0.6, 1.05),
        xpd       = FALSE,
        las=2,
        cex.names = 1)
```

Figure 2.7: The accuracy (proportion of cases correctly classified) in the OliveOilHie data set using the 3 different crisp rules (10 permutations). Figure generated in the code above.



If one of the aims of the analysis is to compare the output of the hierarchical model to the original flat classification, while estimating both hierarchical and flat performance measures, the function `PerformanceFlatRF` takes as input an object of class 'HRF', runs a flat classifier and estimate all performance measures using the multiplicative and permutation based crisp rules. The step-wise majority rule cannot be applied on the flat model since the model does not provide the proportion of OOB votes at all levels of the class hierarchy.

```
# flat classification with hierarchical performance measures
flat.RF.OO <- PerformanceFlatRF(hie.RF = hie.RF.OO,
                               per.index = c("flat.measures",
                                              "hie.F.measure"),
                               crisp.rule = c("multiplicative.majority",
                                              "multiplicative.permutation"),
                               perm.num = 10, div.print = 2)

names(flat.RF.OO)
[1] "flat.RF"
[4] "nodes.measures.columns" "crisp.case.class" "hie.performance"

# an object of class randomForest
class(flat.RF.OO$flat.RF)
[1] "randomForest"

# votes and crisp class (majority rule)
votes.flat <- flat.RF.OO$flat.RF$votes
flat.RF.OO$crisp <- flat.RF.OO$crisp.case.class

# both hierarchical and flat measures of accuracy + permutations
hie.perf.flat <- flat.RF.OO$hie.performance
```

It is important to note that the training data is used to 'learn' the rules that link the explanatory variables with the different habitat in different levels of the class hierarchy and to assess the performance of the HRF model. However, in many cases our main aim is to use the model to predict on an external `new.data` for which we usually do not know the 'true' class. In some cases we may wish to save some of the training data as an external new cross validation data set. In this case, we may need to assess the performance of the `new.data` as well. The package `HieRanFor` provides several functions that allows full functionality for `new.data` as well as performance assessment if needed. However, before describing these functions, 2 important points must be noted:

1. Each case in `new.data` should have a unique case ID, which is also different from the case IDs of the training data. The reason is simple- case IDs are used to identify if a case was in-bag or OOB at any given tree of any given `randomForest`. If a case in `new.data` have the same case ID as a case in the training data, results from all trees in which the training data case was in-bag will be incorrectly ignored.
2. HRF is based on a set of `randomForest` models as local classifiers, and therefore is subject to the same limitations of `randomForest`. That is, a classifier cannot predict any class that

was not included in its training data. Similarly, for categorical explanatory variables, `randomForest` cannot predict for factor levels that were not included in the training data. We note here that `randomForest` will work if the categorical variable will contain 0 examples of a certain factor level, as long as it is registered as a factor level. Therefore, before running the first HRF analysis using the function `RunHRF`, we strongly recommend joining the factor levels of the training data and `new.data` for all categorical variables (in the hierarchy or explanatory). The `HieRanFor` package contains a utility function that joins the levels of two vectors (`JoinLevels`).

As we do not have external validation data for the `OliveOilHie` data-set, we will be using the same data both as training data and validation data. Therefore, we first need to change the `case.ID` of all cases in the version of `OliveOilHie` that we will use as external validation set. Here, there is no need to join the levels as they are identical, yet we provide an example on how it should be done below. As we are using the same data as training and `new.data` the results exemplifies the importance of using only the OOB votes. Once all the cases have unique case IDs and `RunHRF` was executed after joining the factor levels of all categorical variable, we can predict the raw votes and multiplicative proportion of votes using the `S3 predict` function and the `GetMultPropVotes` function, respectively.

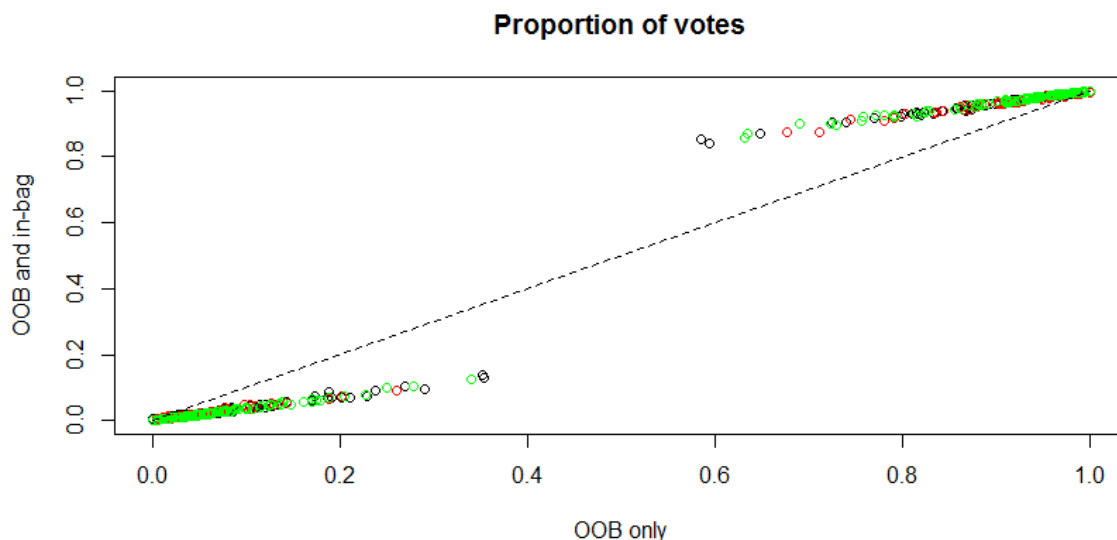


Figure 2.8: The proportion of votes in classifier C.1 that each case received for each category (using the `OliveOilHie` data-set), when using only the OOB votes vs. when using all votes. When a case is in the in-bag of a classification tree, the tree is built to such that it can correctly identify the cases true class. Therefore, when all votes are used all cases received higher proportion of votes for their true class and lower for all other classes. Note the corresponding higher accuracy levels in **Figure 2.9** when using all votes. The figure is generated in the code below.

```

# Change case.IDs for the version used as new.data
OliveOilHie.2 <- OliveOilHie
OliveOilHie.2[, "case.ID"] <- c((max(OliveOilHie$case.ID)+1):
                               (max(OliveOilHie$case.ID)+572))

# Join factor levels for all categorical variables, in this case L1, L2 # and L3
names(OliveOilHie)[2:4]
[1] "L1" "L2" "L3"

# run the JoinLevels function
for (i in 2:4){
  new.Vec <- JoinLevels(vector.1 = OliveOilHie[, i],
                        vector.2 = OliveOilHie.2[, i])
  # Re-assign to the original vectors
  OliveOilHie[, i] <- new.Vec[[1]]
  OliveOilHie.2[, i] <- new.Vec[[2]]
}

# predict for new.data
# for new.data, only the cell.ID and the explanatory variables are needed
Predict.OO.new <- predict(object = hie.RF.OO,
                          train.predict = FALSE,
                          new.data      = OliveOilHie.2[, c(1,5:12)],
                          new.data.case.ID = 1,
                          new.data.exp.var = NULL,
                          bind.train.new  = FALSE)

names(Predict.OO.new)
[1] "prop. vote. new"

prop.vote.new <- Predict.OO.new$prop.vote.new # OOB + in bag

# note the difference in the proportion of votes for the three classes of classifier C.1
# when using all trees compared to using only the OOB votes - Figure 2.8
par(mar=c(4, 4, 3, 2))
plot(prop.vote.train$Centre.North, prop.vote.new$Centre.North, col="black",
     main = "Proportion of votes",
     xlab = "OOB only", ylab = "OOB and in-bag")

points(prop.vote.train$Sardinia, prop.vote.new$Sardinia, col="red")
points(prop.vote.train$South, prop.vote.new$South, col="green")
lines(x=c(0,1), y=c(0,1), lty=2)

# Multiplicative proportion of votes for new.data
multi.prop.votes.new <- GetMultPropVotes(prop.vote = prop.vote.new,
                                         unique.path = unique.path,
                                         all.levels = TRUE)

# all the nodes of level 1
multi.prop.new.L1 <- multi.prop.votes.new[[1]]
names(multi.prop.new.L1)
[1] "train. or. test" "case. ID" "Centre. North" "Sardinia" "South"

# all the nodes of level 2 - internal (e.g. Liguria) and terminal (e.g. Umbria)
multi.prop.new.L2 <- multi.prop.votes.new[[2]]
names(multi.prop.new.L2)
[1] "train. or. test" "case. ID" "Liguria" "Umbria" "Sardinia. coast"
[6] "Sardinia. inland" "Apulia" "Calabria" "Sicily"

# all terminal nodes
multi.prop.new.L3 <- multi.prop.votes.new[[3]]
names(multi.prop.new.L3)
[1] "train. or. test" "case. ID" "Liguria. east" "Liguria. west" "Umbria"
[6] "Sardinia. coast" "Sardinia. inland" "Apulia. north" "Apulia. south" "Calabria"
[11] "Sicily"

```

In many cases the row and multiplicative votes are not enough and the crisp class is also needed for mapping. Therefore, the package `HieRanFor` also contains the function `PredictNewHRF` that provides the raw and multiplicative votes yet takes one additional step and decide on the best class for each case using any of the 3 crisp rules. In addition, the function `PerformanceNewHRF` does the same as `PredictNewHRF` with the addition of estimating all the performance measures if the ‘true’ class of each case in `new.data` is known (i.e. an external validation set).

```
# The raw proportion of votes, multiplicative proportion of votes and crisp class
Full.new <- PredictNewHRF(hie.RF = hie.RF.00,
  new.data = OliveOilHie.2[, c(1,5:12)],
  crisp.rule = c("stepwise majority",
    "multiplicative.majority",
    "multiplicative.permutation"),
  perm.num = 10, div.print = 2)

# The crisp class for each case is given in "crisp.case.class"
names(Full.new)
[1] "raw.votes" "multiplicative.prop" "crisp.case.class" "call"

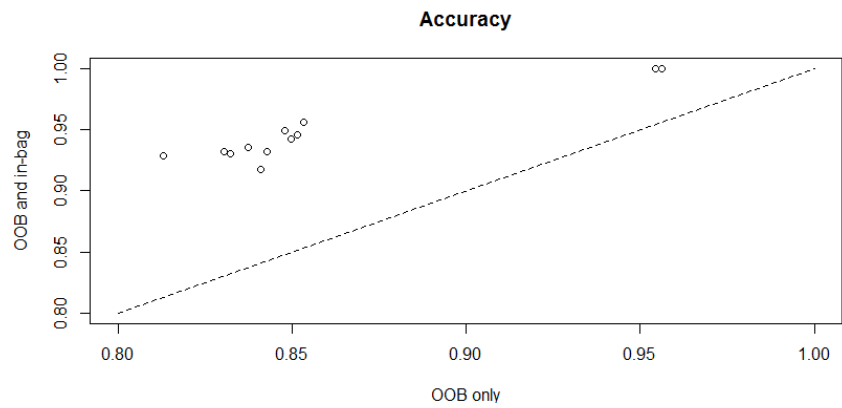
# if the true class of the new.data is known (i.e., an external validation set)
# the function PerformanceNewHRF applies the PredictNewHRF
# as well as estimate flat and hierarchical performance measures.
perf.new.data <- PerformanceNewHRF(hie.RF = hie.RF.00,
  new.data = OliveOilHie.2,
  new.data.case.id = 1,
  new.data.hie = c(2:4),
  crisp.rule = c("stepwise majority",
    "multiplicative.majority",
    "multiplicative.permutation"),
  per.index = c("flat.measures",
    "hie.F.measure"),
  perm.num = 10, div.print = 2, by.node = TRUE)

names(perf.new.data)
[1] "raw.votes" "crisp.case.class" "hie.performance"
[4] "multiplicative.prop" "nodes.measures.columns" "call"

# the crisp class according to each majority rule + permutation for the new data
perf.new.crisp <- perf.new.data$crisp.case.class

# the performance measure. note the increased accuracy relative to PerformanceHRF
# since we have not used only the OOB cases. Figure 2.9 below.
perf.new.hie.perf <- perf.new.data$hie.performance
plot(hie.performance.00$Accuracy,
  perf.new.hie.perf$Accuracy, main="Accuracy",
  xlab = "OOB only", ylab = "OOB and in-bag", xlim=c(0.8,1), ylim=c(0.8,1))
lines(x=c(0.8,1), y=c(0.8,1),lty=2) # unity line
```

Figure 2.9: The accuracy of the HRF analysis in the OliveOilHie data-set is higher when using all votes than when using only the OOB. The group of points on the left are for the 10 permutations, while the two points in the right are for the stepwise and multiplicative majority rule. Note that all points fall above the line of unity. The figure is generated in the code above.



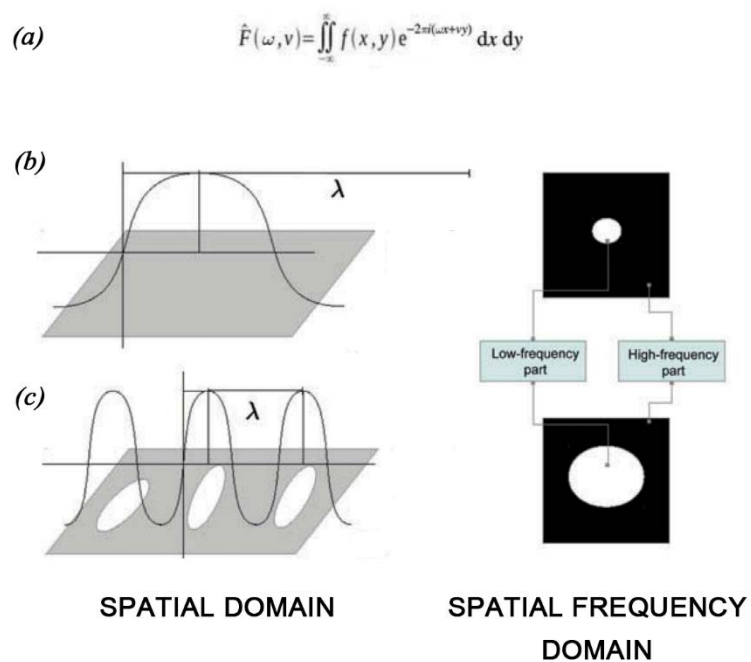
3. FOURIER TRANSFORMS

3.1. Introduction

Multi-temporal analysis of landscape fragmentation is an important tool to trace changes in the ecosystems and its biodiversity status and distribution (Mertens and Lambin 1997). The first step to perform such analysis is the development of land use maps which use a previously prepared set of categories to classified land use/cover types. Such a a-priori categorization implies a loss in the data content like the gradual process of fragmentation, which in turn, results in a loss of detail of the fragmentation process at smaller scales (Palmer et al. 2002). This is of great relevance if the purpose of the research is to monitor the changes the effect of these changes in the landscape and the related effect on the ecosystem types and biodiversity.

The method presented here, and recently published in a scientific journal (Rocchini et al. 2013), deals with the drawbacks of the classification methods by using a Fourier Transform method, which does not rely on a previous classification and detects the changes in the landscape based on continuous transformations (Rocchini et al. 2013).

Figure 3.1: The function $f(x)$ is transformed applying the Fourier transform (a) into sinusoidal functions of low (b) and high (c) frequency, where lower frequencies are located at the centre of the plot, while the higher frequencies are plotted outwards.



3.2. Approach

Based on the Fourier theorem (Fourier 1822), the spatial domain (represented as a continuous function $f(x)$) can be transformed into a continuum of sinusoidal functions with varying frequency (ω) (**Figure 3.1a**). This can be extended into a two dimensional function, $f(x,y)$ in which case the frequency (ω) coordinates are represented as ν . Once the images have been processed using the Fourier transform,

lower frequencies are generally plotted at the centre of the Fourier spectrum (**Figure 3.1b**), while higher frequencies are plotted outward (**Figure 3.1c**).

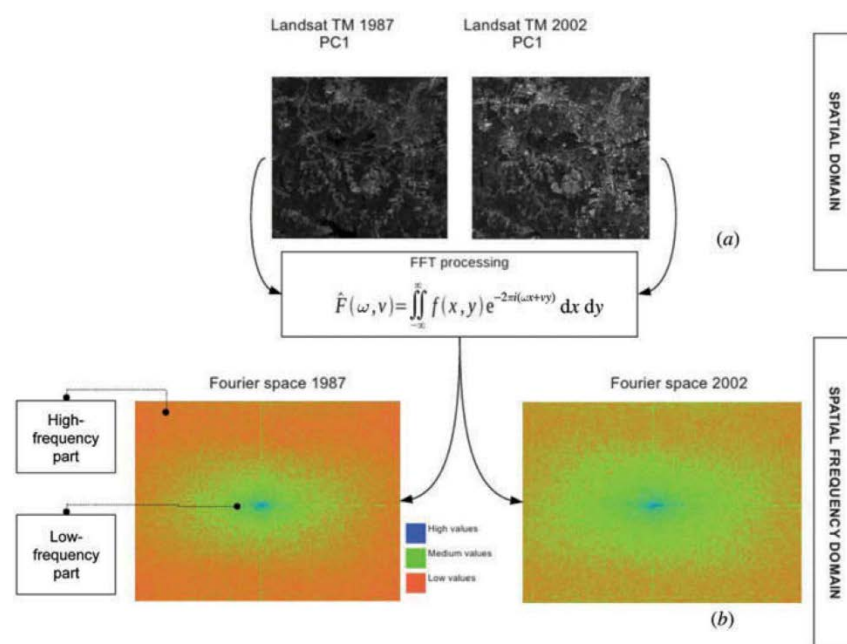
As describe in Rocchini et al. (2013), as fragmentation of the landscape increases so does the frequency (i.e. variation in the spatial domain) and the resulting Fourier spectrum shows higher values at the high frequency part of the spectrum.

3.3. Main features

The Fourier transform method was implemented in the free and open-source software Grass GIS using the Fast Fourier Transform tool (FFT) **i.fft** function devised by Satnik and Clements (Grass GIS manual page <http://grass.osgeo.org/grass64/manuals/i.fft.html>). The i.fft function is based on the FFT algorithm given by Frigo and Johnson (2005).

The analysis of the resulting frequency plots is implemented using R software, package spgrass6 (Bivand 2014) to import the images from Grass GIS and the packages sm (Bowman and Azzalini 2013) and hexbin (Carr et al. 2013) to perform the analysis and plot the results.

Figure 3.2: Example of the Fourier transforms approach and outcomes on two images of the same area but at different years (1987 and 2002). The 1987 Fourier spectrum shows less heterogeneity than the 2002 spectrum, demonstrating an increase in the landscape fragmentation of the area through time (Rocchini et al. 2013).



3.4. Tutorial and examples

The process consists of a series of steps (**Figure 3.2**) that start with the initial set up of the area of interest, the estimation of the principal components of the images to process, the actual calculation of the Fourier transform, and finally, the analyses of the resulting plots in R.

The code described below is available at:

http://grasswiki.osgeo.org/wiki/Fourier_transforms_for_multitemporal_analysis

```
# GRASS GIS: Fourier transform calculation

# The data used in this example is from the North Carolina set available for free # at:
http://grass.osgeo.org/download/sample-data/

#Setting the region of interest based on a Landsat image

g.region rast=lsat7_2002_10 -p

# Performing PCA on a multitemporal set of images (Landsat images in this case)

# 1987

i.pca
input=lsat5_1987_10,lsat5_1987_20,lsat5_1987_30,lsat5_1987_40,lsat5_1987_50,lsat5_1987_70
output=lsat5_1987_pca

# 2002

i.pca
input=lsat7_2002_10,lsat7_2002_20,lsat7_2002_30,lsat7_2002_40,lsat7_2002_50,lsat7_2002_70
output=lsat7_2002_pca

#Showing the first principal component

#1987

d.mon x0
d.rast lsat5_1987_pca.1

#2002

d.mon x1
d.rast lsat7_2002_pca.1

#Fourier transform

#1987

i.fft input=lsat5_1987_pca.1 real=lsat5_1987_pca1.real imaginary=lsat5_1987_pca1.imag

#2002

i.fft input=lsat7_2002_pca.1 real=lsat7_2002_pca1.real imaginary=lsat7_2002_pca1.imag
#Creating Fig. 2 in Rocchini et al. (2013)

#1987

d.mon x2
d.rast lsat5_1987_pca1.real

#2002

d.mon x3
d.rast lsat7_2002_pca1.real

#Example with MODIS data: Fig. 6 in Rocchini et al. (2013)

#Import data: image covering a temporal period from 2005-02-02 to 2005-03-05 (band 2 being
used)

r.in.gdal input=Goodes.EUAS.2005033.band2.tif output=Goodes.EUAS.2005033.band2

#Import data: image covering a temporal period from 2005-07-12 to 2005-08-12 (band 2 being
used)
```

```
r.in.gdal input=Goodes.EUAS.2005193.band2.tif output=Goodes.EUAS.2005193.band2

#Fourier transforms

i.fft input=Goodes.EUAS.2005033.band2 real=Goodes.EUAS.2005033.band2.real
imaginary=Goodes.EUAS.2005033.band2.imag

i.fft input=Goodes.EUAS.2005193.band2 real=Goodes.EUAS.2005193.band2.real
imaginary=Goodes.EUAS.2005193.band2.imag

#Statistical analysis in R

#Importing the Fourier image from GRASS GIS using the spgrass6 library

library(spgrass6)

fft.images<-readRAST6(c('lsat5_1987_pca1.real','lsat7_2002_pca1.real'),
  cat=c(F,F), ignore.stderr=TRUE, plugin=NULL)

#where lsat5_1987_pca1.real and lsat7_2002_pca1.real are the Fourier transform images
derived from GRASS GIS

#Kernel Density Plot: Fig. 4 in Rocchini et al. (2013)

library(sm)

sm.density(fft.images$lsat5_1987_pca1.real, lty=1, lwd=2, ann=T,
  xlim=c(0,255), ylim=c(0,0.013), xlab='Fourier transform value', col='blue', cex.label=4)

par(new=T)

sm.density(fft.images$lsat7_2002_pca1.real,lty=1, lwd=2, ann=F,
  xlim=c(0,255), ylim=c(0,0.013), xlab=' ', col='red', cex.label=4)

legend(110, 0.012, c('Density function 1987', 'Density function 2002'),
  fill = c('blue', 'red'), cex=1.2, box.lty=0)

#Hexagon binning: Fig. 5 in Rocchini et al. (2013)

library(hexbin)

hbin <- hexbin(fft.images$lsat5_1987_pca1.real, fft.images$lsat7_2002_pca1.real, xbins =
50)

plot.bin<-plot(hbin, style = 'nested.lattice', legend=F,
  xlab='Fourier transform value 1987', ylab='Fourier transform value 2002')

hexVP.abline(plot.bin$plot.vp, 0, 1)
```


4. LAND SURFACE TEMPERATURE

4.1. Introduction

Environmental indicators such as temperature are essential for the assessment of current and potential species distributions as well as the study of future trends due to environmental change. These assessments are performed at multiple spatial scales from the global or multinational to the local (field site) scale, which implies a variation in the resolution of the environmental data required for its development.

Sources of data on land surface temperature include ground meteorological stations and satellites with thermal infrared sensors, and while each one of the sources can provide accurate estimates, both are sensitive to gaps in the data quality related to the type of data they are derived from. On one hand, maps derived from meteorological stations have gaps due to the spatial aggregation of weather stations, that is the effect of areas with a high number of weather stations per km² vs. areas with a low (or zero) weather stations per km². On the other hand, satellite-based data are not complete as they have gaps due to clouds.

One of remote sensing instruments, widely used in research, is the Moderate Resolution Imaging Sensor (MODIS) on board the Terra and Aqua satellites. MODIS-terra has global coverage starting from March 2000 onwards, twice per day temporal resolution and a number of layers with information of land surface properties (Coops et al. 2009). Such data is extremely valuable but requires preparation and processing before it can be accurately used in ecological research, to minimize the effect of clouds and to include the variation in the values due to differences in the topography (Neteler 2010).

4.2. Approach

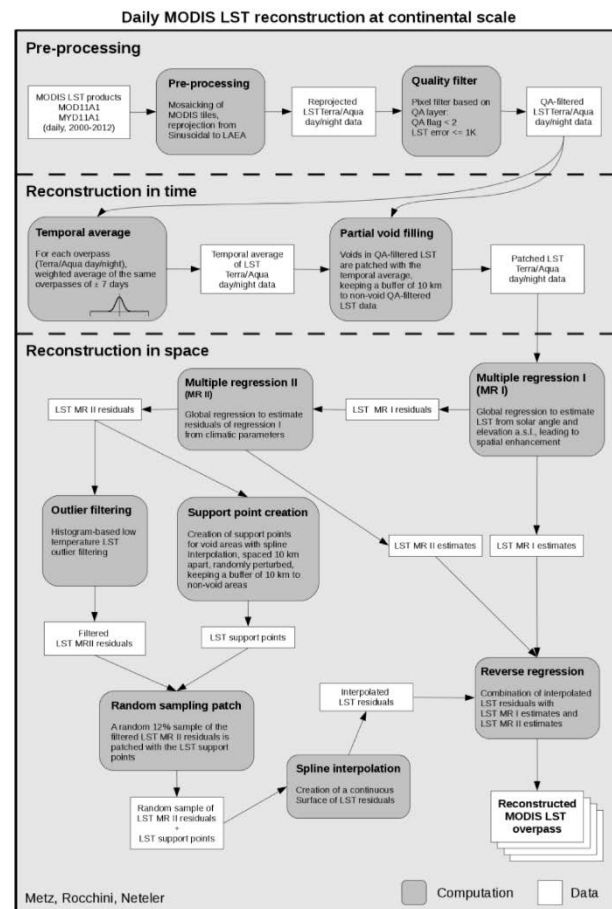
The main objective of the method presented here, and recently published (Metz et al. 2014), is the construction of high-resolution Land Surface Temperature maps and related products. The process consists of three main steps: i) raw MODIS data is stitched together (mosaicked), re-projected and filtered, ii) gaps are removed (if needed) using temporal averaging, and iii) spatial statistical interpolation is applied to enhance spatial detail (**Figure 4.1**).

4.3. Main features

The method is applied using the following commands from GRASS GIS:

- [i.pca](#): Principal Components Analysis (PCA) for image processing
- [r.regression.multi](#): calculates multiple linear regression from raster maps
- [v.surf.bspline](#): performs bicubic or bilinear spline interpolation with Tykhonov regularization
- [pyMODIS](#): Free and Open Source Python based library to work with MODIS data
- [r.bioclim](#): calculates various bioclimatic indices from monthly temperature and optional precipitation time series (install in GRASS GIS 7 with "g.extension r.bioclim")

Figure 4.1: Flow diagram of the method developed (from Metz et al. 2014).



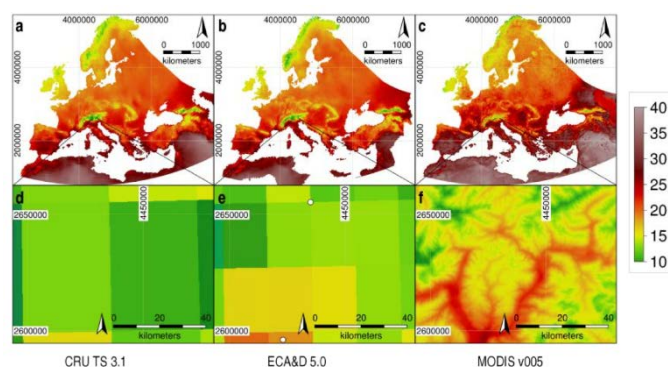
4.4. Tutorial and examples

The resulting LST maps are at a higher spatial resolution (250 m) than any previous temperature time series, demonstrating the high variation in temperature (**Figure 4.2**), at smaller scales, which might have profound effects in the ecological modelling process and outputs.

Reconstructing the LST data for Europe is very intensive (415 million pixels per map) and generates a huge dataset (13 TB); the MODIS LST dataset is currently available for download at:

<http://gis.cri.fmach.it/eurolst/>

Figure 4.2: Differences in resolution at large (above) and small (below) spatial scales of two temperature datasets and the MODIS LST maps developed with the method presented. (a) Climatic Research Unit, (b) European Climate Assessment and Dataset and, (c) reconstructed MODIS LST (from Metz et al. 2014).



5. UPSCALING DIVERSITY

5.1. Introduction

Policy is often concerned with the preservation of biodiversity at coarse spatial scales, national, regional, continental (e.g., Gothenburg targets, 2001) or global levels (e.g., CBD, 2002), whereas most biodiversity monitoring is conducted at very fine spatial scales (sometimes as little as a fraction of a m²). Even with considerable sampling effort at fine spatial scales, not all species occurring in larger extents are likely to be sampled. Therefore, the total number of species found when pooling all samples is generally an underestimate of the actual number of species found in the larger extent.

Over the years, various methods that aim to predict the actual number of species from an incomplete set of local scale samples have been developed (see Xu et al. 2012 and citations within) and many of these methods are available in open source platforms, e.g., the `specpool` function in the `vegan` R package (Oksanen et al. 2015)). However, most of the methods covered by such applications are non-parametric while other area-based methods remains mostly unused by both professionals and the general public. Therefore, one of the objectives of *task 3.2* was to develop and codify a set of models aiming to predict the hard-to-measure property of the number of species in a large extent from a widely available source of input data – usually a limited number of fine resolution samples randomly spread within the focal area. Among the various candidate methods, we focused on methods that due to their usage of additional information or robust theoretical background may provide higher accuracy than the non-parametric methods stated above. Our second objective was to make these advanced methods available for the general public by packing their complex mathematical formulation in easy-to-use executable functions. To meet these two aims we produced the R package ‘UpScaling’.

The ‘UpScaling’ package currently covers four different upscaling methods that differ from one another in their data requirements and in the theoretical basis used to deal with the non-additivity of species richness and the non-linearity of species accumulation curves (SAC) and species-area relationships (SAR). Among the four methods (**Table 5.1**), three require the occupancy-based community matrix (i.e., a presence/absence table for species in sites) as input, sometimes with additional information on the habitat identity of each sample or the location of the samples in the wider extent. One model is not based on the community matrix, but requires knowledge of the mean total abundance and species per sample.

Although the input data requirements of the models differ, all models provide predictions for the same property – the number of species in a large spatial extent. As the different models base their predictions on different assumptions, we expect the models to differ from one another in their predicted values. Therefore, we require knowledge of the true number of species if we wish to assess the performance of the models. With the improved accessibility of these techniques, it is our hope that they will be applied more widely, allowing the accumulation of a wider track record of their reliability

(or lack thereof). One of the challenges that lies ahead is to develop methods that estimate the confidence of the models' predictions without clear and complete knowledge of the true species richness.

In sections 5.2-5.5 we provide a detailed account on each method. For the first three methods we provide a general description of the theoretical background as well as mathematical formulation (when required). We refer the readers to the respective original publications for a more thorough discussion. For the nested SAC method (section 5.5) which was developed under the EU BON framework and is as of yet unpublished, we provide a more detailed account. Section 5.6 provides a general overview of the 'UpScaling' R package including a list of functions and installation guide. Finally, in section 5.7 we provide a tutorial that supplement the original help files of the package (the pdf *SI5.1- UpScaling.pdf*, downloadable from <http://www.eubon.eu/documents/1/>). Although most of the package was codified and prepared under EU BON, some of the functions are based on much appreciated original codes kindly supplied by Dr. Han Xu and Prof. Fangliang He.

Table 5.1: The four upscaling diversity methods currently covered in the package 'UpScaling'

Name (Source)	Required input data	Main functions in package
Simplified maximum entropy (Harte et al. 2009)	<ul style="list-style-type: none"> Mean number of individuals per sample Mean number of species per sample 	SimplMaxEntropyHarte
True & sampled SOD (Shen and He 2008)	<ul style="list-style-type: none"> presence / absence data table 	ShenHe08Model
T-S curve (Ugland et al. 2003)	<ul style="list-style-type: none"> presence / absence data table habitat identity of each sample (recommended) 	ToSpCurve
Nested SAC (Gavish et al. This deliverable)	<ul style="list-style-type: none"> presence / absence data table mean number of species per sample X and Y coordinate of each sample ASCII map for the entire extent 	PrepareNestedSacData
		PlotOnePath
		OptimNestedSac
		PredictNestedSac

5.2. Simplified Maximum Entropy

Harte et al. (2008) and Harte et al. (2009) developed a model that predicts the shape of the SAR based on the theory of Maximum Entropy. The model is based on relatively simple geometric and energetic constraints on ecological systems, such as the linear scaling of the total number of individuals of all species combined with area. Then the model assumes that the system will tend towards the most likely state consistent with the constraints, such that the information entropy is maximized. The resulting

model takes surprisingly little information to parameterize: it requires only the mean number of species found at a single reference scale (e.g. of a sample quadrat) and the mean number of individuals per sample at that scale.

The maximum entropy SAR model starts with the SAR equation:

$$S(A) = S_0 \sum_{n=1}^{N_0} [1 - P(o|n, A, A_0)] \cdot \phi(n|S_0, N_0) \quad \text{E5.1}$$

With S_0 being the number of species in the area A_0 , N_0 being the total number of individuals in area A_0 , and $S(A)$ being the expected number of species in a sub-area A of A_0 . The summation in equation 12 is over all possible population sizes $n=1, 2, \dots, N_0$ of the multiplication of two probabilities:

- $\phi(n|S_0, N_0)$ is the probability of the species abundance distribution, i.e. the probability that a randomly drawn species from the community will have a population size of n .
- $P(o|n, A, A_0)$ is the probability that none of the individuals of a species with a total of n individuals will be found in sub-area A , such that $[1 - P(o|n, A, A_0)]$ is the species' probability of occurrence in A .

Therefore, equation E5.1 estimates the number of species in sub-area A as the total number of species in A_0 multiplied by the probabilities of occurrence of each of those species. Next, Harte et al. (2009) incorporate constraints to the two probabilities listed above and solve for the maximum entropy solution using Lagrange multipliers. Note that in this form, equation E5.1 is a down-scaling method, rather than an up-scaling method, as it uses information on the number of species in the larger extent to predict the species richness in the smaller extent. However, Harte et al. (2009) simplify the solution for the halving of the area case to derive a specific equation that can be used for both down-scaling and up-scaling:

$$S(A) = S(2A) e^{\lambda_{\phi, 2A} A} - N(2A) \frac{1 - e^{-\lambda_{\phi, 2A} A}}{e^{-\lambda_{\phi, 2A} A} - e^{-\lambda_{\phi, 2A} (N(2A)+1)}} \cdot \left(1 - \frac{e^{-\lambda_{\phi, 2A} N(2A)}}{N(2A)+1} \right) \quad \text{E5.2}$$

With $\lambda_{\phi, 2A}$ being a Lagrange multiplier and $S(2A)$ the unknown number of species in the area $2A$. After assuming that total abundance of all species combined scales linearly with area, Harte et al. (2009) set $N(2A) = 2 \cdot N(A)$, thereby reducing the number of parameters in equation E5.2 to two – $\lambda_{\phi, 2A}$ and $S(2A)$. Therefore the model relies on another equation relating the same two parameters that arises from the maximum entropy constraints on the species abundance distribution:

$$\frac{S(2A)}{N(2A)} \cdot \sum_{n=1}^{N(2A)} e^{-\lambda_{\phi, 2A} n} = \sum_{n=1}^{N(2A)} \frac{e^{-\lambda_{\phi, 2A} n}}{n} \quad \text{E5.3}$$

Finally, the $S(2A)$ is found by numerically solving equation E5.2 and E5.3, and an iterative procedure can be used to up-scale to any large scale that is a multiple of two times the area of A (e.g., $S(4A)$, $S(8A)$, etc.).

The advantage of the simplified maximum entropy methods is in the relatively little information it requires for parameterization. However, it is important to note that the method only provides predictions for doublings of area, and cannot be used to predict the number of species in any area without an additional step of fitting a function to the models predictions.

5.3. True and Sampled SOD

Most spatially-implicit methods that estimate species richness in a large area from a set of random samples taken within it are based on resampling with replacement. Therefore, they are more applicable to mobile organisms, and less so for sedentary organisms. In fact, when applying a sampling-with-replacement based estimator to sedentary organisms, the predicted number of species will increase with the number of samples. Therefore, the estimators will tend to overestimate species richness when the number of samples increases, resulting in a greater, rather than smaller, deviation from the true species richness with increasing sampling intensity. In other words, for sedentary organisms (e.g., plants) sampling-with-replacement based estimators will not converge to the true value with increasing number of samples as expected from a good estimator.

Shen and He (2008) developed a novel incidence-based approach that relies on sampling without replacement. The basic idea behind the method is to estimate the number of unsampled species without the usage of complex combinatorial terms. To do so, Shen and He (2008) make two simplifying assumptions. First, they assume that the number of occupied quadrats of a species in sampled and unsampled locations (i.e. the true Species occupancy Distribution- SOD) follows a zero-truncated binomial distribution. The zero is truncated from the binomial distribution to ensure the probability distribution will predict the correct number of species when the entire area is sampled. Secondly, Shen and He (2008) assume that the species' probability of occupancy in a quadrat (that is required by the binomial distribution) follows a modified beta distribution.

The model starts with a zero truncated binomial distribution that describes the probability of each species ($i=1, 2, \dots, S$) occurring in exactly Φ_i quadrats, out of T quadrats covering the entire extent. The zero-truncated binomial distribution is parameterized by T and p_i (the binomial probability of occurrence of species i). The probability mass function (pmf) of the zero-truncated binomial distribution is thus given by:

$$P(\Phi_i = \varphi | p_i) = \binom{T}{\varphi} \cdot \frac{p_i^\varphi \cdot (1-p_i)^{T-\varphi}}{1-(1-p_i)^T} \quad \varphi = 1, 2, \dots, T \quad \text{E5.4}$$

Next, a random sample of t quadrats is taken from the above pmf, and the expression for the observed number of sampled quadrats in which species i is found (noted as X_i) can be developed as a hypergeometric distribution:

$$P(X_i = x | \Phi_i, p_i) = \frac{\binom{\Phi_i}{x} \cdot \binom{T-\Phi_i}{t-x}}{\binom{T}{t}} \quad \text{E5.5}$$

Shen and He (2008) then average equation E5.5 over all realizations of Φ_i , thereby retaining a pmf that is conditional only on p_i :

$$P(X_i = x | p_i) = \binom{t}{x} \cdot \frac{p_i^x \cdot (1-p_i)^{t-x}}{1-(1-p_i)^T} - \frac{(1-p_i)^T \cdot I(x=0)}{1-(1-p_i)^T} \quad x = 0, 1, 2, \dots, t \quad \text{E5.6}$$

With $I(\cdot)$ being an indicator function. As the number of species S and the binomial probability p_i of each species $i=1, 2, \dots, S$, are unknown, Shen and He (2008) assume that p_1, p_2, \dots, p_S are random draws

from a modified beta distribution, parameterized by $\alpha > 0$ and $\beta > 0$. Thus the unconditional distribution of X_i becomes:

$$P(X_i = x) = \begin{cases} K(\alpha, \beta) \cdot \binom{t}{x} \frac{\Gamma(x+\alpha) \cdot \Gamma(t+\beta-x)}{\Gamma(t+\alpha+\beta)} & x > 0 \\ K(\alpha, \beta) \left[\frac{\Gamma(\alpha) \cdot \Gamma(t+\beta)}{\Gamma(t+\alpha+\beta)} - \frac{\Gamma(\alpha) \cdot \Gamma(T+\beta)}{\Gamma(T+\alpha+\beta)} \right] & x = 0 \end{cases} \quad \text{E5.7}$$

With $K(\alpha, \beta)$ being a normalizing factor:

$$K(\alpha, \beta) = \left[\frac{\Gamma(\alpha) \cdot \Gamma(\beta)}{\Gamma(\alpha+\beta)} - \frac{\Gamma(\alpha) \cdot \Gamma(T+\beta)}{\Gamma(T+\alpha+\beta)} \right]^{-1} \quad \text{E5.8}$$

The above unconditional distribution is then used to parameterize a multinomial probability of f_k - the number of species that occurred exactly $k=0, 1, 2, \dots, t$ times in the set of t samples (i.e., the observed SOD). That is to say, Shen and He (2008) assume that $(f_0, f_1, f_2, \dots, f_t)$ is a multinomial distribution with a total S and probabilities $(\rho_0, \rho_1, \rho_2, \dots, \rho_t)$ that sum to one and satisfy: $\rho_k = P(X_i=k)$ of equation 4. The likelihood function of this multinomial distribution is given by:

$$L(S, \alpha, \beta) = \frac{S!}{(S-D)! \cdot \prod_{k=1}^t f_k!} \cdot \rho_0^{S-D} \cdot \prod_{k=1}^t \rho_k^{f_k} \quad \text{E5.9}$$

with D being the observed number of species in the set of t quadrats. An estimator of species richness is then given by finding the values of α, β and S that maximize the likelihood of equation E5.9.

However, Shen and He (2008) suggest further decomposing equation E5.9 into two terms:

$$L(S, \alpha, \beta) = L_b(S, \alpha, \beta) \times L_c(\alpha, \beta) \quad \text{E5.10}$$

The first term is a binomial likelihood function with respect to D , and the second term is the likelihood function in respect only to the shape of the observed species occupancy distribution:

$$L_b(S, \alpha, \beta) = \frac{S!}{(S-D)! \cdot D!} \cdot (1 - \rho_0)^D \cdot \rho_0^{S-D} \quad \text{E5.11}$$

$$L_c(\alpha, \beta) = \frac{D!}{\prod_{k=1}^t f_k!} \cdot \prod_{k=1}^t \left(\frac{\rho_k}{1-\rho_0} \right)^{f_k} \quad \text{E5.12}$$

This decomposed form allows finding first $\hat{\alpha}$ and $\hat{\beta}$ - the solution of α and β that maximizes the fit to the observed distribution of f_k (i.e., without the unknown f_k for $k=0$) using equation 9, and then finding the value of S that maximizes the likelihood according to equation 8, given $\hat{\alpha}$ and $\hat{\beta}$:

$$\hat{S}_{CMLE} = D \cdot \left[\frac{1 - \frac{\Gamma(\hat{\alpha}+\hat{\beta})}{\Gamma(\hat{\beta})} \cdot \frac{\Gamma(T+\hat{\beta})}{\Gamma(T+\hat{\alpha}+\hat{\beta})}}{1 - \frac{\Gamma(\hat{\alpha}+\hat{\beta})}{\Gamma(\hat{\beta})} \cdot \frac{\Gamma(t+\hat{\beta})}{\Gamma(t+\hat{\alpha}+\hat{\beta})}} \right] \quad \text{E5.13}$$

Finally, Shen and He (2008) also develop equations to estimate the variance around the predicted number of species.

Although preliminary analyses of several datasets suggest that the true and sampled SOD method can provide accurate predictions for the number of species in a given extent, it does not provide prediction for the entire shape of the SAR, as the T-S curve and nested SAC methods. In addition, the method can be considered as spatially explicit as it does not require any information on the spatial location of each sample. We note though, that the method is not entirely spatially implicit as the shape

of the observed SOD is probably highly dependent on the spatial distribution of the samples, as suggested by the effect of extent of SOD's shape (McGeoch and Gaston 2002).

5.4. Total-Species (T-S) Curve

Most assemblages have a complex covariance structure between species and sub-areas (e.g., habitats). This leads to a largely unrecognized aspect of predicting the number of species by up-scaling: with the addition of new sub-areas the observed species accumulation curve will not only extend the previous accumulation curve, but also tend to lie above the accumulation curves for smaller sub-areas. The rate of (vertical) increase of the species-accumulation curves provides the best estimate of total species richness. Ugland et al. (2003) first derived an exact analytical expression for the expectance and variance of the sample-based species accumulation curve in all random subsets from a given area (note that one of the required parameters of the expression is the actual number of species in the entire extent).

Next, the whole area is divided into sub-areas, and an increasing sequence of accumulation curves is constructed as follows. The first accumulation curve (the bottom curve) is obtained by taking the average of all single sub-areas. The second accumulation curve is obtained by taking the average of all accumulation curves based on two randomly chosen sub-areas. For example, if there are five sub-areas, the total number of subsets of two sub-areas is the binomial coefficient $5! / (2! \times 3!) = 10$, so the second accumulation curve will be the average of 10 curves. In the same way the third accumulation curve is the average of accumulation curves based on all possible subsets of three sub-areas. This procedure is repeated until we end up with the last accumulation curve which is obtained by a randomization of all available samples in the data set. It is the terminal points of this increasing sequence species accumulation curves that contain the crucial information of the accumulation rate of new species as sampling effort is increased to new sub-areas. The total species curve (the T-S curve) is then defined as the curve connecting these end points. In a semi-logarithmic plot these curves frequently appear linear, and the T-S curve estimator is then simply the linear extrapolation of the T-S curve to the whole area in the semi-log plot.

It is important to note that the T-S log-linear model requires *a-priori* grouping of samples to sub-areas (habitats) Therefore, it may be suitable for systems for which the set of fine-resolution samples are stratified according to habitats or land-covers. In other cases, samples may be grouped according to spatial proximity. Another option is to add an additional analytical step prior to the T-S curve analysis in which the samples are assigned to classes. For example, Jobe (2008) used a hierarchical clustering algorithm known as partitioning around medoids, that optimise both the class affiliation of each sample as well as the number of classes. However, any other unsupervised classification algorithm may be suitable as well. In fact, *kmeans* clustering is the default option in the *ToSpCurve* function of this R-package.

5.5. Nested SAC models

A set of fine-scale samples spread around a larger spatial extent can be used to estimate the number of species through two main approaches. First, one may choose a fixed starting point, and then examine all samples in a progressively wider areas around it, while monitoring the increase in the number of species with the increase in the extent of sampling. Second, one may hold the total extent of interest constant, and take progressively larger numbers of samples from within it, and explore the increase in the number of species with the increase in the number of samples. The first approach accounts for the change in extent yet ignores differences in the number of samples encountered as the area increases, and the inevitable effects of incomplete sampling. The second approach accounts for the number of samples, yet ignores their spatial locations and the change in effective extent of sampling as more samples are added. Alone, each approach seem incomplete, missing the component explicitly covered by the other. In task 3.2 we unified the two approach in a set of models that predict the number of species while accounting for both the extent and number of samples.

The first approach is closely related to what is referred to as nested species-area relationships (type I SAR *sensu* Scheiner 2003)- a biodiversity pattern that explores the change in the number of species with area, when each small area is completely contained within all larger areas. The nested SAR can be used to predict the number of species in a given extent by first fitting it with a descriptive mathematical model (e.g., power model), and then extrapolating to the required extent. However, when doing so, we traditionally assume that all the areas in the data-set were fully sampled, such that the observed number of species equals the real species richness. When the area is relatively small, it is sometimes possible to sample it entirely, yet as the area increases, a complete census becomes an impossible methodological challenge. As a result, the difference between the observed species richness and the real species richness in the data used to fit the model is likely to increases with area. Therefore, the predicted model, which is later used to extrapolate to any extent, may underestimate the real number of species.

The second approach, usually referred to as sample-based Species Accumulation Curves (SAC, see Ugland et al. 2003), explores analytically or through permutations the mean number of species encountered in one sample, two samples, three samples and so on until the full number of samples is reached. The number of species in a large extent can be predicted by fitting an equation to the SAC and extrapolating to the desired area (similar to the T-S curve above). The rate of increase in the number of species usually declines with the number of samples. However, SAC does not account for the extent actually sampled. If we extrapolate the SAC to the entire focal extent, we would expect it to overestimate the number of species, as the focal area is continuous in space, whereas the samples are likely to be dispersed. For example, imagine a 4×4 landscape in which we took 4 samples, located at the four corners. The 4 samples span the entire extent and as a results of the distance decay of similarity (Nekola and White 1999, Soininen et al. 2007) and the patchiness of most species' distribution patterns, they are expected to have more species than 4 adjacent samples. Indeed, for any

number of samples j the observed SAC lies above the expected number of species in a continuous piece of land with the same total area as the j samples combined. The only exception to the above rule is for $j=1$, for which the extent and area of a sample are identical, and the SAC reflects the mean alpha diversity. Thus, when extrapolating to the entire extent (16 in this example), the results from our initial sample of 4 will usually have more than their fair share of species, jeopardising our ability to extrapolate.

Here we describe a new method that unifies the nested SAR and the SAC, and which we consequently name the “nested SAC” approach. In this method we aim to predict the parameters of a descriptive model that describe the real, unbiased nested SAR, while accounting for the number of samples available at each point in the data set. Below we provide a more detailed mathematical development of the nested-SAC method. However, before we first need to clarify three points.

1. The nested SAR and SAC should provide similar predictions whenever a single sample is taken and this prediction should equal the mean alpha diversity (the mean species richness of a single sample). Therefore, the nested-SAC models are constrained to produce the mean alpha diversity for a single sample.
2. The nested-SAC method is based on finding the parameters of a descriptive nested SAR model, and the choice of descriptive model can greatly affect the predictions. In fact, many descriptive SAR models are described in the literature (Tjørve 2003, 2009) and we do not have the information to assess the different models’ performance when predicting the real number of species. Therefore, we rate models by their ability to predict the observed number of species and use modelling averaging techniques to account for uncertainty in model selection.
3. A single nested SAR model applied to an incompletely sampled area is expected to return a curve with irregular stairsteps (see figure 2 in Scheiner 2003), reflecting fast increase in the number of species whenever a boundary (or transition zone) between two communities is crossed. The exact area in which these boundaries are crossed may highly depend on the choice of starting location. In addition, none of the descriptive models is flexible enough to predict such a stairstep curve. Therefore, we recommend exploring various starting locations in a given landscape and averaging the predictions to a smoothed line.

Mathematical description of the nested SAC models

To develop the nested-SAC model we start with the equation for the SAC described in Ugland et al. (2003), which states that the expected number of species in a set of a samples taken within an extent A (with the units being the area of a single sample) is given by:

$$E(S|a, A) = \sum_{s=1}^{O(S|A)} \left[1 - \frac{\binom{A-n_s}{a}}{\binom{A}{a}} \right] = \sum_{k=1}^A \left[\left[1 - \frac{\binom{A-k}{a}}{\binom{A}{a}} \right] \cdot R_k \right] \quad \text{E5.14}$$

with $E(S|a, A)$ being the expected number of species in a set of a samples in extent A , $O(S/A)$ being the observed number of species in the entire extent if we had sampled all of it (i.e., the real number of

species above), n_s being the number of sites in which species s occurred (if we had sampled the entire extent A), and R_k being the number of species that occurred in exactly k sites in the entire extent.

To simplify E5.14 we define and rearrange to get:

$$(\delta|k, a, A) = \left[1 - \frac{\binom{A-k}{a}}{\binom{A}{a}} \right] \quad \text{E5.15}$$

$$E(S|a, A) = \sum_{s=1}^{O(S|A)} [(\delta|k, a, A)] = \sum_{k=1}^A [(\delta|k, a, A) \cdot R_k] \quad \text{E5.16}$$

In an ideal research, $O(S/A)$ as well as the shape of the SOD (i.e, the values of all R_k) are known. In such cases, $E(S/a, A)$ is the expected number of species under random sampling of exactly a samples from the A available ones. Unfortunately, in most cases, we have a known extent A , a set of samples (a) and the observed number of species that were recorded in the set of samples ($O(S/a, A)$), while $O(S/A)$ and R_k are unknown. Therefore, the challenge is to reverse the usage of E5.16 and thereby estimate the values of $O(S/A)$ and all R_k . To do so, we assume that $O(S/A)$ changes in a predicted manner with area (i.e., a nested SAR) and that knowledge of this change also provide knowledge on the values of R_k . We make use of a special property of nested-SAR – the ability to estimate R_k directly from the SAR equation. More specifically, if $E(S/A, par)$ is the expected number of species in the entire extent A according to a nested SAR model with parameters par , then R_k is given by E5.17, that when plugged into E5.16, results with E5.18

$$R_{k, par} = E(S|A - k + 1, par) - E(S|A - k, par) \quad \text{E5.17}$$

$$E(S|a, A, par) = \sum_{k=1}^A [(\delta|k, a, A) \cdot [E(S|A - k + 1, par) - E(S|A - k, par)]] \quad \text{E5.18}$$

Next, we are adding a constrain such that when $a=1$, the nested SAC model will predict the mean alpha diversity ($\hat{\alpha}$ -mean number of species per sample). To do so we force the total number of occupancies according to the nested SAR to equal the mean alpha diversity multiplied by the number of samples in the extent (E5.19). Then we multiply E5.18 with E5.19 and rearrange to get E5.20 and E5.21:

$$1 = (\hat{\alpha} \cdot A) / \sum_{k=1}^A (k \cdot R_{k, par}) \quad \text{E5.19}$$

$$E(S|a, A, par) = (\hat{\alpha} \cdot A) \cdot \sum_{k=1}^A \left[(\delta|k, a, A) \cdot \frac{[E(S|A - k + 1, par) - E(S|A - k, par)]}{\sum_{k=1}^A [k \cdot [E(S|A - k + 1, par) - E(S|A - k, par)]]} \right] \quad \text{E5.20}$$

$$E(S|a, A, par) = (\hat{\alpha} \cdot A) \cdot \sum_{k=1}^A \left[(\delta|k, a, A) \cdot \frac{R_{k, par}}{\sum_{k=1}^A [k \cdot R_{k, par}]} \right] \quad \text{E5.21}$$

E5.21 is the general form of the nested SAC, with the exact shape of $R_{k, par}$ being dependent on the choice of SAR model. This equation predicts the expected number of species according to a nested SAR model with parameters par in an area of size A if we had sampled a set of a samples within it. Given the values of a , A , and par we can estimate the difference between the observed and predicted number of species ($O(S/a, A) - E(S/a, A, par)$) and find the values of par that minimise the residuals sum of squares, using optimization algorithms. After finding the optimal par values, we can set $a=A$ in E5.20 or E5.21 and get the prediction for the real number of species in area A (i.e., $E(S/A, par)$).

Table 5.2: The full list of SAR models covered in the nested SAC method, including the codes used to represent the models in the R package.

Function	Code	R _{k,par}
Power	Pow	$(A - k + 1)^z - (A - k)^z$
Extended power 1	EP1	$(A - k + 1)^{b \cdot (A - k + 1)^d} - (A - k)^{b \cdot (A - k)^d}$
Extended power 2	EP2	$(A - k + 1)^{b \cdot (d / (A - k + 1))} - (A - k)^{b \cdot (d / (A - k))}$
Persistence model 1	Pe1	$(A - k + 1)^b \cdot \exp(-d \cdot (A - k + 1)) - (A - k)^b \cdot \exp(-d \cdot (A - k))$
Persistence model 2	Pe2	$(A - k + 1)^b \cdot \exp\left(\frac{-d}{(A - k + 1)}\right) - (A - k)^b \cdot \exp\left(\frac{-d}{(A - k)}\right)$
Logarithmic model	Log	$\log(A - k + 1) - \log(A - k)$
Kobayashi logarithmic	Kob	$\log\left(1 + \frac{(A - k + 1)}{b}\right) - \log\left(1 + \frac{(A - k)}{b}\right)$
Negative exponential	NeE	$\exp(-b \cdot (A - k)) - \exp(-b \cdot (A - k + 1))$
Common logistic	CoL	$\frac{1}{(1 + \exp(-b \cdot (A - k + 1) + d))} - \frac{1}{(1 + \exp(-b \cdot (A - k) + d))}$
Archibald logistic	ArL	$\frac{1}{(b + d^{A - k + 1})} - \frac{1}{(b + d^{A - k})}$
Gompertz	Gom	$\exp(-\exp(-b \cdot (A - k + 1) + d)) - \exp(-\exp(-b \cdot (A - k) + d))$
Extreme-value function	EVF	$[\exp(-\exp(b \cdot (A - k) + d)) - \exp(-\exp(b \cdot (A - k + 1) + d))]$
Monod	Mon	$\frac{(A - k + 1)}{(b + A - k + 1)} - \frac{(A - k)}{(b + A - k)}$
Asymptotic regression	AsR	$d^{-(A - k)} - d^{-(A - k + 1)}$
Rational function	Rat	$\frac{(1 + z \cdot (A - k + 1))}{(1 + d \cdot (A - k + 1))} - \frac{(1 + z \cdot (A - k))}{(1 + d \cdot (A - k))}$
Chapman-Richards	ChR	$[1 - \exp(-b \cdot (A - k + 1))]^d - [1 - \exp(-b \cdot (A - k))]^d$
Cumulative Weibull	CuW	$\exp(-b \cdot (A - k)^d) - \exp(-b \cdot (A - k + 1)^d)$
Morgan-Mercer-Flodin	MMF	$\frac{(A - k + 1)^b}{(d + (A - k + 1)^b)} - \frac{(A - k)^b}{(d + (A - k)^b)}$
Lomolino function	Lom	$\frac{1}{(1 + b^{\log(d / (A - k + 1))})} - \frac{1}{(1 + b^{\log(d / (A - k))})}$
Cumulative beta-P	CuB	$[1 + ((A - k) / b)^z]^{-d} - [1 + ((A - k + 1) / b)^z]^{-d}$
Tjorve (2012)-model 1	Tm1	$\left[[C_1 + b \cdot \log_{10}(A - k + 1)]^{\frac{(A - k + 1)}{(A - k + 1 + n)}} \cdot [C_2 \cdot (A - k + 1)^z]^{1 - \frac{(A - k + 1)}{(A - k + 1 + n)}} \right] - \left[[C_1 + b \cdot \log_{10}(A - k)]^{\frac{(A - k)}{(A - k + n)}} \cdot [C_2 \cdot (A - k)^z]^{1 - \frac{(A - k)}{(A - k + n)}} \right]$
Tjorve (2012)-model 2	Tm2	$\left[[C_2 \cdot (A - k + 1)^z]^{\frac{(A - k + 1)}{(A - k + 1 + n)}} \cdot [C_1 + b \cdot \log_{10}(A - k + 1)]^{1 - \frac{(A - k + 1)}{(A - k + 1 + n)}} \right] - \left[[C_2 \cdot (A - k)^z]^{\frac{(A - k)}{(A - k + n)}} \cdot [C_1 + b \cdot \log_{10}(A - k)]^{1 - \frac{(A - k)}{(A - k + n)}} \right]$

The residuals sum of squares can also be used to calculate the model's likelihood, which along with the number of model parameters (plus one for the error) is used to calculate AICc and AICc weights (wAICc) (Burnham and Anderson 2002). When fitting more than one SAR model, the wAICc can be used as weights when averaging the predictions of the various models for the real observed or real number of species. In this function we cover a total of 22 published SAR models (see **Table 5.2** for a full list and mathematical description), including the 20 models described in Tjorve (2009) and two additional models from Tjorve (2012). We note, that the alpha diversity constraints usually result in a reduction of one in the number of parameters. This is true for 19 of the 22 models covered here. In the

logarithmic model both parameters are cancelled out such that no optimization can be made. However, it can still be used to estimate E5.17 and therefore to provide prediction to E5.21. In the two models of Tjørve (2012), none of the six parameters are cancelled out.

Interestingly, E5.21 is conceptually similar to E5.1 of the simplified maximum entropy method, only with occupancies instead of individuals. E5.1 is the summation over all abundance levels of the multiplication of two probabilities. The first is the probability that a species will have a certain number of individuals, and the second is the probability of a species with that number of individuals to occur in the sampled area. E5.21 has similar structure- it is the sum over all occupancy levels of the multiplication of two probabilities. The first is the probability of a randomly chosen species to be form a certain occupancy level, while the second is the probability of observing this species in the set of a samples. Furthermore, when dividing both sides of E5.21 with $\hat{\alpha}$, we end up with a term that equals $E(S|a, A, par)/\hat{\alpha}$, which is the ratio between gamma and alpha diversity. Therefore, E5.21 follows the multiplicative definition of beta diversity (Tuomisto 2010a, b).

5.6. Main features of the ‘UpScaling’ package

To R package ‘UpScaling’ includes the following functions:

1. `SimplMaxEntropyHarte`

This function takes as input the mean number of individual in a sample and the mean number of species per sample and uses Harte et al. (2009) simplified maximum entropy method to estimate the expected number of specie in consecutive doubling of the area. Based on the original code written by Han Xu (see details in Xu et al. 2012).

2. `ShenHe08Model`

This function takes as input presence-absence data with species as columns and sampled sites as rows. The aim is to predict the number of species if we had sampled additional samples. Thus, the function fits Shen and He (2008) model to a vector of number of samples specified by the user. Based on the original code written by Han Xu (see details in Xu et al. 2012).

3. `ToSpCurve`

This function takes as input a species by samples presence/absence data frame. The samples may be pre-allocated to different habitat categories or are allocated to a user defined number of categories using `kmeans` clustering. Then, the function applies species accumulation curves for all possible combinations of 1, 2, ..., k habitats. Next, the T-S Curve is built by fitting a model to the average total number of species in 1, 2, ..., k habitats. Here, the semi-logarithmic model, $S = c * \ln(X) + b$, is fitted with S being the number of species in X samples, while c and b are fitted parameters. Finally, the fitted curve is extrapolated to user defined number of samples (optional).

4. `PrepareNestedSacData`

The nested SAC models takes as input a data frame with the number of cells in the entire extent, the number of sampled cells, and the number of species observed in the set of sampled cells. This function aims to create the data for the nested SAC model. It takes as input the presence/absence data and an ASCII file, creates a raster from the ASCII, and link each sampled site in the presence/absence table with a cell in the raster object. Next, the function creates paths that start with a random or user defined starting cell and adds adjacent cells until all valid cells are covered. Along this path, information on the extent, number of sampled sites and number of species is kept and returned to the user for further analysis with `OptimNestedSac`. In addition, all information on the path itself is returned for further analysis and/or visualization.

5. `OptimNestedSac`

This function takes as input the values for the extent, number of samples, the observed number of species and the mean alpha diversity. Next, the function find the parameter values that minimize the sum of square differences between the observed and predicted number of species according to (up to) 22 different nested SAC models. Finally, for each model, the function predict the expected number of species if we had sampled all samples within the extent, as well as AICc, wAICc and model averaged predictions.

6. `PredictNestedSac`

This function predicts the expected number of species for a given extent and number of samples, given a user defined SAR model and parameters values.

7. `PlotOnePath`

A dynamic visualization function for the paths created by `PrepareNestedSacData`. The function takes as input the raster, `cell.info` and a single path. Then, the points are added several at a time according to the path.

8. `RandPAData`

The function takes as input the number of species, the number of sites and the number of occupancies and returns a species (columns) by site (rows) data frame with randomly distributed presence/absence data (1,0), while ensuring at least one occupancy is allocated to each site and species (i.e., none of the marginal sums is zero). The function can be used to construct null models around predicted species richness.

9. `RandomNestedSacData`

The function creates an ASCII with user defined dimensions. Each cell in the ASCII file is either valid (1) or invalid (0). All valid cells are connected to at-least one other valid cell. Next, the function creates a random presence/absence data frame, using the function `RandPAData` and assigns each sampled site to a valid cell. The number of sampled site can

be lower than the number of valid cells, and no valid cell contain more than 1 sampled site. The package returns the presence/absence table and the ASCII file along with a raster object and some information on each cell of the raster. Can be used to construct null models for the nested SAC method.

Installation, version control, maintenance, and future development.

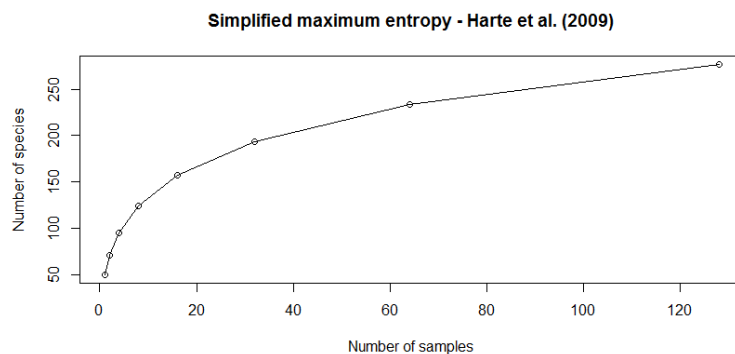
Currently, the R package is fully functioning and passed all CRAN checks. The package will be submitted to CRAN soon and thereafter it could be installed directly from CRAN. In addition, the package will be committed to R-Forge for version control. For now, the package can be installed directly from the zipped files *UpScaling_1.0.tar.gz* or *UpScaling_1.0.zip* (downloadable at <http://www.eubon.eu/documents/1/>). Regular maintenance of the package will be carried by Yoni Gavish (University of Leeds, gavishyoni@gmail.com). In the future, we intend to include additional upscaling models as described in milestone MS321, with the paired correlation function of Azaele et al. (2015) as the first priority. For now, we provide a Mathematica code for this method (*SI5.2-Azaele_2015_PCF.pdf*), downloadable at <http://www.eubon.eu/documents/1/>, courtesy of Dr. Sandro Azaele.

5.7. ‘UpScaling’ package – Tutorial

In this tutorial we will cover the main functions of the package *UpScaling*, following the order of methods in **Table 5.1**. We first start with the simplified maximum entropy method which is supplied in the package as the function `SimplMaxEntropyHarte`. The minimum input for the function is the number of individuals per sample, the number of species per sample and the number of area doublings. We recommend using the default setting of `lam.start = "auto"` that will automatically select the starting values for lambda ($\lambda_{\phi,2A}$ the Lagrange multiplier) at each area doubling. Although it still requires additional testing, it seems that using the optimal solution for lambda of one doubling as the starting value for lambda of the next area doubling results with good convergence. In addition, the default optimization method may return a warning message, yet it seems to converge well and the warning can be ignored. None-the-less, the function allows the users to control various aspects of the optimization, including the starting values of lambda, the optimization method and the lower and upper limits of the parameter. The output of the function is a data frame with columns for the total area (in units of 1 sample in the original, smallest scale), the total number of individuals per sample, the predicted number of species, the optimal value of lambda and the starting value of lambda.

```
# Run the simplified maximum entropy model
# the 'Nelded-Mead' warnings can be ignored
Harte.try <- SimplMaxEntropyHarte(ind.per.sam = 200, spe.per.sam = 50, num.doubling = 7)
names(Harte.try)
[1] "area"      "num.ind"    "num.species" "lam.fitted" "lam.in"
```

```
# plot the species area relationship
plot(Harte.try$area, Harte.try$num.species,
     main = "Simplified maximum entropy - Harte et al. (2009)",
     xlab = "Number of samples", ylab = "Number of species")
lines(Harte.try$area, Harte.try$num.species)
```



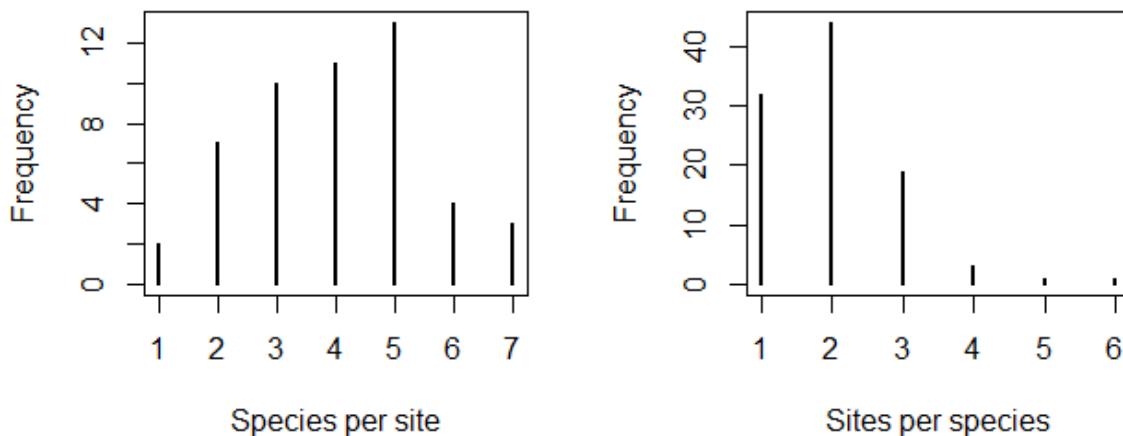
```
# Manually set starting values of lambda for each doubling
# ensure: length(lam.start) == num.doubling
Harte.try.2 <- SimplMaxEntropyHarte(ind.per.sam = 200, spe.per.sam = 50,
                                   num.doubling = 3,
                                   lam.start = c(0.1, 0.05, 0.01))

# similar results as with lam.start = "auto"
cbind(Harte.try[1:4, "num.species"], Harte.try.2[, "num.species"])
      [, 1]      [, 2]
[1, ] 50.00000 50.00000
[2, ] 70.42067 70.42067
[3, ] 95.23466 95.23466
[4, ] 124.26530 124.26530
```

Before continuing with the Shen and He (2008) model, we introduce the function `RandPAData`. This function allows the user to create random presence/absence data frames, with species as column and sites as rows. Presence is given by a value of 1 and absence as a value of 0. The P/A data frame is filled with user defined number of occupancies (the argument `mat.fill`), while ensuring that none of the rows (sites) or columns (species) have a sum of zero. Therefore, the minimum value of `mat.fill` is $\max(\text{num.species}, \text{num.site})$, while the maximum value is $\text{num.species} \times \text{num.sites}$.


```
# returns as error since mat.fill < max(num.species, num.sites)
set.seed(865)
PA.data <- RandPAData(num.species = 25, num.sites = 30, mat.fill = 29)
Error in RandPAData(num.species = 25, num.sites = 30, mat.fill = 29) :
  minimum value for mat.fill should be higher than max(num.species, num.sites) to
  ensure the matrix dimension
```

```
# create random data with the minimum fill of the matrix
set.seed(865)
PA.data <- RandPAData(num.species = 25, num.sites = 30, mat.fill = 30)
# the minimum and maximum row sums and the minimum columns sum
c(min(rowSums(PA.data)), max(rowSums(PA.data)), min(colSums(PA.data)))
[1] 1 1 1
sum(PA.data)
[1] 30
# random P/A with
set.seed(865)
PA.data <- RandPAData(num.species = 100, num.sites = 50, mat.fill = 200)
par(mfrow=c(1,2))
plot(table(rowSums(PA.data)), xlab="Species per site", ylab="Frequency")
plot(table(colSums(PA.data)), xlab="Sites per species", ylab="Frequency")
```



```
sum(PA.data)
[1] 200
```

The True and observed SOD method of Shen and He (2008) is covered by the function `ShenHe08Model`. The function requires a minimum of two input objects. First, the user must supply a species (columns) by sites (rows) data frame (`PA.data`) with 1 and 0 representing presence and absence, respectively. Here, we use the `PA.data` generated above by `RandPAData`. Second, the user should supply a numerical vector (`num.samples`) with the areas for which the function will predict the number of species. The only limitation for this vector is that its minimal value should not be smaller than the number of sites in the presence/absence table. In addition, the function also accepts a logical argument that if `TRUE`, the variance around the expected number of species will be estimated as well. We note that sometimes the variance returns `NAs`. Finally, for easing the drawing of a SAR curve, the user can also specify the area of a single sample, such that additional area columns is added to the output data frame.

```
# Run the model, the PA.data from before
ShenHeResults <- ShenHe08Model(PA.data = PA.data,
                               num.samples = c(seq(50, 140, 10),
                                                  seq(150, 1000, 50),
                                                  seq(1100, 1900, 100),
                                                  seq(2000, 10000, 1000)),
                               sample.area = 20)

# explore the results
names(ShenHeResults)
[1] "expected.spec" "par.values"

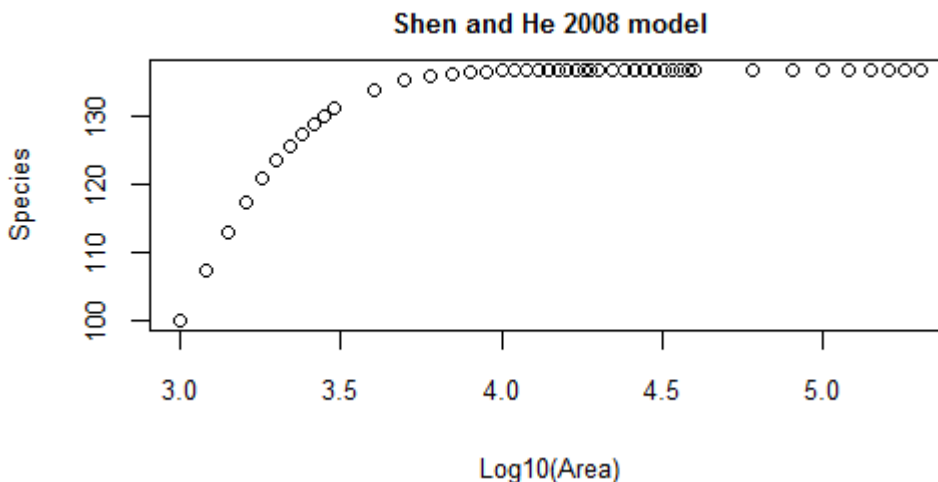
# note that the parameters values are independent of the num.samples
# further note that the likelihood term is given as approximate likelihood since
# several constants are ignored in the computations
ShenHeResults$par.values
$alpha
[1] 4.591587

$beta
[1] 149.3834

$approx.likelihood
[1] 131.6903

# in this case, convergence to 136 species
exp.spec <- ShenHeResults$expected.spec

par(mfrow= c(1,1), mar=c(4,4,2,1))
plot(log10(exp.spec$area), exp.spec$exp.spec, xlab = "Log10(Area)", ylab = "Species",
     main = "Shen and He 2008 model", cex.lab = 0.8, cex.axis = 0.8, cex.main = 0.8)
```



The T-S curve method of (Ugland et al. 2003) is given in the function `ToSpCurve`. It's very minimum requirement is the same presence / absence data frame as `ShenHe08Model`. In such a case, the function will use `kmeans` clustering to allocate the different samples to a user defined number of classes (default is 5 classes, set by `num.habitat`), prior to estimating the species accumulation curves and fitting the semi-logarithmic curve. In this case, the user can control several aspects of the clustering stage (see the detailed help files).

```

set.seed(865)
# create random presence/absence data
PA.data <- RandPAData(num.species = 25, num.sites = 50, mat.fill = 300)

# example without pre-allocated habitats and new.data
ts.min <- ToSpCurve(PA.data = PA.data,
                    new.data = NULL,
                    habitats = NULL,
                    num.habitat = 5)

# Note the function added an additional column in the PA.data with the assigned
# habitat for each sample
names(ts.min$PA.data)
[1] "Habitat.Cat" "Sp1" "Sp2" "Sp3" "Sp4" "Sp5"
[7] "Sp6" "Sp7" "Sp8" "Sp9" "Sp10" "Sp11"
[13] "Sp12" "Sp13" "Sp14" "Sp15" "Sp16" "Sp17"
[19] "Sp18" "Sp19" "Sp20" "Sp21" "Sp22" "Sp23"
[25] "Sp24" "Sp25"

```

However, we note here that the T-S curve method is more suitable for cases in which the class affiliation of each sample is known and pre-defined according to spatial location or habitat/land-cover/land-use. If the habitat identity of each sample is known, then the argument `habitats` should specify the column name of number in `PA.data` that holds the habitat information. The output of the method includes the original `PA.data` along with the 'nls' object for the semi-logarithmic curve. However, the more meaningful information is given in two data frames. The `SAC.info` data frame includes all the information from all species accumulation curve. It starts with a set of columns named after each class (habitat). The values of this columns is logical (TRUE or FALSE), signifying if the samples from this habitat were included in the focal SAC. This is followed by two columns, the first with the number of habitat in the SAC, and the second with the total number of samples in the SAC. The remaining columns gives the expected number of species for a given number of samples.

```

# example with pre-defined habitats
# create 6 habitats, labelled H1, H2, ... H6
PA.data$Habitat <- paste("H", sample(1:6, 50, replace = TRUE), sep="")

# run the TS model
ts.full <- ToSpCurve(PA.data = PA.data,
                    habitats = "Habitat")

names(ts.full)
[1] "PA.data" "SAC.info" "TS.semi.log" "TS.data" "call"

# the object with the semi-logarithmic nls
class(ts.full$TS.semi.log)
[1] "nls"

# all the info on the species accumulation curves
# a row for each SAC, info on the habitats included in the SAC
# the number of habitats and the total number of samples in the combination,
# the number of species for each step in the SAC
ts.SAC.info <- ts.full$SAC.info

```

```

# the main data frame with the T-S curve data this are the values on which the semi-
# logarithmic curve was fitted the mean number of species in combination of 1, 2, ...6
# habitats additional information on the number of combinations, number of samples
# standard deviation and standard error + the predicted values of the semi-logarithmic
# curve
ts.full.data <- ts.full$TS.data

names(ts.full.data)
[1] "data.source" "num.habitats" "num.combin" "num.samples" "mean.species"
"sd.species"
[7] "se.species" "predicted.sp"

```

The `TS.data` data frame contains the summary of the `SAC.info` data frame that is required for fitting the semi-logarithmic total species curve. For each number of habitats (the column `num.habitats`) the table includes information on the mean, standard deviation and standard error of the number of species observed. Other columns state the number of different combination of the focal number of habitats and the number of samples for which the mean was taken (the minimum number of samples in any of the combinations, see section 5.4). The first column (`data.source`) distinguishes between the data that originated from the `PA.data` and `new.data` (see below). Finally, the `predicted.sp` gives the predicted number of species for the given number of samples according to the fitted semi-logarithmic curve.

An important additional option of the `ToSpCurve` function is given by `new.data` - a numeric vector of positive integers specifying the number(s) of samples to which the T-S Curve should be extrapolated. The predicted values for the `new.data` are based on the `predict` method for the 'nls' object (the semi logarithmic curve). The values are added to the `TS.data` data frame and are labelled as `new.data` under the `data.source` column.

```

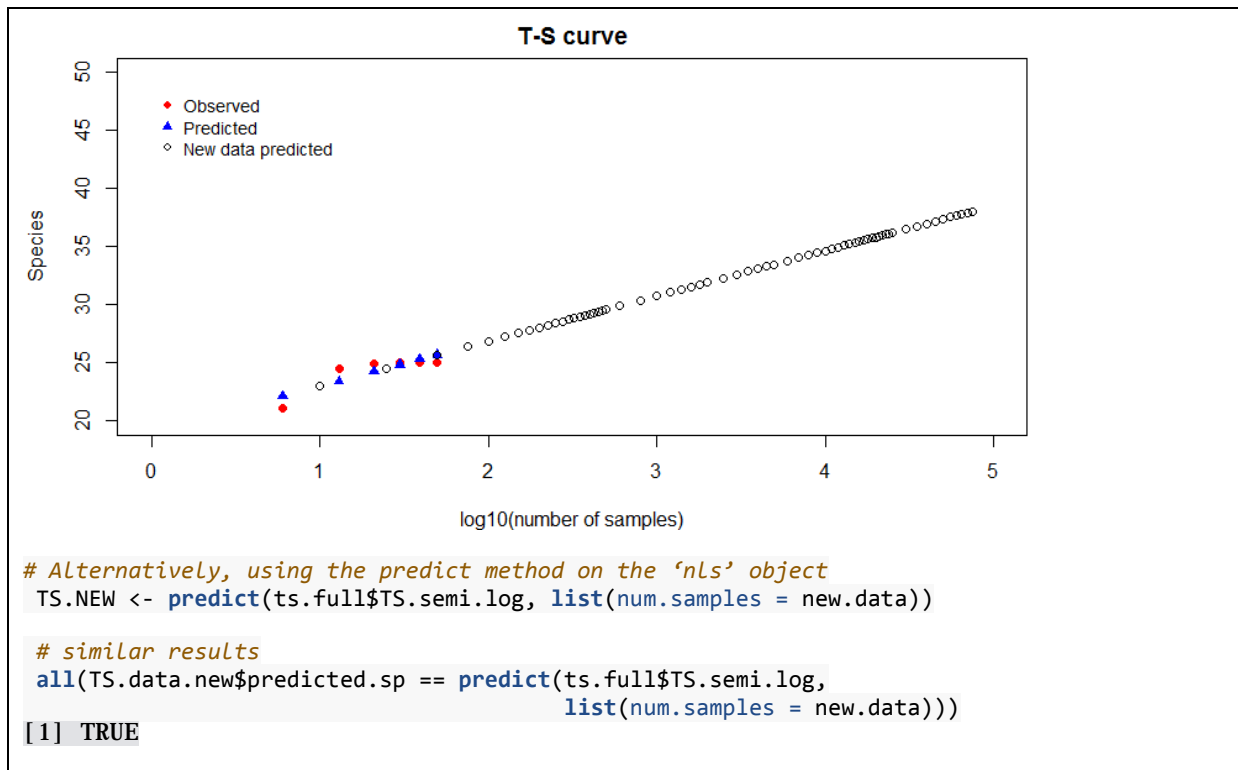
# the T-S curve with new.data
# the number(s) of samples to which the TS-Curve should be extrapolated
new.data <- c(10, seq(25, 500, 25), seq(600, 2000, 200), seq(2500, 5000, 500),
             seq(6000, 25000, 1000), seq(30000, 75000, 5000))

# run the T-S curve analysis
ts.new <- ToSpCurve(PA.data = PA.data, new.data = new.data, habitats = "Habitat")

# same as before with the additional predictions for the new.data
# within the TS.data data frame. Distinguished from the original data as
# "new.data" under the column "data.source"
ts.new.full <- ts.new$TS.data
TS.data.new <- ts.new.full[ts.new.full[, "data.source"] == "new.data" , ]

# plot the SAR
plot(c(0,5),c(20,50),type="n", xlab="log10(number of samples)",
     ylab="Species", main = "T-S curve")
points(log10(ts.full.data$num.samples), ts.full.data$mean.species, pch=16, col="red")
points(log10(ts.full.data$num.samples), ts.full.data$predicted.sp, pch=17, col="blue")
points(log10(TS.data.new$num.samples), TS.data.new$predicted.sp, pch= 1, col="black")
legend(0,49, c("Observed ", "Predicted ", "New data predicted "), pch =c(16, 17, 1),
      col = c("red", "blue", "black"), bty = "n", cex =0.9)

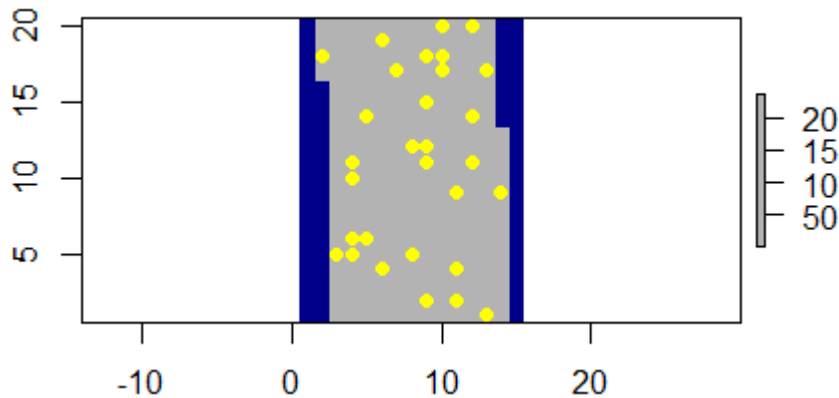
```



Before starting with examples for the nested SAC method, we introduce the function `RandomNestedSacData`. This function is used to create random data to test and explore the nested SAC method. The user can control the number of columns and rows in the landscape, as well as the number of sampling sites, the number of species observed in the sampled sites and the total number of occurrences in the presence/absence table (generated by `RandPAData`). The function returns a list with 4 objects, including the presence/absence data, an ASCII file, a raster file and an additional information on each cell of the raster. The main objective for this function, along with the `RandPAData` function, is to create null models that gives the expected number of species under 3 different scenarios:

1. *Both species occurrence in sampled sites and locations of sampled sites are random:*
Use `RandomNestedSacData` as is.
2. *Species occurrence in sites is random, but sampled sites locations is fixed by users:*
Use `RandPAData` to create the presence absence data, with each sample in the randomly generated data taking the spatial location of one real sample.
3. *Species occurrence in sites is fixed, but sampled sites locations are random*
Use `RandomNestedSacData` to generate a random ASCII file, but use the original presence / absence data.

```
set.seed(350)
# create random data
rand.input <- RandomNestedSacData(num.col = 15, num.row = 20, num.species = 95,
                                  num.sites = 30, mat.fill = 180)
```



```
names(rand.input)
[1] "PA.data" "asc.out" "raster.out" "cell.info"

# made with RandPADData, additional columns for x,y coordinate + site ID.
PA.data <- rand.input$PA.data

# the ASCII is of class "asc" from the 'SDMTools' package
# the input ascii for PrepareNestedSacData should have similar structure
# valid cells (e.g., land) given by 1 and invalid cells (e.g., sea) given by 0
class(rand.input$asc.out)
[1] "asc"

# the raster is of class "RasterLayer" from the 'raster' package
class(rand.input$raster.out)
[1] "RasterLayer"
attr(,"package")
[1] "raster"

# information on each cell of the raster- the cell.num covering all the cells
# and the cell.ID with NAs for invalid cells - i.e. - cells outside the area of
# interest that are given as 0 in the ascii file
cell.info <- rand.input$cell.info
```

The ASCII and the presence/absence table are the main input data needed for a nested SAC model. The ASCII (argument `asc.in`) should be rectangular and may contain only 0 or 1 values. The 0 values represent area that are considered invalid for the community of interest and that should not be included in the nested SAC analysis. For example, when focusing on terrestrial organisms on an island, 1 will represent cells inside the island and 0 will represent cells in the sea. We refer the user here to the upgraining vignette of section 6 of this deliverable for an important discussion regarding the creation of gridded cells from point data. The presence/absence data frame (`PA.data`) should be a data frame with the first 3 columns representing the X coordinate, the Y coordinate and a unique site ID (at this order), followed by additional columns for each species. Each row in the `PA.data` data frame is for one sampled site, with 1 and 0 representing presence or absence, respectively. It is important to note that not all cells in the `asc.in` need to be sampled, but any cell in `asc.in` can

only contain a single sampled site. The function `PrepareNestedSacData` will return an error if more than two sampled sites are located within the same cell. For a good example of the required input data, we refer the users to the objects generated by `RandomNestedSacData`.

Once the input data is ready, the function `PrepareNestedSacData` can be used to prepare the data for a nested SAC analysis with `OptimNestedSac`. The preparation of data involves creating the information on the extent, number of samples and number of species. This is done by selecting a random or user defined starting point, and adding adjacent cells one by one, until all valid cells are encountered. After any addition of a cell, the extent, number of samples and number of species is updated. The extent always increases by 1. If the added cell is not sampled, the number of samples and number of species remains unchanged. If the added cell was sampled, then the number of samples increases by 1, and the number of species is updated. The function allows three different algorithms to grow a path from a starting location (`grow.method`, see below). The starting location (`start.cell`) can be supplied by the user (as x,y coordinates) or randomly chosen (default). The default is to grow one path at a time, yet multiple paths can be grown by adjusting the value of `num.paths`.

The output of the function for a single path include a list with 5 objects, including the `PA.data`, and `cell.info` data frames as in `RandomNestedSacData`, only with additional information on the linkage between the samples and the raster. The function also returns a raster object with the `cell.ID` as values for valid cells and NA for invalid cells. The `cell.num` in the raster always start from the top right corner and advances row by row until the bottom left corner, covering both valid and invalid cells. The two important data frame for the nested SAC analysis are the `path.info` and `nested.SAC.object` data frames. The `path.info` includes all the information on the cells following the order by which they were encountered. This may be used for the dynamic plotting of `PlotOnePath`, or for extracting the ‘true’ data if information on all the cells is available. The `nested.SAC.object` contains the number of samples, extent and number of species that were estimated and updated each time a new sample was encountered along the path. This data frame can be plugged directly to the `OptimNestedSac` function.

```
# Single path with the default path growing method. see below for other options
Nested.Sac.Data <- PrepareNestedSacData(PA.data = rand.input$PA.data,
                                       asc.in   = rand.input$asc.out,
                                       grow.method = "min.dist.start")

# the output is a list with 5 object
names(Nested.Sac.Data)
[1] "PA.data" "raster.out" "cell.info" "path.info" "nested.SAC.object"

# PA.data- same as above with additional columns linking each sample with
# a raster cell
names(Nested.Sac.Data$PA.data)[1:8]
[1] "cell.num" "cell.ID" "X.cord" "Y.cord" "site.ID" "Sp1" "Sp2" "Sp3"
```

```

# same as above, with additional information on the raster cell.num and cell.ID
cell.info <- Nested.Sac.Data$cell.info

# the raster object with NAs for invalid cells and cell.ID for valid cells in
# the value field
raster.out <- Nested.Sac.Data$raster.out
sum(is.na(raster.out@data@values))
[1] 63
sum(!is.na(raster.out@data@values))
[1] 237

# the information for the single path. The first row is for the starting location of the
# path. each row is a newly added valid cell to the path,
path.info <- Nested.Sac.Data$path.info
names(path.info)
[1] "cell.num" "cell.ID" "X.cord" "Y.cord" "site.ID"

# the number of samples, extent and number of species to be passed to OptimNestedSAC
names(Nested.Sac.Data$nested.SAC.object)
[1] "num.sample" "extent" "num.spec"

# Creating figure 5.1 below
par(mfrow = c(2,2), mar = c(2.5, 4, 2.5, 2.5))
# plot the raster, all the valid and invalid cells, and the sampled cells
plot(raster.out, colNA = "gray", xlab = "X coordinate",
     ylab = "Y coordinate", main = "a) raster and samples")
points(cell.info[!is.na(cell.info$cell.ID), "X.cord"],
       cell.info[!is.na(cell.info$cell.ID), "Y.cord"], cex = 0.5)
points(Nested.Sac.Data$PA.data$X.cord, Nested.Sac.Data$PA.data$Y.cord,
       pch=16, col="red", cex = 0.5)

nested.SAC.object <- Nested.Sac.Data$nested.SAC.object

# the extent at which each sample was added to the path
plot(nested.SAC.object$num.sample, nested.SAC.object$extent,
     xlab = "Number of samples", ylab = "Extent",
     main = "b) extent vs. num samples in a path")

# the accumulation of species with extent and number of samples
plot(nested.SAC.object$extent, nested.SAC.object$num.spec,
     xlab = "Extent", ylab = "Number of species",
     main = "c) species vs. extent in a path")

plot(nested.SAC.object$num.sample, nested.SAC.object$num.spec,
     xlab = "Number of samples", ylab = "Number of species",
     main = "d) species vs. num.samples in a path")

```

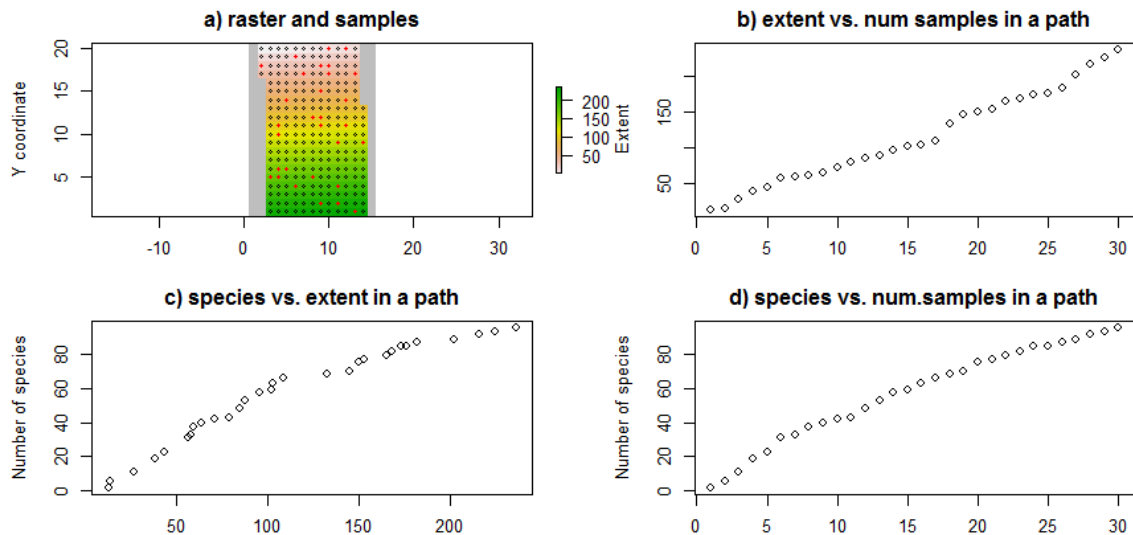



Figure 5.1: Plotting the output of `PrepareNestedSacData` for a single path. **a)** the raster with invalid cells in grey, and valid cells as a gradient according to `cell.num`. Sampled cells are given in red and unsampled as black symbols. **b)** the extent at which each new sample was encountered along the path. **c)** The accumulation of species with extent. **d)** The accumulation of species with the number of samples. The figure is created in the code above.

The `PrepareNestedSacData` function allows choosing one of three different algorithms to grow a path. All three algorithm add adjacent cells one by one until all valid cells are encountered. They differ in the creation of the list of adjacent cells at each step, and allow a continuum between a clumped path with no holes to a rather dispersed path with many holes. The three options include:

1. `'min.dist.start'` -

Always select a cell from the list of cells with minimum distance from the starting cell. This method results with the most clumped path that grows outwards from the starting cell. The method does not results with 'holes' in the distribution of travelled cells.

2. `'random.adj.4'` -

The list of adjacent cells contain all the cells adjacent to all cells that were already travelled. Adjacent cells are based on 4 neighbouring cells. A random cell is selected from the list of adjacent cells, thereafter, the list is updated. The procedure continues until all cells are added. The method may result with 'holes' in the distribution of travelled cells. See the 'rook' (or 4) option in the `adjacent` function of the `raster` package for details on adjacency.

3. `'random.adj.8'` -

Same as above, only with 8 neighbouring cells. See the 'queen' (or 8) option in 'adjacent' function of the `raster` package for details. This method results in a very dispersed path that may contain 'holes' in the distribution.

To illustrate the difference between the three growing methods, we use the `PlotOnePath` function, which we advise the users to try in the R console. This function takes as input the

`raster.out` object and the `cell.info` data frame generated by `PrepareNestedSacData`. In addition, the function takes the `cell.num` information for a single path. Then, the raster is plotted and the cells of the path are added few at a time to the plot. The user can control the size and colours of the different cells. In addition, the user can control the number of cells added to the plot in each step and the length of the pause between each step (with `pause.sec`). We recommend using the default of `'min.dist.start'`.

```
par(mfrow = c(1,1), mar = c(1, 1, 1, 1))
# note the differences between the 3 path growing methods in spread and holes
# All are starting from the same cell (in red)
# The 'min.dist.start' path growing method - very clumped, no holes
set.seed(250) # same starting location
Nested.Sac.Data.1 <- PrepareNestedSacData(PA.data = rand.input$PA.data,
                                         asc.in  = rand.input$asc.out,
                                         grow.method = "min.dist.start")

# Plot a single path
PlotOnePath(raster.out = Nested.Sac.Data.1$raster.out,
            cell.info   = Nested.Sac.Data.1$cell.info,
            path.cell.num = Nested.Sac.Data.1$path.info$cell.num,
            cex = 1, every.n.cells = 15)

# The 'random.adj.4' path growing method - more dispersed, some holes
set.seed(250)
Nested.Sac.Data.2 <- PrepareNestedSacData(PA.data = rand.input$PA.data,
                                         asc.in  = rand.input$asc.out,
                                         grow.method = "random.adj.4")

PlotOnePath(raster.out = Nested.Sac.Data.2$raster.out,
            cell.info   = Nested.Sac.Data.2$cell.info,
            path.cell.num = Nested.Sac.Data.2$path.info$cell.num,
            cex = 1, every.n.cells = 15)

# The 'random.adj.8' path growing method - very dispersed, many holes
set.seed(250)
Nested.Sac.Data.3 <- PrepareNestedSacData(PA.data = rand.input$PA.data,
                                         asc.in  = rand.input$asc.out,
                                         grow.method = "random.adj.8")

PlotOnePath(raster.out = Nested.Sac.Data.3$raster.out,
            cell.info   = Nested.Sac.Data.3$cell.info,
            path.cell.num = Nested.Sac.Data.3$path.info$cell.num,
            cex = 1, every.n.cells = 15)
```

As stated above, the default of the `PrepareNestedSacData` function is to create a single path, which can be plugged directly to the `OptimNestedSac` function for further analysis (see below). However, we do not recommend basing the entire inference on a single path, since the resulting path may depend on the starting point. We recommend averaging the predictions of `OptimNestedSac` for multiple paths. There are two main analytical flows to follow such an analysis. Assuming we wish to average the results of Z paths, we can run a loop that runs z times and in each turn creates a path, fit it with a set of nested SAC models, and keeps the results of the fitting.

This will be followed by averaging of the z results. However, each time the function `PrepareNestedSacData` is called, it creates a raster from the ASCII file and runs various other time consuming stages that are general for all paths. Therefore, a more efficient analytical flow is first to create all the Z paths in a single call to the function `PrepareNestedSacData`, and then run a loop that fit every path with a set of nested SAC models.

The `PrepareNestedSacData` function allows the user to follow any of this options by setting the number of paths (`num.paths`). However, it is impractical to create to data frames for each path, as the length of the return list will be quite long. Therefore, the output list changes when `num.paths > 1`. Instead of two data frames for each path, the function create a single data frame for each property and adds a column for each path:

- `'path.cell.num'` - Data frame with the `'cell.num'` of each cell (rows) along each path (columns).
- `'path.cell.ID'` - Data frame with the `'cell.ID'` of each cell (rows) along each path (columns).
- `'path.cell.X'` - Data frame with the X coordinate of each cell (rows) along each path (columns).
- `'path.cell.Y'` - Data frame with the Y coordinate of each cell (rows) along each path (columns).
- `'path.cell.site.ID'` - Data frame with the site ID (from the original `PA.data`) of each cell (rows) along each path (columns).
- `'path.extent'` - Data frame with the extent at which each new sample was reached in each path. The first column contains the number of samples (`num.samples`) followed by a column for each path.
- `'path.num.spec'` - Same as `'path.extent'` only with the number of species in the set of samples already encountered along the path.

For the first five data frame, column i contains the information for path i . for the two last data frame, the first column contains information on the number of samples, such that the information for path i is located at column $i + 1$.

```
# example with several paths
Nested.Sac.Data <- PrepareNestedSacData(PA.data = rand.input$PA.data,
                                         asc.in  = rand.input$asc.out,
                                         num.paths = 5)

names(Nested.Sac.Data)
[1] "PA.data"      "raster.out"    "cell.info"     "path.cell.num"
[5] "path.cell.ID" "path.cell.X"   "path.cell.Y"   "path.cell.site.ID"
[9] "path.extent"  "path.num.spec"

# path.info and nested.SAC.object no longer returned
# instead a separate data frame for each column of path.info
```

```

# order of columns are from the first path to the last
path.cell.num <- Nested.Sac.Data$path.cell.num

# similar for the nested.SAC.object, only the number of samples is not reuturn
# and is added as the first column in path.extent and path.num.spec.
# order of columns are from the first path to the last
path.extent <- Nested.Sac.Data$path.extent
path.num.spec <- Nested.Sac.Data$path.num.spec

# to create an object to use as input in OptimNestedSAC
#example for path i (here, i=3)
i=3
nested.SAC.object <- as.data.frame(cbind(path.extent[, 1],
                                         path.extent[, i + 1],
                                         path.num.spec[, i + 1]))
names(nested.SAC.object) <- c("num.samples", "extent", "num.species")

```

Once the data for analysis is created, the function `OptimNestedSac` is the main function that fits any of the 22 different nested SAC models (**Table 5.2**). The function expect as input a data frame with at least three columns, for the number of samples, extent and number of species. The user also needs to specify the column names (or numbers) that holds these three inputs and to provide the mean alpha diversity. The main output of the function is the predicted number of species in a set of samples taken within the extent according to each of the specified nested SAC models (when `predict.part = TRUE`), as well as the predicted number of species if all the extent was sampled (when `predict.full = TRUE`). In addition, the function provides the predicted parameter values, model log likelihood, AICc, delta AICc and wAICc. The function also estimates the mean over all models that converged for the partial and full number of species using the wAICc as weights. Finally, the function returns an object of class `'nls.lm'` for each chosen nested SAC model.

The optimization is based on the *Levenberg-Marquardt* optimization algorithm as implemented in the `nls.lm` function of the `minpack.lm` R package. We chose this optimization algorithm since our experience suggest it can converge to optimal solution even when the starting parameters are far from the target. Various aspects of the optimization can be controlled by the `control` argument (we refer the user to the help files of `nls.lm.control` for more details). However, we note, that controlling the starting values of the parameters, as well as their lower or upper limits is only possible when calling a single SAR function. In such a case, the user should provide a named list for the starting values, and a vector for the lower and upper values. Order of the limits in the vectors should be according to the order of the named list. We further suggest increasing the maximum number of iterations (see the example below).

In most cases, more than one nested SAC model is fitted to the same data. If this case, the default starting parameters specified in **Table 5.3** are used. To call more than a single model, the user should provide a vector of codes for the `sar.model` argument, with the codes given in **Table 5.3**. Similar to running the function with a single SAR model, the function will return an object of class `'nls.lm'` for each SAR model. In addition to these objects, the main output of the function is summarized in

two data frames. The first, named `sar.model.full`, will contain for each nested SAC model (row) information on the the values of each parameter, the deviance, log likelihood, AICc, delta AICc, model likelihood and AICc weight, based on the `aictabCustom` function of the `AICcmodavg` package. We note here, that models that fail to converge will have NA in various columns. This model will not be used when estimating the model selection information and the model averaging.

```
# the mean alpha diversity: 180/30 = 6
mean.alpha <- sum(PA.data[, -(1:3)])/ 30

# single SAR model Cumulative beta-p with 3 parameters, unbounded
optim.out <- OptimNestedSac(object = nested.SAC.object,
                           extent = 2, num.sample = 1, num.spec = 3,
                           mean.alpha = mean.alpha,
                           sar.model = c("CuB"),
                           start.par = list(z=0.5, b=0.5, d=0.5),
                           control = minpack.lm::nls.lm.control(maxiter = 1000))

optim.out$sar.model.full[,c(4:6,12)]
      z      b      d    AICc
20  8.784586 31.27959 -3.910908 97.25806

# setting a lower/upper bound to parameters values
# Note the usage of Inf and -Inf if only some of the parameters are to be bounded
optim.out.2 <- OptimNestedSac(object = nested.SAC.object,
                              extent = 2, num.sample = 1, num.spec = 3,
                              mean.alpha = mean.alpha,
                              sar.model = c("CuB"),
                              start.par = list(z=0.5, b=0.5, d=0.5),
                              upper = c(8, 30, Inf),
                              lower = c(-Inf, 0, -2),
                              control = minpack.lm::nls.lm.control(maxiter = 1000))

optim.out.2$sar.model.full[,c(4:6,12)]
      z      b      d    AICc
20    8     30     -2  191.1173
```

Table 5.3: the code and name of each nested SAC model, along with the number, starting values and names of the model parameters.

Code	Name	#	z	b	d	n	C1	C2
Pow	Power	1	0.5	-	-	-	-	-
EP1	Extended Power 1	2	-	-0.5	2	-	-	-
EP2	Extended Power 2	2	-	0.5	0.5	-	-	-
Pe1	Persistence Model 1	2	-	10	10	-	-	-
Pe2	Persistence Model 2	2	-	0.5	10	-	-	-
Log	Logarithmic Model	0	-	-	-	-	-	-
Kob	Kobayashi logarithmic	1	-	0.5	-	-	-	-
NeE	Negative Exponential	1	-	0.5	-	-	-	-
CoL	Common Logistic	2	-	10	10	-	-	-
ArL	Archibald Logistic	2	-	-0.5	2	-	-	-
Gom	Gompertz	2	-	-0.5	2	-	-	-

Code	Name	#	z	b	d	n	C1	C2
EVF	Extreme-Value function	2	-	-0.5	2	-	-	-
Mon	Monod	1	-	1000	-	-	-	-
AsR	Asymptotic regression	1	-	-	-0.5	-	-	-
Rat	Rational Function	2	0.5	-	2	-	-	-
ChR	Chapman-Richards	2	-	0.5	2	-	-	-
CuW	Cumulative Weibull	2	-	0.5	-0.5	-	-	-
MMF	Morgan-Mercer-Flodin	2	-	10	1000	-	-	-
Lom	Lomolino Function	2	-	100	100	-	-	-
CuB	Cumulative Beta-P	3	0.5	0.5	0.5	-	-	-
Tm1	Tjorve (2012)-model 1	6	0.5	0.5	0.5	0.5	0.5	0.5
Tm2	Tjorve (2012)-model 2	6	0.5	0.5	0.5	0.5	0.5	0.5

The second data frame is the input object data frame, with additional columns for the predicted partial (if `predict.part = TRUE`) and/or full (if `predict.full = TRUE`) number of species for each model. These are given in columns named `part.Code` and `full.Code`, respectively (e.g., for the power model with the code `Pow`: `part.Pow` and `full.Pow`). Finally, if the user set the argument `mod.avg` to `TRUE`, then additional columns will be added with the model averaged predicted number of species, with the AICc weights being used to weight the predicted value of each model. That is, if the `Pow` and `CuB` models predicted for a given area 20 and 30 species and their AICc weights are 0.75 and 0.25, respectively, the model averaged prediction will be 22.5. These predictions appear under `part.average` and `full.average` in the object data frame.

```
# run the optimization on two models 'Pow' and 'CuB'
optim.out <- OptimNestedSac(object = nested.SAC.object,
                           extent = 2, num.sample = 1, num.spec = 3,
                           mean.alpha = mean.alpha,
                           sar.model = c("Pow", "CuB"),
                           control = minpack.lm::nls.lm.control(maxiter = 1000))

names(optim.out)
[1] "Pow" "CuB" "sar.model.full" "object"
# the nls.lm object for the each model:
class(optim.out$Pow) # The power model
[1] "nls.lm"
class(optim.out$CuB) # Cumulative Beta-P
[1] "nls.lm"

# information on each model
sar.model.out <- optim.out$sar.model.full
sar.model.out[,c("code", "AICcWt")]
  code AICcWt
1  Pow 0.9186344
20 CuB 0.0813656

# Predicted species richness + plotting
sar.object <- optim.out$object

# Creating Figure 5.2 below
par(mfrow = c(2,2), mar=c(4, 4, 3, 3))
## 1 Pow num.sample
plot(c(0,30),c(0,200),type="n", xlab="num.sample",
     ylab="Species", main = "a) Pow: Species vs. num.samples")
points(sar.object$num.sample, sar.object$num.spec, pch= 2, col="black")
lines(sar.object$num.sample, sar.object$part.Pow, col="blue")
lines(sar.object$num.sample, sar.object$full.Pow, col="red")

## 2 Pow extent
plot(c(0,250),c(0,200),type="n", xlab="extent",
     ylab="Species", main = "b) Pow: Species vs. extent")
points(sar.object$extent, sar.object$num.spec, pch= 2, col="black")
lines(sar.object$extent, sar.object$part.Pow, col="blue")
lines(sar.object$extent, sar.object$full.Pow, col="red")

## 3 CuB num.sample
plot(c(0,30),c(0,200),type="n", xlab="num.sample",
     ylab="Species", main = "c) CuB: Species vs. num.samples")
points(sar.object$num.sample, sar.object$num.spec, pch= 2, col="black")
lines(sar.object$num.sample, sar.object$part.CuB, col="blue")
lines(sar.object$num.sample, sar.object$full.CuB, col="red")
```

```
## 4 CuB extent
plot(c(0,250),c(0,200),type="n", xlab="extent",
     ylab="Species", main = "d) Cub: Species vs. extent")
points(sar.object$extent, sar.object$num.spec, pch= 2, col="black")
lines(sar.object$extent, sar.object$part.CuB, col="blue")
lines(sar.object$extent, sar.object$full.CuB, col="red")
```

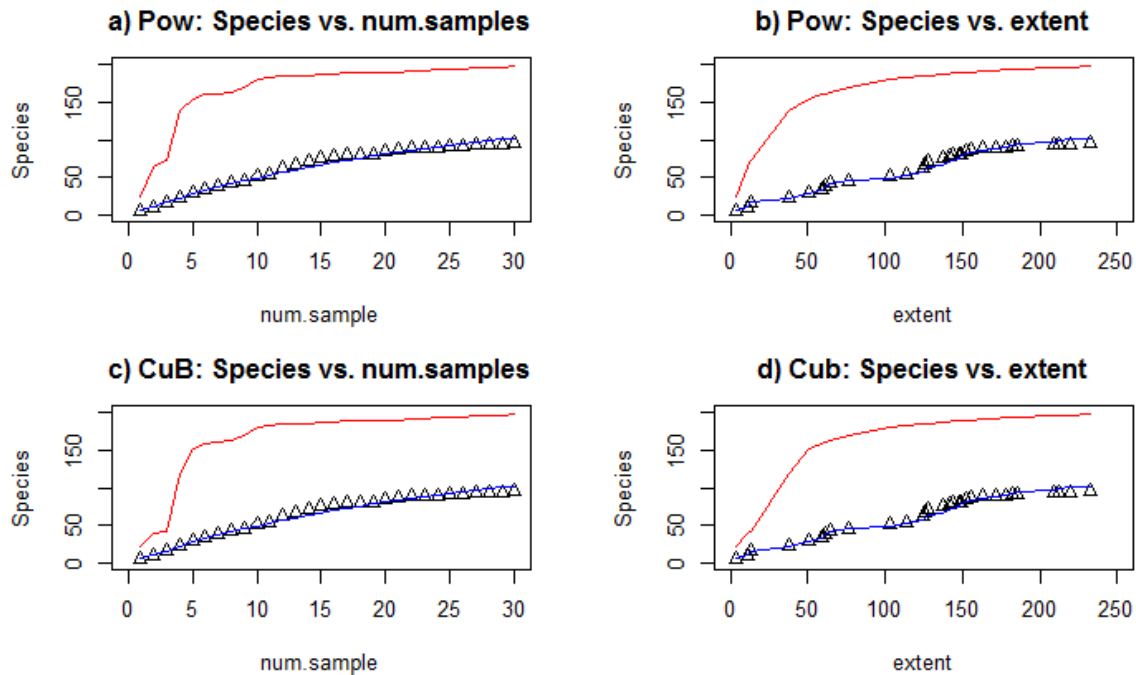


Figure 5.2: The observed (black triangle), predicted partial (blue line), and predicted full (red line) number of species against the number of samples (*a* and *c*) or extent (*b* and *d*) for the power (*a* and *b*) and Cumulative beta-P (*c* and *d*) nested SAC models. Note, that the full nested SAC returns a smooth curve when plotted against the extent, while the predicted partial take a stairstep shape. The opposite pattern emerges when the number of species is explored against the number of samples, with a smooth curve that fits the data nicely for the predicted partial and an irregular stairsteps curve for the predicted full. The figure is created in the code above.

The final function of the package is `PredictNestedSac`, which allows the user to predict the partial or full number of species according to a nested SAC model with user defined values for each parameter. The function takes similar input as `OptimNestedSac` (a data frame with columns for extent, number of samples and number of species + alpha diversity), along with a named list for the parameters. The function can only run on a single nested SAC model at a time. To predict the partial number of species, set the arguments for `num.sample` (the column name or number with the number of samples) to a different column from the `extent` column. To predict the full number of species (i.e., when the number of samples equal the extent), set the column for `num.sample` to the same column as the `extent`.

```
# Power model - incomplete sampling with z = 0.9  
# different column for extent and number of samples  
predcit.Pow <- PredictNestedSac(object = nested.SAC.object, extent = 2,  
                                num.sample = 1, mean.alpha = 10,  
                                sar.model = "Pow", par= list(z = 0.9))  
  
# Power model - complete sampling with z = 0.9  
# The same column for extent and number of samples  
predcit.Pow.2<- PredictNestedSac(object = nested.SAC.object, extent = 2,  
                                num.sample = 2, mean.alpha = 10,  
                                sar.model = "Pow", par= list(z = 0.9))  
  
# model with more than 1 parameters  
predcit.EP2<- PredictNestedSac(object = nested.SAC.object, extent = 2,  
                                num.sample = 1, mean.alpha = 10,  
                                sar.model = "EP2", par= list(b = 0.9, d = 0.5))
```


6. DOWNSCALING OCCUPANCY

6.1. Introduction

In order to assess and manage the status of a species we need to know the abundance of individuals in their population(s) and their changes over time. For the vast majority of species this information is not available. However, one important proxy of abundance is the area occupied by the species (Hartley and Kunin 2003). For example, the area of occupancy (AOO) is a little-used measure of conservation status in the IUCN red list for plants as the majority of endangered species have <10 records yet over 500 records would be required to obtain a non-threatened status at the IUCN recommended grain size of 4 km² (IUCN 2014). Although easier to estimate than true abundance, the difficulty in estimating AOO lies in the extensive sampling required across the full range of the species at a grain size sufficiently fine to give meaningful estimates (and this grain size may vary with taxa, habitat or region making a single grain size inappropriate). For the majority of species this is still impractical or unfeasible at these grain sizes. However, as we estimate occupancy at increasing grain sizes we increase our confidence in our presence-absence predictions. Such coarse-grain atlas data, generally generated from opportunistic recording over extended periods of time are much more widely available, however, at such coarse grain sizes we also lose resolution in our status estimates as occupancy rates at large grain sizes are less closely correlated with true abundance (Hartley and Kunin 2003).

A solution is to employ the occupancy-area relationship (OAR); that is the area occupied by a species increases as grain size increases (Kunin 1998). If the relationship can be described for occupancy at these coarser grain sizes, where confidence is high, then we can extrapolate the occupancy predictions to the fine grain sizes that are more closely related to the true abundance, distribution and conservation status.

Many models have been proposed to model this geometric relationship, and it appears that no one model consistently provides the best predictions (Azaele et al. 2012, Barwell et al. 2014). The ‘downscale’ R package provides functions for ten commonly applied models, along with functions for preparing coarse-scale data, plotting results, and an ensemble method for running multiple models and averaging their predictions. This greatly increases the accessibility and usability of the models to prospective users, facilitating their application for more species and data sets across Europe.

6.2. The ‘downscale’ package structure

Ten downscaling models are available (Nachman, power law, logistic, poisson, negative binomial, generalised negative binomial, improved negative binomial, finite negative binomial, Thomas and Hui models). Code was based upon preliminary R code written and kindly provided by L. Barwell and C. Hui. **Table 6.1** presents the original equations for eight of the models together with the codes implemented in R. The Thomas model and Hui model involve multiple equations – more detail on

each of these models can be found in the help files (*SI6.1-downscale.pdf*, downloadable at <http://www.eubon.eu/documents/1/>), the source code and the supplementary material of Barwell et al. (2014).

Table 6.1: Details of eight of the downscaling models, along with the original model equations translated in to R code. The R code equations are log transformed as model fitting is carried out in log-log space.

Model	Code	R code	Model equation
Nachman	"Nachman"	$\log(1 - \exp(-C * \text{area}^z))$	$1 - e^{-cA^z}$
Power law	"PL"	$\log(C * \text{area}^z)$	cA^z
Logistic	"Logis"	$\log((C * (\text{area}^z)) / (1 + (C * (\text{area}^z))))$	$\frac{cA^z}{1 + cA^z}$
Poisson	"Poisson"	$\log(1 - (\exp(-\lambda * \text{area})))$	$1 - e^{-\gamma A}$
Negative binomial	"NB"	$\log(1 - (1 + (C * \text{area}) / k)^{-k})$	$1 - \left(1 + \frac{\gamma A}{k}\right)^{-k}$
Generalised negative binomial	"GNB"	$\log(1 - (1 + (C * \text{area}^z) / k)^{-k})$	$1 - \left(1 + \frac{cA^z}{k}\right)^{-k}$
Improved negative binomial	"INB"	$\log(1 - ((C * \text{area}^{(b-1)})^{((r * \text{area}) / (1 - C * \text{area}^{(b-1))})))$	$1 - [c(\gamma A)^{b-1}]^{\frac{\gamma A}{1 - c(\gamma A)^{b-1}}}$
Finite negative binomial	"FNB"	$\log(1 - ((\text{gamma}(W + ((\text{extent} * k) / \text{area}) - k) * \text{gamma}(\text{extent} * k) / \text{area}) / (\text{gamma}(W + ((\text{extent} * k) / \text{area})) * \text{gamma}(((\text{extent} * k) / \text{area}) - k)))$	$1 - \frac{\Gamma\left(N + \frac{A_0 k}{A} - k\right) \Gamma\left(\frac{A_0 k}{A}\right)}{\Gamma\left(N + \frac{A_0 k}{A}\right) \Gamma\left(\frac{A_0 k}{A} - k\right)}$

The general flow of the package, and the inputs required for each function is presented in **Figure 6.1**.

The package provides three sets of functions for each stage of analysis:

- 1) `upgrain` and `upgrain.threshold` prepare atlas data for downscaling.
- 2) `downscale` and `hui.downscale` model the OAR to the prepared data for one of ten possible downscaling models.
- 3) `predict.downscale` and `plot.predict.downscale` take the model outputs and predict occupancy at finer grains.

Finally, `ensemble.downscale` will run `downscale` and `predict.downscale` for a number of selected downscaling functions and calculate the mean predicted occupancies across all models.

The user may input three types of data:

- 1) A data frame of grain sizes (cell area) and occupancies in that order;
- 2) A data frame of sample (cell) coordinates and presence-absence data (presence = 1; absence = 0). Column names must be "lon", "lat", and "presence";

3) A raster layer of presence-absence data (presence = 1; absence = 0; no data = NA).

If the user wishes to carry out downscaling with the Hui model (Hui et al. 2006, Hui et al. 2009) or prepare atlas data for downscaling using `upgrain` and `upgrain.threshold` then the input data must be of type 2 or 3. **Table 6.2** shows the functions to use to achieve desired objectives with regards to input data.

Figure 6.1: Structure of the downscale package showing all eight functions (yellow) and the three output object classes (orange).

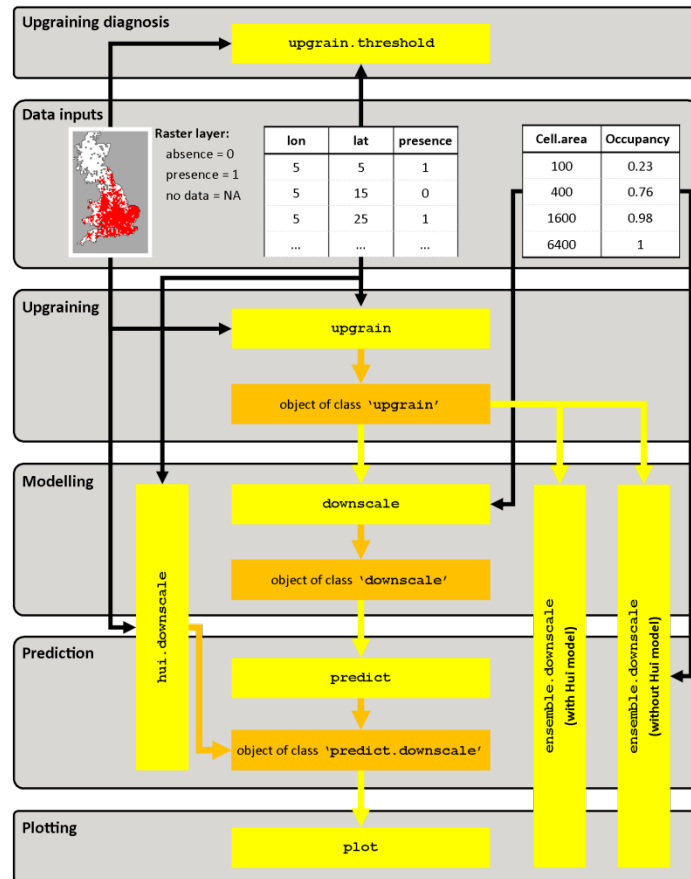


Table 6.2: Flow of functions for different objectives depending on data input type.

Input data type	Objective	Function flow
Data frame of cell areas and occupancies	Downscale (excluding Hui model)	<code>downscale</code> → <code>predict</code> → <code>plot</code>
Data frame of cell coordinates and presence-absence data	Downscale (excluding Hui model)	<code>upgrain.threshold</code> → <code>upgrain</code> → <code>downscale</code> → <code>predict</code> → <code>plot</code>
Raster layer of presence-absence data	Downscale (excluding Hui model)	<code>upgrain.threshold</code> → <code>upgrain</code> → <code>downscale</code> → <code>predict</code> → <code>plot</code>
Data frame of cell coordinates and presence-absence data	Downscale (Hui model)	<code>hui.downscale</code> → <code>plot</code>
Raster layer of presence-	Downscale (Hui model)	<code>hui.downscale</code> →

absence data		plot
Data frame of cell coordinates and presence-absence data	Ensemble modelling (excluding Hui model)	ensemble.downscale
Raster layer of presence-absence data	Ensemble modelling (with or without Hui model)	upgrain.threshold → upgrain → ensemble.downscale

6.3. Installing and using the ‘downscale’ package

The package may be installed directly from CRAN using the code:

```
install.packages("downscale")
```

The user may also have to install the following packages if they are not already present:

```
install.packages("raster")
install.packages("cubature")
install.packages("minpack.lm")
install.packages("Rmpfr")
```

Here a simple downscaling procedure is presented using the package example atlas data. The data is in the form of a data frame of sample cell coordinates and presence-absence data. First, we prepare this atlas data for downscaling by calculating occupancy at larger grain sizes. Then we model and predict occupancy at multiple finer grain sizes using ensemble modelling to average across all ten downscaling models.

```
## Load in the package
library(downscale)

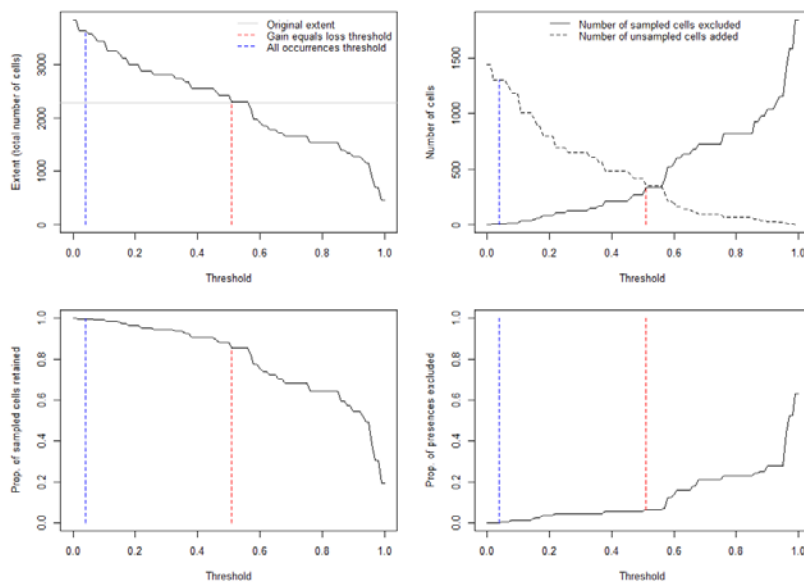
## Load in the example atlas data
data.file <- system.file("extdata", "atlas_data.txt", package = "downscale")
atlas.data <- read.table(data.file, header = TRUE)

## the data frame must have the column names "lon", "lat" and "presence":
head(atlas.data)

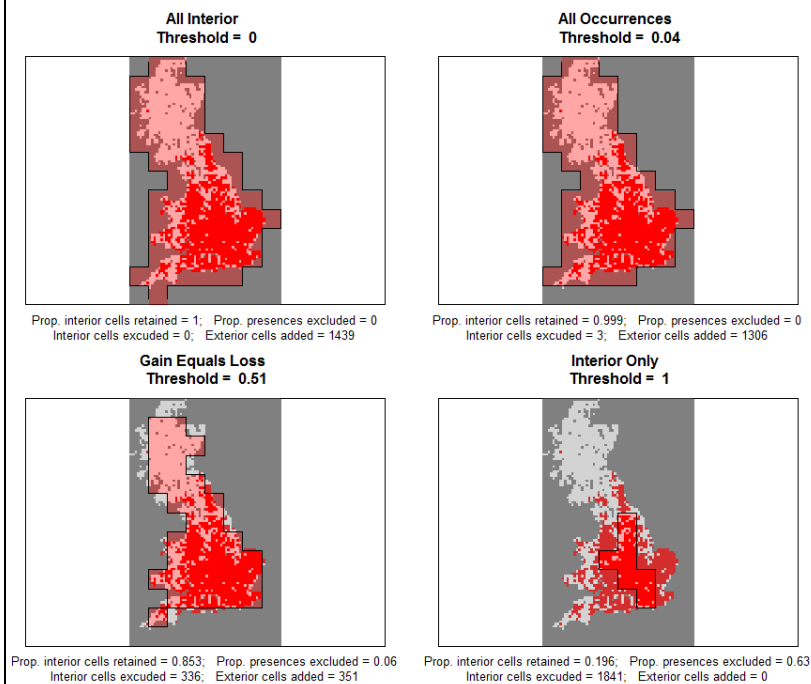
  lon lat presence
1 8170 10        0
2 8130 20        0
3 8140 20        1
4 8160 20        0
5 8170 20        1
6 8140 30        0

## explore thresholds using upgrain.threshold
thresh <- upgrain.threshold(atlas.data = atlas.data,
                           cell.width = 10,
                           scales = 3,
                           thresholds = seq(0, 1, 0.01))

## The first set of four plots explore the trade-offs of upgraining (see below)
```



The second set are the standardised atlas data generated after applying four different
threshold criteria

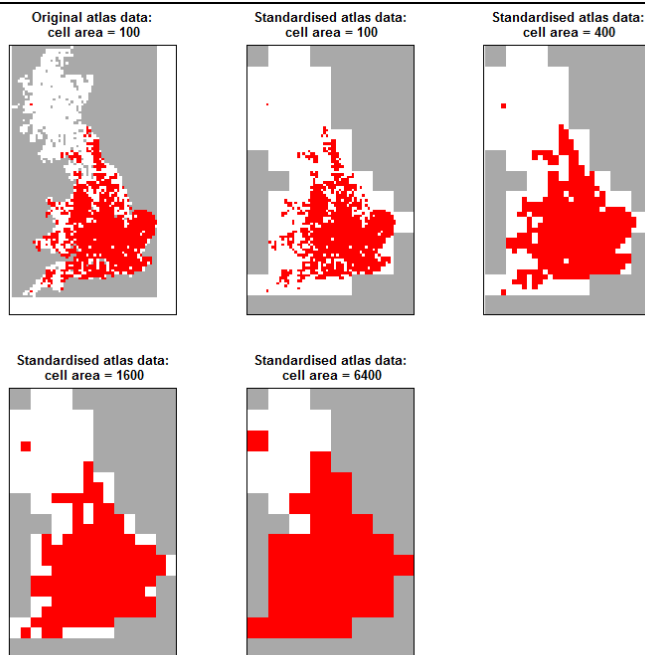


see the four optional thresholds
thresh\$Thresholds

	All_Sampled	All_Occurrences	Gain_Equals_Loss	Sampled_Only
1	0	0.04	0.51	1

upgrain atlas data using "All Presences" threshold to calculate standardised occupancy
at multiplescales ready for downscaling

```
occupancy <- upgrain(atlas.data,
  cell.width = 10,
  scales = 3,
  method = "All_Presences")
```



```
## Look at the standardised occupancy data
occupancy
```

```
$threshold
[1] 0.04
```

```
$extent.stand
[1] 364800
```

```
$occupancy.stand
  Cell.area Extent Occupancy
1      100 364800 0.2713542
2      400 364800 0.4122807
3     1600 364800 0.5219298
4     6400 364800 0.7017544
```

```
$occupancy.orig
  Cell.area Extent Occupancy
1      100 228900 0.4552206
2      400 364800 0.4122807
3     1600 364800 0.5219298
4     6400 364800 0.7017544
```

```
$atlas.raster.stand
class      : RasterLayer
dimensions : 104, 64, 6656 (nrow, ncol, ncell)
resolution : 10, 10 (x, y)
extent     : 8085, 8725, -65, 975 (xmin, xmax, ymin, ymax)
coord. ref.: NA
data source: in memory
names      : layer
values     : 1, 3 (min, max)
```

```
attr("class")
[1] "upgrain"
```

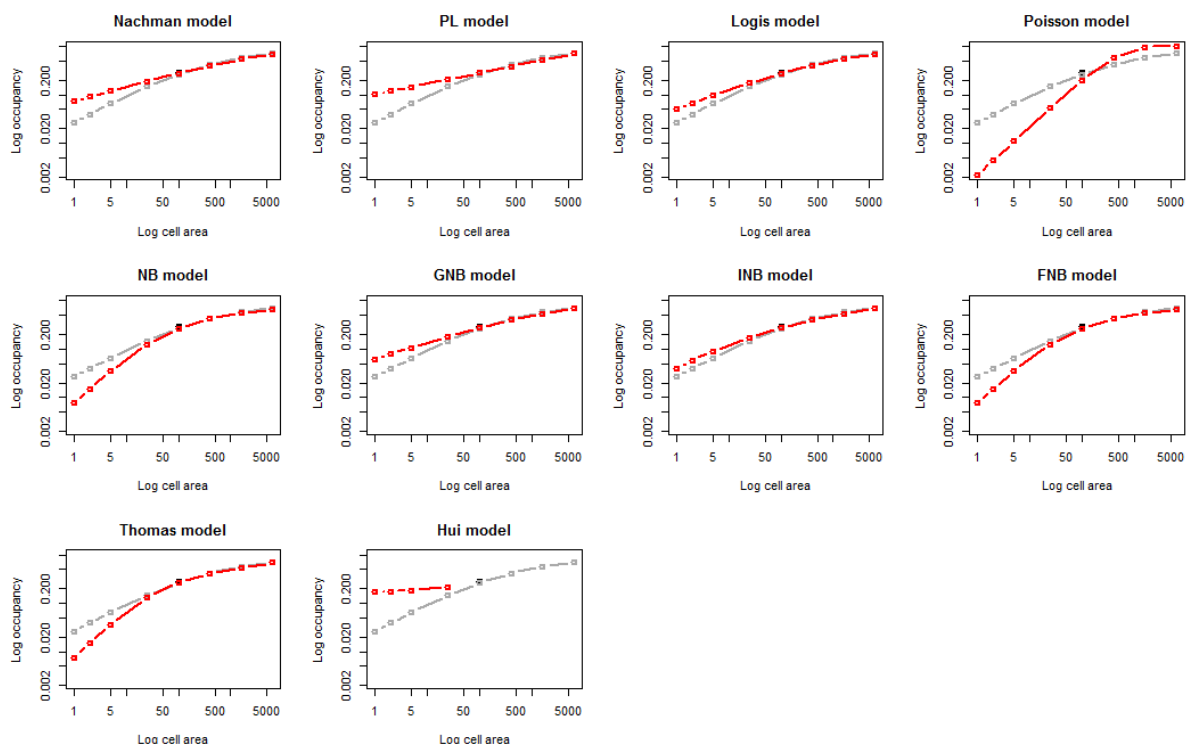
```
## downscale to fine grains using ensemble modelling of all models
```

```
ensemble <- ensemble.downscale(occupancy,
                               new.areas = c(1, 2, 5, 25, 100, 400, 1600, 6400),
```

```
cell.width = 10,
models = "all",
tolerance_mod = 1e-3,
plot = TRUE)
```

Nachman model is running... complete
 PL model is running... complete
 Logis model is running... complete
 Poisson model is running... complete
 NB model is running... complete
 GNB model is running... complete
 INB model is running... complete
 FNB model is running... complete
 Thomas model is running... complete
 Hui model is running... complete

the function plots all the model predictions (red), the original occupancy data (black) and the model averaged predictions (grey)



the predicted proportion of occupancies
 ensemble\$Occupancy

	Cell.area	Nachman	PL	Logis	Poisson	NB	GNB	INB
1	1	0.07322648	0.1015821	0.05126291	0.002195346	0.00758859	0.06196908	0.04004078
2	2	0.09024092	0.1185320	0.06744437	0.004385873	0.01483293	0.07869006	0.05772612
3	5	0.11853792	0.1453546	0.09610840	0.010928642	0.03476200	0.10730126	0.08951098
4	25	0.18888295	0.2079890	0.17303032	0.053461840	0.12556445	0.18094404	0.17240589
5	100	0.27659701	0.2831894	0.27264896	0.197301392	0.26703360	0.27444758	0.27353505
6	400	0.39395517	0.3855791	0.40175762	0.584845223	0.42057497	0.39783617	0.39990124
7	1600	0.53910499	0.5249887	0.54610120	0.970294275	0.55204406	0.54313505	0.54326569
8	6400	0.69821601	0.7148031	0.68308963	0.999999221	0.65587554	0.69052987	0.68879834
	FNB	Thomas	Hui	Means				
1	0.007677883	0.007663313	0.1717034	0.02631520				
2	0.015000598	0.015057206	0.1765820	0.03917022				
3	0.035112599	0.035746223	0.1863919	0.06531786				
4	0.126253351	0.133108488	0.2205252	0.14867123				
5	0.267257684	0.272739029	NA	0.26365838				
6	0.419923680	0.404167084	NA	0.41984459				

```
7 0.551269977 0.533122133 NA 0.57781285
8 0.657090227 0.695469992 NA 0.71461570
```

```
## the predicted area of occupancies (A00)
ensemble$A00
```

	Cell.area	Nachman	PL	Logis	Poisson	NB	GNB	INB
1	1	26713.02	37057.16	18700.71	800.8624	2768.318	22606.32	14606.88
2	2	32919.89	43240.47	24603.71	1599.9665	5411.054	28706.13	21058.49
3	5	43242.63	53025.37	35060.35	3986.7686	12681.177	39143.50	32653.61
4	25	68904.50	75874.40	63121.46	19502.8792	45805.911	66008.39	62893.67
5	100	100902.59	103307.48	99462.34	71975.5478	97413.859	100118.48	99785.59
6	400	143714.85	140659.24	146561.18	213351.5374	153425.749	145130.63	145883.97
7	1600	196665.50	191515.87	199217.72	353963.3515	201385.672	198135.66	198183.33
8	6400	254709.20	260760.18	249191.10	364799.7159	239263.398	251905.29	251273.63
	FNB	Thomas	Hui	Means				
1	2800.892	2795.576	62637.42	9599.787				
2	5472.218	5492.869	64417.12	14289.295				
3	12809.076	13040.222	67995.77	23827.957				
4	46057.222	48557.976	80447.59	54235.263				
5	97495.603	99495.198	NA	96182.577				
6	153188.159	147440.152	NA	153159.307				
7	201103.288	194482.954	NA	210786.129				
8	239706.515	253707.453	NA	260691.809				

Two vignettes are provided within the package to guide users in downscaling analyses and using the package functions. Both are provided as supplementary material in this deliverable. The first provides a detailed tutorial on using the package (*SI6.2-Downscaling_tutorial.pdf*, downloadable at <http://www.eubon.eu/documents/1/>), with examples using pre-existing atlas data, occupancy data as well as code for generating atlas data from GBIF data. The vignette can be accessed from within R with the following code:

```
vignette("Downscaling", package = "downscale")
```

The vignette tutorial is in section 3 (“Package tutorial”) of the vignette and follows this structure:

3.1 A quick example

A basic example where the user already has occupancy data for multiple scales. The section models occupancy, then predicts and plots occupancy at finer grain sizes.

3.2 Using atlas data

Input data is a data frame of cell coordinates and presence-absence data. The section upgrains this data to calculate occupancy at multiple spatial scales ready for downscaling analyses using the functions `upgrain` and `upgrain.threshold`.

3.3 Downscaling – more detailed examples

This section takes the data from the previous section and explores modelling occupancy using various downscaling models and user options, and then predicts and plots occupancy for finer grain sizes.

3.4 Ensemble modelling

This section explores the `ensemble.downscale` function. It uses both previous types of input data, and runs multiple downscaling models simultaneously, predicting and plotting occupancy at finer grain sizes for each model. It then averages occupancy estimates across all models.

3.5 Creating atlas data from point records

This section uses the ‘`spocc`’ library that automatically harvests GBIF data from within R. In the tutorial we harvest data for a UK butterfly species (*Polyommatus coridon*), and explore methods for creating atlas data from such point record data. We compare the downscale predictions of two methods of generating such atlas data.

The second vignette provides more detailed guidance on how to prepare atlas data in a way suitable for downscaling analyses, along with descriptions of the two functions provided for this purpose: `upgrain` and `upgrain.threshold`. In order to model occupancy against area we must ‘upgrain’ our atlas data to larger grain sizes to provide enough data points to fit the model. However, in order to maintain a constant extent across scales we must assign unsampled cells as absences (**Figure 6.2**)

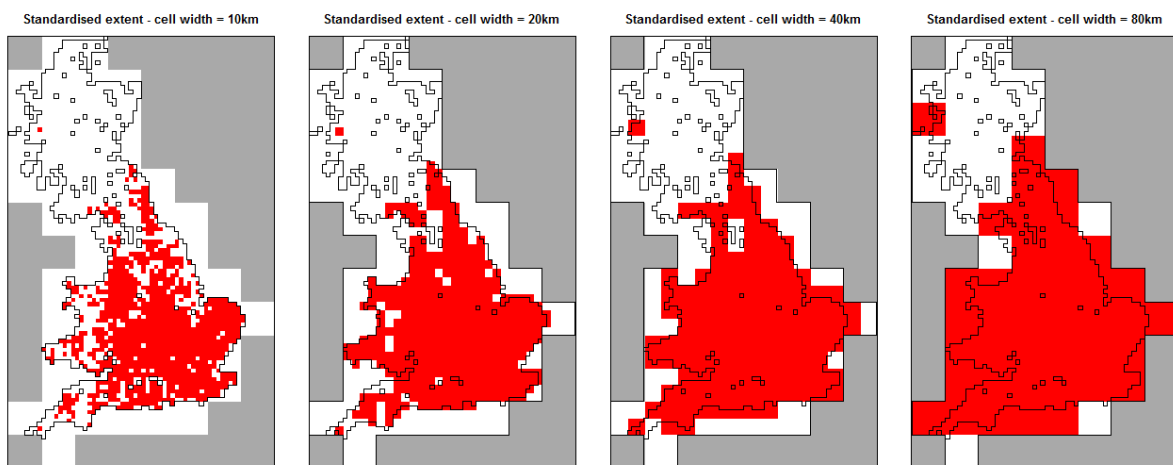


Figure 6.2: Upgrained presence (red cells) and absence (white cells) maps for a UK species after standardising extent to the largest grain size. Unsampled cells are dark grey. The extent of the atlas data is extended to that of the largest grain size by assigning absences to unsampled cells.

Alternatively we can set a threshold whereby only those cells at the largest grain size that contain a certain proportion of sampled cells are included. Therefore, there is a trade-off between converting unsampled cells to absences, and converting sampled cells to NA's. The `upgrain.threshold` function allows us to explore this trade-off through four diagnostic plots and the occupancy data required for downscaling is obtained through `upgrain`.

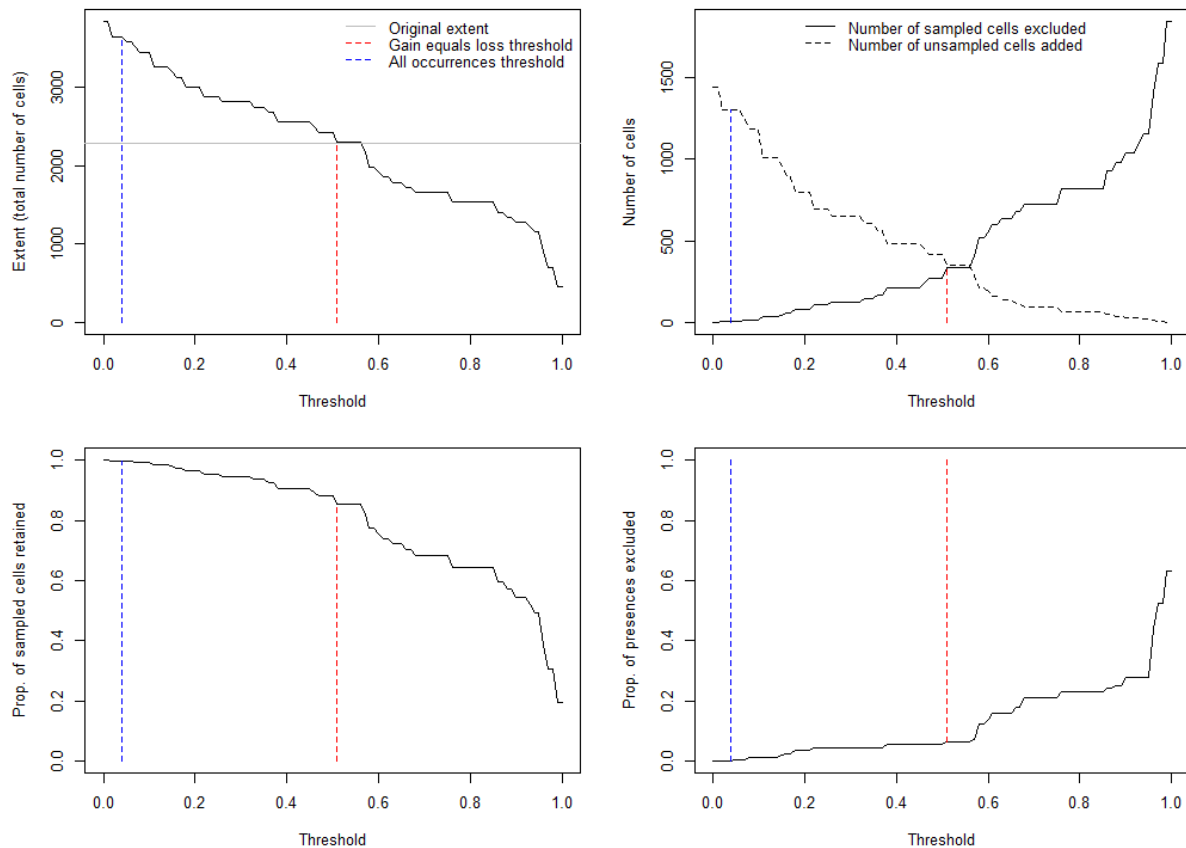


Figure 6.3: Diagnostic plots produced by `upgrain.threshold` used to explore the trade-off between assigning large areas of unsampled areas as absence, and discarding sampled areas and known presences. Two possible thresholds in the quantity of unsampled area allowed within cells at the largest grain size are identified: the “all occurrences” threshold (blue line) and the “gain equals loss” threshold (red line).

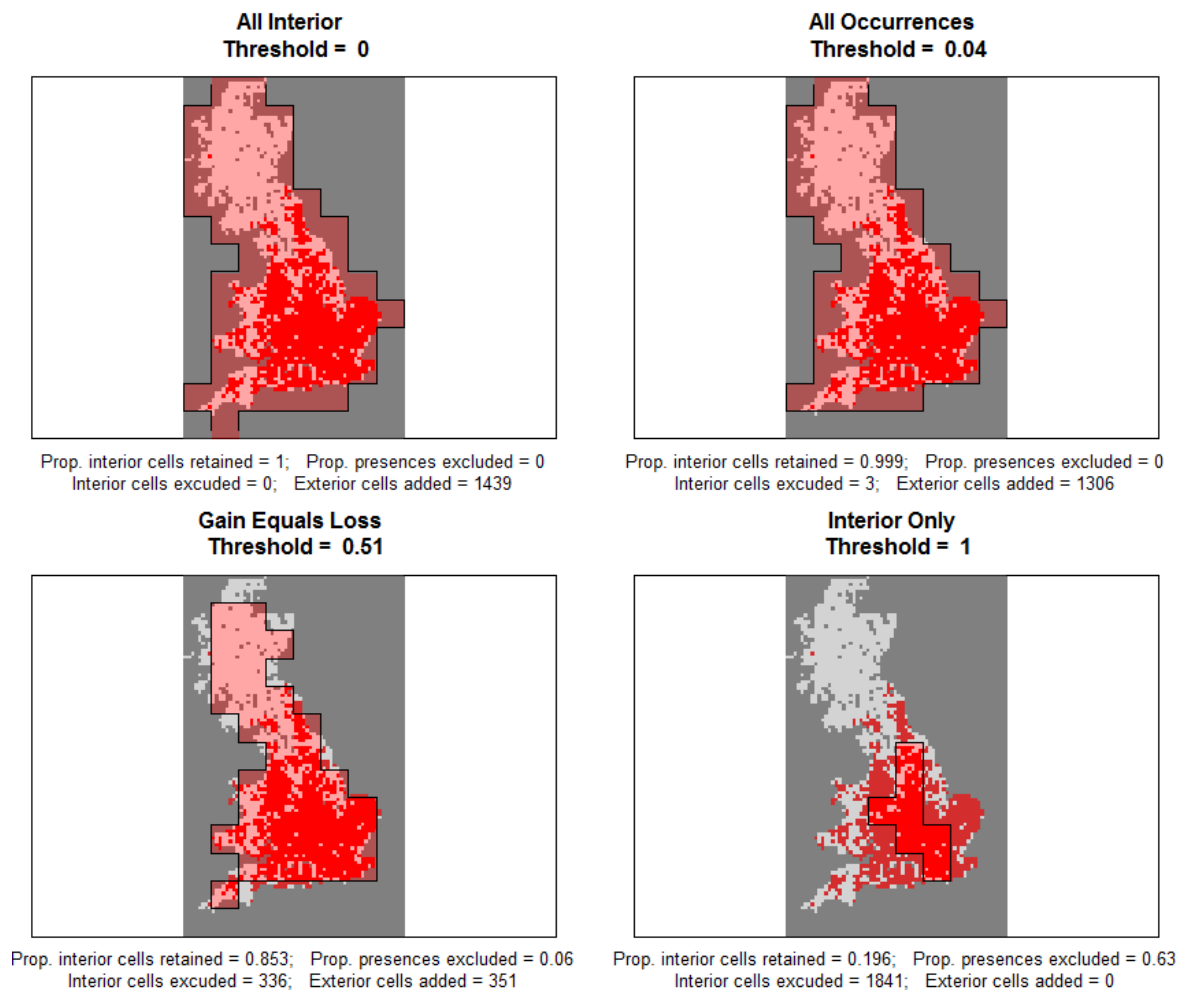
Although when using the `upgrain` function the user may input any threshold, `upgrain.threshold` proposes four possible trade-off criteria (**Table 6.3**, **Figure 6.3**). The final choice of threshold may have to be made on a case-by-case basis. For example, for a largely rectangular region there may be not much loss of data using the “Sampled Only” threshold. However, as our example atlas data has many unsampled cells within it, this threshold would result in a large loss of presence-absence data from the modelling (**Figure 6.4**).

Detailed information is available in the vignette (*SI6.3-Upgraining_tutorial.pdf*) accessed within the package by inserting the code:

```
vignette("Upgraining", package = "downscale")
```

Table 6.3: The four possible threshold criteria proposed by [upgrain.threshold](#).

Threshold	Name	Description
0	“All Sampled”	All of the original atlas data is included.
Species-specific	“All Occurrences”	The threshold where no occurrences in the atlas data are excluded.
Atlas-specific	“Gain Equals Loss”	The threshold where the number of sampled atlas cells reclassified as No Data equals the number of unsampled exterior cells reclassified as absence. In this threshold the new standardised extent also equals the extent of the original atlas data.
1	“Sampled Only”	Only cells that contain 100% sampled atlas data are included.

**Figure 6.4:** Maps of the atlas data (red = presence; light grey = absence; unsampled = dark grey) overlain with polygons showing the standardised extent after applying each of four possible thresholds.

7. VIRTUAL ECOLOGIST

7.1. Introduction

Key objective of long-term monitoring is to assess the status and trends of species and ecosystems, with particular attention to identifying changes over time. Such changes may be gradual or abrupt; are often non-linear, and can be masked by large spatiotemporal variation emerging from, e.g., environmental- and demographic stochasticity. Cost-effective design of monitoring efforts is therefore key to identifying spatiotemporal trends with limited efforts, and overcoming problems relating to the scaling of knowledge and existing data. Typical questions are how to downscale information from species' distribution to local patterns of presence or abundance; how to upscale from local (site-level) observations to large-scale patterns of change in distribution or community structures; or how to set sampling sites for maximizing the chance of identifying these changes while minimizing on efforts and costs. Among the many relevant factors to consider, some are the type, cost and skills of different observer types – especially paid experts versus volunteers. Experts may detect species with higher probability and identify them with lower levels of errors but volunteers can offer huge manpower for much lower costs. The work with volunteers (citizen scientists), however, must consider behavioural aspects such as missed visits, higher errors and preferences to sample one habitat or another (e.g., it is harder to conduct voluntary sampling in remote or unappealing environments). Another important set of constraining factors is the species' characteristics, such as life cycle, the duration of adult activity (e.g. butterflies versus birds or mammals), reproduction rates and hence the sensitivity to environmental variables and speed of that response.

The Virtual Ecologist approach offers the means to handle the large number of variables and constraints affecting cost-efficiency questions of monitoring efforts, while reducing the largest limitation of not knowing the actual, exact distribution or abundance of species. The approach builds on “producing” a certain “reality” using a certain model, and then sampling from it to identify a) the impacts of observer-behavior on sampling outcomes; b) the probability to succeed in capturing the “real” spatiotemporal pattern; and, thereby, b) the best sampling design given the (potentially large) set of constraints.

Within EU BON, the UFZ has developed a “virtual ecologist” tool which can sample either from population models or from real data, using alternative set of parameters defining observer types (especially, professional versus volunteers), to maximize the effectiveness of monitoring efforts in downscaling from larger-scale knowledge of species' distribution and density, to local scale patterns of presence and abundance. The tool is based on sampling from the outcomes of spatially-explicit models, individual-based simulation models such as RangeShifter (Bocedi et al. 2014), which produce either a population projection (=temporal pattern) or a set of maps with species' abundance- and distribution pattern (=spatiotemporal pattern) for one or more species.

The power of using individual-based, dynamic simulation models to produce these outcomes is that it allows strengthening the link between ecological processes and observed patterns (Grimm and Railsback 2005, Grimm 2008, Railsback and Grimm 2011). Processes occurring at the individual level translate into larger-scale, emergent patterns such as connectivity, population viability, trends in abundance or distribution (e.g. species' decline, range-shifts, or changes in community structures if examined across multiple species), or community dynamics. Individual-based models offer not only tractable route from processes to patterns but also the opportunity of testing their emerging outputs against a range of known ecological patterns – an approach known as “Pattern oriented modelling” (Railsback 2001).

In the case of a “Virtual Ecologist”, the approach we take here is to link two individual-based steps. First, a user can employ an IBM for the production of ecological patterns that are controlled to a certain extent, and can be considered as “virtual reality” that is fully known. An example for an IBM which can be used to produce such simulations is *RangeShifter* (Bocedi et al. 2014). In a second step, the virtual ecologist itself simulates the process in which an ecologist samples from that environment – thus producing imperfect information about the spatiotemporal patterns of abundance or presence. By employing a model which mimics this observation process, we gain a means to assess how different sampling strategies and intensities diverge from “reality”. Thereby, we can test the impacts of key parameters such as sampling design, behavior of the observers etc. – aspects that, without such a model, can only be speculated upon. The Virtual Ecologist can therefore aid in interpreting (existing) biodiversity data by producing null models; assessing the efficiency of alternative sampling designs (e.g. frequency, intensity); identifying sources of error (e.g. species detectability or recognition) and highlight potential biases introduced by certain observation behaviors (e.g. “habitat preferences” of volunteers). The virtual-ecologist approach has been used by modelers for over 15 years (e.g., Grimm et al. 1999, Moilanen 1999, Tyre et al. 2001), but has only taken off recently following a review by Zurell et al. (2010) describing the power, applications and potentials of the approach. Recent applications of the approach, of relevance to the EU BON project, relate to assessing the scale-related relationships between species and their environments (Lechner et al. 2012) and optimization of monitoring design and efforts (Albert et al. 2010, Nuno et al. 2013).

7.2. Model Concepts and structure

The following model descriptions follow the ODD (Overview, Design concepts, Details) protocol (Grimm et al. 2006, Grimm et al. 2010) developed originally for describing individual and agent based models.

7.2.1. Purpose

This model aims to optimize the cost-efficiency of monitoring programs when aiming to detect the status and trends of species' populations and communities, as well as downscaling from (large-scale) distributions to (local) abundances and upscaling from local information to larger-scale projections. It

specifically seeks to address questions regarding the optimal number of sites, sampling frequency and balance between observer types (professionals versus volunteers), given certain limitations on budget or manpower, considerations of species-attributes, and the desired ecological patterns to identify.

7.2.2. Elements, entities, scales and variables

A **Scheme** is characterised by its aims (to sample a given species or a community), its spatial and temporal scale, sampling design (e.g. random, stratified or stratified-random), budget or manpower limitations, and the type of observers it seeks to engage. Sampling locations within a scheme can relate to plots, points or transects but they have no further spatial characterization (i.e., no sub-sections are applied).

The main **entities** within a scheme are the observers, who need to identify the (individuals of) species sampled. Observers can be experts or volunteers, and are characterized by the following variables: a) their capacity to detect species (detection probability), b) identification error, c) probability of missed visits, d) cost per person and visit, and e) habitat preference. The probability of missed visits is an important consideration in our model, since volunteers often fail to perform visits to their site in a prescribed time. The “habitat preference” issue is important because voluntary-based schemes are known to be affected by the preference of volunteers to establish sampling sites in specific habitats but less so in others – hence limiting the potential density of sites in less preferred regions.

Species are characterised by their abundance (or density), habitat requirements, movement capacities and demography, but these properties are external to our model – namely, they are predetermined by the input data provided to the Virtual Ecologist by “step 1”, where inputs are produced by other (existing) modelling frameworks. Within the Virtual Ecology model itself, the only characterising variable is the detection probability, which affects the relation between the total number of individuals occurring within sites (=“reality”) and the number that observers actually detect and report on per site and time-step (=emergent outcome of the sampling process).

Other model variables relate to the way in which the Virtual Ecologist model operates – namely, how sampling sites are established (e.g. added or removed from a pre-determined starting distribution), what are the initial conditions (e.g. existing scheme or a new one), how many iterations of the sampling (= time horizon) and how many simulations.

7.2.3. Processes

The two core processes in the model are sampling and optimization. The sampling process describes the way in which input data are sampled from. It is an iterative process, where subsampling from the same input data can be done with alternative frequencies, intensities etc. – to produce a certain probability of detecting the (pre-determined) status or trend of species or communities. The sampling process produces a sampling pattern (when and where sampling occur) and how much it costs.

The user may wish to explore the impacts of specific variables on the model outcome, using a global sensitivity analysis or a systematic exploration of a certain parameter space. However, a more effective process for identifying optimal parameter spaces is based on employing a “simulated annealing” algorithm. Simulated Annealing (SA) is a broadly used optimization approach which is highly effective for identifying optimal parameter combinations within a complex parameter space containing multiple parameters. The algorithm is based on performing repeated simulations over the same dataset, starting with large variance between simulations (altering the starting parameters at random) and then selecting “best combinations” among the outcomes (lowest costs, higher probability to detect a trend) and slowly reducing the randomness (amount of variance between simulations) over the simulation iterations.

The optimization process can start from a given (existing) scheme or aid in establishing a new one. In the case of an existing scheme it asks whether sites should be added or could be removed; whether sampling frequency may need to increase or alternatively could be reduced; or whether preference should be given for professionals or volunteers.

7.3. Design concepts

The model is designed to be as simple and intuitive as possible, and to mimic sampling principles in a way that can be easily communicated with field ecologists, conservation practitioners and decision-makers while offering useful scientific tools for either quantitative evaluations of existing schemes, or scenario-based assessments of new schemes. Some important principles of the tool are:

- a) Robustness of the desired outcome (e.g., probability to detect a trend) could be either *emergent properties* of the analysis or imposed as a desired threshold.
- b) Model outcomes should be directly comparable to the *objective* of a given monitoring scheme, as a “fitness check” against its targets.
- c) *Stochasticity* is considered by the a) selection of plots, b) *behaviour* of samplers, and c) errors during the reporting stage. Note that other sources of stochasticity, such as environmental and the demographic stochasticity, are external to the model and should be considered when preparing the inputs = dataset to be sampled by the model.

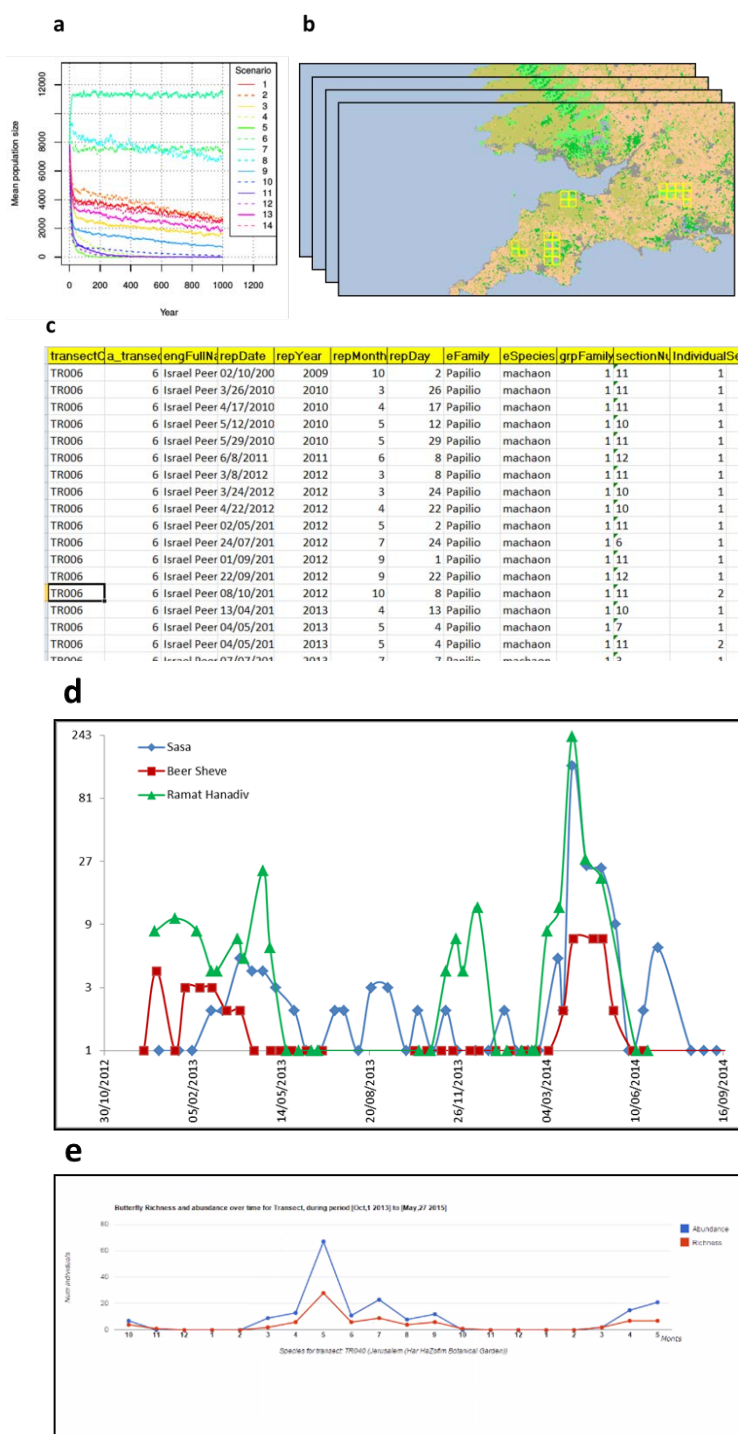
7.4. Implementation

To run the Virtual Ecologist, an input dataset should contain information on a) where (coordinates of existing sampling points if existing), what to sample (number of individuals over time and space per species, namely a time-series or a set of abundance-maps over time), when (a set of parameters defining the desired, or actual sampling within and between years) and in what context (a map describing the land-use/land-cover of the designated area, and potentially habitat patches within it). Other input parameters include “habitat preference” of observers, and batch-filing options for repeated simulations.

Exemplary inputs are shown in **Figure 7.1**. We note here that the model can work both with simulated data and real data originating from monitoring activities, but in the latter case, one cannot identify deviations from a known “reality” since the input data are already affected by a certain sampling behaviour.

Figure 7.1: Examples of potential sources of input for the Virtual Ecologist model:

- simulated series of population size over time, in this case produced over various scenarios for a virtual species using RangeShifter (Source: Trouillier et al. in prep.);
- simulated set of abundance-distribution maps, in this case demonstrated through a virtual species over a real landscape (here, the south of England; source: Bocedi et al. 2014, RangeShifter); or
- real data originating in this case from the Israeli butterfly monitoring scheme – here showing a time-series for a given species (*Papilio machaon*). From such data, one can derive
 - a time-series per species (here, *Vanessa cardui*) or
 - a community (here, example from one transect site). Real data used here originate from the Israeli BMS (kindly provided by GlueCAD).



7.5. Outputs

The output of the model is a dataset of the sampled plots with the sampled number of individuals, the sampler (expert or citizen) and the costs. The dataset can be used for further statistical evaluation and for comparison with the input dataset – and summarized by e.g. the number of individuals observed (abundance), trend, proportion of simulations where the trend was successfully identified, richness or diversity if the scheme aims toward communities, and total costs.

7.6. Additional notes

The model is being developed as an R-package, to be deposited at a later stage on CRAN. The R-package provides three different functions for the sampling method of the virtual ecologist:

`PrepareDataset`, `CreateEcologist` and `Sampling`. The function `PrepareDataset` transforms the input data into a standardized form which will be used by the `Sampling` function.

The `CreateEcologist` generate vectors with the characteristics of a type of virtual ecologists. It can run twice, namely once for an expert and once for a volunteer. The `Sampling` function requires the results of the two other functions and additional inputs (see above, “Inputs”). Two additional functions are aimed at “batch-filing” (for systematic parameter exploration) and “VEoptimization”.

8. REFERENCES

- Albert, C. H., N. G. Yoccoz, T. C. Edwards, C. H. Graham, N. E. Zimmermann, and W. Thuiller. 2010. Sampling in ecology and evolution - bridging the gap between theory and practice. *Ecography* **33**:1028-1037.
- Azaele, S., S. J. Cornell, and W. E. Kunin. 2012. Downscaling species occupancy from coarse spatial scales. *Ecological Applications* **22**:1004-1014.
- Azaele, S., A. Maritan, S. J. Cornell, S. Suweis, J. R. Banavar, D. Gabriel, and W. E. Kunin. 2015. Towards a unified descriptive theory for spatial ecology: predicting biodiversity patterns across spatial scales. *Methods in Ecology and Evolution* **6**:324-332.
- Barwell, L. J., S. Azaele, W. E. Kunin, and N. J. B. Isaac. 2014. Can coarse-grain patterns in insect atlas data predict local occupancy? *Diversity and Distributions* **20**:895-907.
- Bivand, R. 2014. spgrass6: Interface between GRASS 6 and R. . R package version 0.8-6.
- Bocedi, G., S. C. F. Palmer, G. Pe'er, R. K. Heikkinen, Y. G. Matsinos, K. Watts, and J. M. J. Travis. 2014. RangeShifter: a platform for modelling spatial eco-evolutionary dynamics and species' responses to environmental changes. *Methods in Ecology and Evolution* **5**:388-396.
- Bowman, A. W., and A. Azzalini. 2013. R Package 'sm': Nonparametric Smoothing Methods. Accessed October 20, 2013. <http://www.stats.gla.ac.uk/~adrian/sm>, http://azzalini.stat.unipd.it/Book_sm.
- Bradter, U., T. J. Thom, J. D. Altringham, W. E. Kunin, and T. G. Benton. 2011. Prediction of National Vegetation Classification communities in the British uplands using environmental data at multiple spatial scales, aerial images and the classifier random forest. *Journal of Applied Ecology* **48**:1057-1065.
- Breiman, L. 2001. Random forests. *Machine Learning* **45**:5-32.
- Burnham, K. P., and R. Anderson. 2002. Model selection and multimodel inference - A practical information - theoretic approach. Second edition. Springer Press.
- Carr, D., N. Lewin-Koh, and M. Maechler. 2013. hexbin: Hexagonal Binning Routines. R Package Version 1.26.0. Accessed October 20, 2013. <http://cran.r-project.org/package=hexbin>.
- Chen, C., and L. Breiman. 2004. Using random forests to learn imbalanced data, Technical Report 666. Statistics Department of University of California at Berkeley.
- Coops, N. C., M. A. Wulder, and D. Iwanicka. 2009. Demonstration of a satellite-based index to monitor habitat at continental-scales. *Ecological Indicators* **9**:948-958.
- Fourier, J. B. J. 1822. *Théorie Analytique De La Chaleur*. Paris: Didot.
- Frigo, M., and S. G. Johnson. 2005. The Design and Implementation of FFTW3 Proceedings of the IEEE **93**:216-231.
- Grimm, V. 2008. Individual-based models. Pages 1959-1968 in S. E. Jorgensen and B. D. Fath, editors. *Encyclopedia of Ecology*. Elsevier, Amsterdam.
- Grimm, V., U. Berger, F. Bastiansen, S. Eliassen, V. Ginot, J. Giske, J. Goss-Custard, T. Grand, S. K. Heinz, G. Huse, A. Huth, J. U. Jepsen, C. Jorgensen, W. M. Mooij, B. Muller, G. Pe'er, C. Piou, S. F. Railsback, A. M. Robbins, M. M. Robbins, E. Rossmannith, N. Ruger, E. Strand, S. Souissi, R. A. Stillman, R. Vabo, U. Visser, and D. L. DeAngelis. 2006. A standard protocol for describing individual-based and agent-based models. *Ecological Modelling* **198**:115-126.
- Grimm, V., U. Berger, D. L. DeAngelis, J. G. Polhill, J. Giske, and S. F. Railsback. 2010. The ODD protocol: A review and first update. *Ecological Modelling* **221**:2760-2768.
- Grimm, V., and S. F. Railsback. 2005. *Individual-based Modeling and Ecology*. Princeton University Press, Princeton, N.J.
- Grimm, V., T. Wyszomirski, D. Aikman, and J. Uchmanski. 1999. Individual-based modelling and ecological theory: synthesis of a workshop. *Ecological Modelling* **115**:275-282.
- Harte, J., A. B. Smith, and D. Storch. 2009. Biodiversity scales from plots to biomes with a universal species-area curve. *Ecology letters* **12**:789-797.
- Harte, J., T. Zillio, E. Conlisk, and A. B. Smith. 2008. Maximum entropy and the state-variable approach to macroecology. *Ecology* **89**:2700-2711.
- Hartley, S., and W. E. Kunin. 2003. Scale dependency of rarity, extinction risk, and conservation priority. *Conservation Biology* **17**:1559-1570.
- Hoffmann, A., J. Penner, K. Vohland, W. Cramer, R. Doubleday, K. Henle, U. Köljal, I. Kühn, W. E. Kunin, J. Negro, L. Penev, C. Rodríguez, H. Saarenmaa, D. Schmeller, P. Stoev, W. Sutherland, É. Ó. Tuama, F. Wetzel, and C. Häuser. 2014. The need for an integrated biodiversity policy support process – Building the European contribution to a global Biodiversity Observation Network (EU BON). . *Nature Conservation* **6**:49-65.

- Hothorn, T., K. Hornik, and A. Zeileis. 2006. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics* **15**:651-674.
- Hui, C., M. A. McGeoch, B. Reyers, P. C. le Roux, M. Greve, and S. L. Chown. 2009. Extrapolating population size from the occupancy-abundance relationship and the scaling pattern of occupancy. *Ecological Applications* **19**:2038-2048.
- Hui, C., M. A. McGeoch, and M. Warren. 2006. A spatially explicit approach to estimating species occupancy and spatial correlation. *Journal of Animal Ecology* **75**:140-147.
- IUCN. 2014. Guidelines for using the IUCN Red List categories and criteria.
- Jobe, R. T. 2008. Estimating landscape-scale species richness: Reconciling frequency- and turnover-based approaches. *Ecology* **89**:174-182.
- Kiritchenko, S., S. Matwin, and F. Famili. 2005. Functional annotation of genes using hierarchical text categorization. In: *Proc. of the ACL Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics*.
- Kunin, W. E. 1998. Extrapolating species abundance across spatial scales. *Science* **281**:1513-1515.
- Lechner, A. M., W. T. Langford, S. D. Jones, S. A. Bekessy, and A. Gordon. 2012. Investigating species-environment relationships at multiple scales: Differentiating between intrinsic scale and the modifiable areal unit problem. *Ecological Complexity* **11**:91-102.
- Lucas, R., K. Medcalf, A. Brown, P. Bunting, J. Breyer, D. Clewley, S. Keyworth, and P. Blackmore. 2011. Updating the Phase 1 habitat map of Wales, UK, using satellite sensor data. *Isprs Journal of Photogrammetry and Remote Sensing* **66**:81-102.
- Lucas, R., A. Rowlands, A. Brown, S. Keyworth, and P. Bunting. 2007. Rule-based classification of multi-temporal satellite imagery for habitat and agricultural land cover mapping. *Isprs Journal of Photogrammetry and Remote Sensing* **62**:165-185.
- McGeoch, M. A., and K. J. Gaston. 2002. Occupancy frequency distributions: patterns, artefacts and mechanisms. *Biological Reviews* **77**:311-331.
- Melgani, F., and L. Bruzzone. 2004. Classification of hyperspectral remote sensing images with support vector machines. *Ieee Transactions on Geoscience and Remote Sensing* **42**:1778-1790.
- Mertens, B., and E. F. Lambin. 1997. Spatial Modelling of Deforestation in Southern Cameroon: Spatial Disaggregation of Diverse Deforestation Processes. *Applied Geography* **17**:143-162.
- Metz, M., D. Rocchini, and M. Neteler. 2014. Surface Temperatures at the Continental Scale: Tracking Changes with Remote Sensing at Unprecedented Detail. *Remote Sensing* **6**:3822-3840.
- Moilanen, A. 1999. Patch occupancy models of metapopulation dynamics: Efficient parameter estimation using implicit statistical inference. *Ecology* **80**:1031-1043.
- Nekola, J. C., and P. S. White. 1999. The distance decay of similarity in biogeography and ecology. *Journal of Biogeography* **26**:867-878.
- Neteler, M. 2010. Estimating daily Land Surface Temperatures in mountainous environments by reconstructed MODIS LST data. *Remote Sensing* **2**:333-351.
- Neteler, M., M. H. Bowman, M. Landa, and M. Metz. 2012. GRASS GIS: A multi-purpose open source GIS. *Environmental Modelling & Software* **31**:124-130.
- Nuno, A., N. Bunnefeld, and E. J. Milner-Gulland. 2013. Matching observations and reality: using simulation models to improve monitoring under uncertainty in the Serengeti. *Journal of Applied Ecology* **50**:488-498.
- Oksanen, J., F. Guillaume Blanchet, R. Kindt, P. Legendre, P. R. Minchin, R. B. O'Hara, G. L. Simpson, P. Solymos, M. H. S. Stevens, and H. Wagner. 2015. *vegan: Community Ecology Package*. R package version 2.2-1.
- Palmer, M. W., P. Earls, B. W. Hoagland, P. S. White, and T. Wohlgemuth. 2002. Quantitative Tools for Perfecting Species Lists. *Environmetrics* **13**:121-137.
- R Development Core Team. 2011. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, <http://www.R-project.org>.
- Railsback, S. F. 2001. Getting "results": The pattern-oriented approach to analyzing natural systems with individual-based models. *Natural Resource Modeling* **14**:465-474.
- Railsback, S. F., and V. Grimm. 2011. *Agent-Based and Individual-Based Modeling: A Practical Introduction*. Princeton University Press, Princeton, N.J.
- Rocchini, D., M. Metz, C. Ricotta, M. Landa, A. Frigeri, and M. Neteler. 2013. Fourier transforms for detecting multitemporal landscape fragmentation by remote sensing. *International Journal of Remote Sensing* **34**:8907-8916.
- Scheiner, S. M. 2003. Six types of species-area curves. *Global Ecology and Biogeography* **12**:441-447.

- Shen, T. J., and F. L. He. 2008. An incidence-based richness estimator for quadrats sampled without replacement. *Ecology* **89**:2052-2060.
- Silla, C. N., and A. A. Freitas. 2011. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery* **22**:31-72.
- Soininen, J., R. McDonald, and H. Hillebrand. 2007. The distance decay of similarity in ecological communities. *Ecography* **30**:3-12.
- Strobl, C., A. L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis. 2008. Conditional variable importance for random forests. *Bmc Bioinformatics* **9**.
- Thoonen, G., T. Spanhove, J. Vanden Borre, and P. Scheunders. 2013. Classification of heathland vegetation in a hierarchical contextual framework. *International Journal of Remote Sensing* **34**:96-111.
- Tjørve, E. 2003. Shapes and functions of species-area curves: a review of possible models. *Journal of Biogeography* **30**:827-835.
- Tjørve, E. 2009. Shapes and functions of species-area curves (II): a review of new models and parameterizations. *Journal of Biogeography* **36**:1435-1445.
- Tjørve, E. 2012. Arrhenius and Gleason revisited: new hybrid models resolve an old controversy. *Journal of Biogeography* **39**:629-639.
- Tuomisto, H. 2010a. A diversity of beta diversities: straightening up a concept gone awry. Part 1. Defining beta diversity as a function of alpha and gamma diversity. *Ecography* **33**:2-22.
- Tuomisto, H. 2010b. A diversity of beta diversities: straightening up a concept gone awry. Part 2. Quantifying beta diversity and related phenomena. *Ecography* **33**:23-45.
- Tyre, A. J., H. P. Possingham, and D. B. Lindenmayer. 2001. Inferring process from pattern: Can territory occupancy provide information about life history parameters? *Ecological Applications* **11**:1722-1737.
- Ugland, K. I., J. S. Gray, and K. E. Ellingsen. 2003. The species-accumulation curve and estimation of species richness. *Journal of Animal Ecology* **72**:888-897.
- Xu, H., S. Liu, Y. Li, R. Zang, and F. He. 2012. Assessing non-parametric and area-based methods for estimating regional species richness. *Journal of Vegetation Science* **23**:1006-1012.
- Zurell, D., U. Berger, J. S. Cabral, F. Jeltsch, C. N. Meynard, T. Munkemüller, N. Nehrbass, J. Pagel, B. Reineking, B. Schroder, and V. Grimm. 2010. The virtual ecologist approach: simulating data and observers. *Oikos* **119**:622-635.

9. ACKNOWLEDGMENTS

We wish to acknowledge the contribution of the following colleagues from outside the EU BON consortium: Jerome O’Connell (University of Leeds, UK) provided GIS analysis for the hierarchical random forest model; Sandro Azaele (University of Leeds, UK) provided code for pair correlation functions and advice on downscaling methods; Louise Barwell (CEH and University of Leeds, UK) provided draft R code for multiple downscaling methods; Cang Hui (Stellenbosch University, South Africa) provided draft code for the Hui downscaling model; Han Xu and Fangliang He (University of Alberta, CA) provided draft code for the simplified maximum entropy and Observed and True SOD upscaling models; and Even Tjørve (Lillehammer University College, NO) provided advice on SAR functions.

10. APPENDICES

This deliverable includes the following appendices:

- A2.1** - Table with definitions used in the hierarchical randomForest tool
- A2.2** - Performance measures that account for the class hierarchy and their estimation from a confusion matrix.

A2.1. Definitions of terms used in the hierarchical random forest section (section 2) and in the help file of the 'HieRanFor' R package.

Term	Definition and examples
<i>class hierarchy</i>	A tree structure containing the tree root and all the internal and terminal nodes. In HieRanFor, the class structure should be a directed tree with directions pointing from nodes closer to the tree root to those further down the hierarchy. Furthermore, each node may have only one parent node. See example in Figure 2.1B .
<i>hierarchical level</i>	An integer specifying the distance from the tree root. Each hierarchical level contains a set of nodes within the same distance from the tree root. Nodes in the same hierarchical level cannot be linked directly to one another in a tree like class hierarchy. e.g., 'Level 0', 'Level 1', and 'Level 2' in the Figure 2.1B .
<i>node</i>	A class in the class hierarchy. e.g., 'H1', 'H6' or 'H7' in Figure 2.1B .
<i>tree root</i>	The node at the lowest hierarchical level of the class hierarchy. The only node that has no parent node. All other nodes are descended from the tree root node. e.g., 'H0' in Figure 2.1B .
<i>internal node</i>	A node that has at least one children node. e.g., 'H1'.
<i>terminal node</i>	A node that has no children nodes. e.g., 'H3'.
<i>parent node</i>	The linked node one level closer to the tree root from a focal node. In a tree like class structure, each internal or terminal node have a single parent node. e.g., for the focal node 'H5', the parent node is 'H2'.
<i>children node</i>	A linked node one level below a focal node in the class structure. All internal nodes have at least one children node. Terminal nodes have no child nodes. e.g., for the parent node 'H4' the children nodes are 'H7', 'H8' and 'H9'.
<i>sibling nodes</i>	All nodes that have the same parent node. e.g., 'H7', 'H8' and 'H9'.
<i>tree depth</i>	The number of hierarchical levels in the class hierarchy where the tree root is considered as level 0. The tree depth of Figure 2.1B is 2.
<i>path</i>	A linked sequence of nodes starting from the tree root, moving uni-directionally and ending at a terminal node. e.g., 'H0' → 'H2' → 'H5'.
<i>flat classification</i>	A classification in which all terminal nodes are considered to be in hierarchical level 1 (i.e., children nodes of the tree root). A flat classification will classify all the terminal nodes in a single local classifier, ignoring the class hierarchy (Figure 2.1A).
<i>hierarchical classification</i>	A classification in which at least one terminal node is from hierarchical level > 1 (i.e., there is at least one internal node). In Figure 2.1B , 3 local flat classifiers are required to run a single hierarchical classification.
<i>local classifier</i>	A single flat classification within a hierarchical classification. A local classifier is a randomForest algorithm that classifies all the children nodes of a given parent node. In Figure 1B , the local classifier 'C3' classifies 'H7', 'H8' and 'H9' (the children nodes of 'H4').
<i>case</i>	A single data point in the training data or new data.
<i>vote</i>	The output of a single classification tree in a single local classifier for a single case.

A2.2. Performance measures that account for the class hierarchy can be estimated directly from a confusion matrix.

<i>Index</i>	<i>Level</i>	<i>Equation</i>	<i>Source</i>
Hierarchical precision	General	$hP = \sum_j \sum_k (S_{j,k} \cdot F_{j,k}) / \sum_k (S_{k,k} \cdot F_{+,k})$	Kiritchenko et al. (2005)
	Per node	$hP_k = \sum_j (S_{j,k} \cdot F_{j,k}) / (S_{k,k} \cdot F_{+,k})$	This deliverable
Hierarchical Recall	General	$hR = \sum_j \sum_k (S_{j,k} \cdot F_{j,k}) / \sum_j (S_{j,j} \cdot F_{j,+})$	Kiritchenko et al. (2005)
	Per node	$hR_j = \sum_k (S_{j,k} \cdot F_{j,k}) / (S_{j,j} \cdot F_{j,+})$	This deliverable
Hierarchical F measure	General	$hF = [(\beta^2 + 1) \cdot hP \cdot hR] / (\beta^2 \cdot hP + hR)$	Kiritchenko et al. (2005)
	Per node	$hF_{j=k} = [(\beta^2 + 1) \cdot hP_k \cdot hR_j] / (\beta^2 \cdot hP_k + hR_j)$	This deliverable

j - An observed class. k - A predicted class. $S_{j,k}$ - The length of the path from the first common ancestor of classes j and k to the root (for $S_{j,j}$ and $S_{k,k}$ – the depth of classes j and k , respectively). $F_{j,k}$ - The number of cases observed in class j that were classified as class k . $F_{+,k}$ - The total number of cases classified as class k . $F_{j,+}$ - The total number of cases observed as class j . β - The relative weight of precision and recall (set to 1 for equal weights).