HABITUS 2.1.0

HABITUS Reference Book

Screens, parameters and their effects on results

Seda ÖRÜCÜ & Ömer K. ÖRÜCÜ

1. Introduction and workflow
2. Data I: records, layers and study boundary (① Data)
3. Data II: background, pseudo-absences, accessible area and thinning (① Data)
4. Variables (② Variables)
5. Models I: algorithms and their settings (③ Models → Algorithm Settings)
6. Models II: cross-validation, threshold, ensemble and results (③ Models)
7. Evaluation (④ Evaluation)
8. Future scenarios (⑤ Future Scenarios)
9. Range change (⑥ Range Change)
10. Independent validation (⑦ Validation)
11. Report, ODMAP and multi-species runs (⑧ Report)
12. Map, Help and output files (🗺 Map, ⓘ Help)

Source code references are to HABITUS 2.1.0 · Built: 21 November 2026

Contents

1. Introduction and workflow 7
- 1.1 What HABITUS is 7
- 1.2 How to use this book 7
- 1.3 Installation and platforms 8
- 1.4 Start-up 8
- 1.5 Layout of the main window 9
- 1.6 Tab bar and lock scheme 11
- 1.7 Bottom status bar: progress bar and log 14
- 1.8 Progress, cancellation and closing the window 15
- 1.9 Output folder and automatic analysis folder 16
- 1.10 Random seed and reproducibility 16
- 1.11 Example data 17
- 1.12 Expert questions 18
- 1.13 Common mistakes and pitfalls 19
- 1.14 References 19

2. Data I: records, layers and study boundary (① Data) 20
- 2.1 What this tab does 20
- 2.2 Screen layout 21
- 2.3 Species records: Occurrence Data (CSV) 22
- 2.4 Predictor layers: Continuous Rasters and Categorical Rasters 25
- 2.5 Study boundary: Study Boundary (mask, optional) 28
- 2.6 Reproducibility: Reproducibility 30
- 2.7 Output folder: Output Directory 30
- 2.8 The Load Data button: order of operations and record filtering 31
- 2.9 Outputs and what passes to the next step 34
- 2.10 Expert questions 35
- 2.11 Common mistakes and pitfalls 36
- 2.12 References 37

3. Data II: background, pseudo-absences, accessible area and thinning (① Data) 38
- 3.1 What this chapter covers 38
- 3.2 Screen layout 38
- 3.3 Pseudo-absences: ML Models 39
- 3.4 Your own absence records: Own Absences 43
- 3.5 MaxEnt background: MaxEnt 45
- 3.6 Sampling bias: Sampling Bias 47
- 3.7 ENFA background: ENFA / Mahalanobis 49
- 3.8 Accessible area: Study Area & Thinning (upper half) 50
- 3.9 Thinning the records: Study Area & Thinning (lower half) 53
- 3.10 Outputs and what passes to the next step 55
- 3.11 Expert questions 55
- 3.12 Common mistakes and pitfalls 57
- 3.13 References 57

4. Variables (② Variables) 59
- 4.1 What this tab does 59
- 4.2 Screen layout 60
- 4.3 Which sample the computations use 61

4.4 Analysis settings (Analysis Settings) 61
4.5 Highly correlated pairs and the heatmap 64
4.6 Variable list and bulk selection 65
4.7 Confirmation (Confirm) and the records-per-predictor check 69
4.8 Export 70
4.9 Advanced analysis (Advanced Analysis) 71
4.10 Outputs and what passes to the next step 76
4.11 Expert questions 76
4.12 Common mistakes and pitfalls 78
4.13 References 78

5. Models I: algorithms and their settings (③ Models → Algorithm Settings) 80
5.1 What this tab does 80
5.2 Screen layout 81
5.3 Algorithm selection (Select Algorithms) 82
5.4 Rules common to all algorithms 82
5.5 Guide to choosing algorithms 85
5.6 GLM: generalised linear model 87
5.7 GBM: gradient boosting (scikit-learn) 88
5.8 BRT: boosted regression trees 90
5.9 RF: random forest 91
5.10 SVM: support vector machine 93
5.11 ANN: artificial neural network (multilayer perceptron) 94
5.12 XGBoost: extreme gradient boosting 95
5.13 LightGBM: light gradient boosting 97
5.14 CatBoost: categorical boosting 98
5.15 GAM: generalised additive model 100
5.16 MaxEnt: maximum entropy 101
5.17 ENFA: ecological niche factor analysis 104
5.18 Mahalanobis distance 105
5.19 BIOCLIM: climate envelope 106
5.20 Domain: Gower distance 107
5.21 GLMNET: elastic-net logistic regression 108
5.22 CART: classification tree 109
5.23 ESM: ensemble of small models 110
5.24 Outputs and what passes to the next step 112
5.25 Expert questions 112
5.26 Common mistakes and pitfalls 115
5.27 References 115

6. Models II: cross-validation, threshold, ensemble and results (③ Models) 118
6.1 What this tab does 118
6.2 Screen layout 119
6.3 Algorithm selection (Select Algorithms) 120
6.4 Training settings (Training Settings) 121
6.5 Evaluation settings: threshold, omission reference, Boyce 127
6.6 Ensemble settings (Combining Results) 130
6.7 Tuning search (Tuning (optional)) 134
6.8 Evaluation results (Evaluation Results) 136
6.9 Current distribution map (Current Distribution Map) 137
6.10 Outputs and what passes to the next step 138
6.11 Expert questions 139

6.12 Common mistakes and pitfalls 141
6.13 References 141

7. Evaluation (④ Evaluation) 143
7.1 What this tab does 143
7.2 Screen layout 144
7.3 Model scores (Model Scores) 145
7.4 ROC curves (ROC Curves) 147
7.5 Omission, precision-recall and calibration (Omission & PR) 148
7.6 Variable importance (Variable Importance) 149
7.7 Response curves (Response Curves) 150
7.8 Boyce detail (Boyce Detail) 152
7.9 Classification thresholds (Classification Thresholds) 153
7.10 Bottom buttons: refresh, save 154
7.11 Outputs and what passes to the next step 155
7.12 Expert questions 155
7.13 Common mistakes and pitfalls 157
7.14 References 157

8. Future scenarios (⑤ Future Scenarios) 159
8.1 What this tab does 159
8.2 Screen layout 160
8.3 Information banner 161
8.4 Scenario name and GCM (Projection Scenario) 161
8.5 Algorithms to project (Algorithms to Project) 162
8.6 Raster list and the order rule (Environmental Rasters for This Scenario) 163
8.7 Output and display settings (Output Settings) 164
8.8 Transfer safety options: MESS, MESS mask, clamping 165
8.9 ExDet novelty surfaces (always written) 168
8.10 Grid alignment, clipping to the study boundary and strip-wise projection 169
8.11 Outputs and the Completed Projections table 170
8.12 Map viewer (Projection Map Viewer) 171
8.13 Outputs and what passes to the next step 172
8.14 Expert questions 172
8.15 Common mistakes and pitfalls 174
8.16 References 174

9. Range change (⑥ Range Change) 175
9.1 What this tab does 175
9.2 Screen layout 176
9.3 Selecting the binary maps (Input Binary Projection Rasters) 176
9.4 Quick fill (Quick Fill from Completed Projections) 177
9.5 Analysis name (Output · Analysis name) 177
9.6 Extent within which change is counted (Count change within) 178
9.7 Result codes and writing of the map 179
9.8 Statistics table (Range Change Statistics) 180
9.9 Map 182
9.10 Which ensemble map should be chosen: EMwmean_bin or EMca_bin? 182
9.11 Dispersal assumption 183
9.12 Outputs and what passes to the next step 183
9.13 Expert questions 184
9.14 Common mistakes and pitfalls 185

9.15 References 185

10. Independent validation (⑦ Validation) 186

10.1 What this tab does 186
10.2 Screen layout 187
10.3 Map selection (1 Map Selection) 188
10.4 Comparison mode and reclassification (2 Reclassification) 189
10.5 Sampling design (3 Sampling Settings) 191
10.6 Running (Run Validation) 193
10.7 Metrics and the result text 194
10.8 Export 196
10.9 Expert questions 196
10.10 Common mistakes and pitfalls 198
10.11 References 198

11. Report, ODMAP and multi-species runs (⑧ Report) 200

11.1 What this tab does 200
11.2 Screen layout 200
11.3 Report options (Report Sections) 201
11.4 Generate HTML Report and Open in Browser 202
11.5 Sections of the HTML report 202
11.6 ODMAP export (Export ODMAP protocol) 207
11.7 Publication maps 207
11.8 Multi-species batch run (Run Selected Species…) 208
11.9 Outputs 209
11.10 Expert questions 209
11.11 Common mistakes and pitfalls 211
11.12 References 211

12. Map, Help and output files (🗺 Map, ⓘ Help) 212

12.1 What this chapter covers 212
12.2 The place of the 🗺 Map tab 212
12.3 The 🗺 Map tab 213
12.4 The ⓘ Help tab 216
12.5 File catalogue of the output folder 219
12.6 Run record: habitus_run_config.json 222
12.7 Logs 224
12.8 Environment variables 225
12.9 Troubleshooting 225
12.10 System requirements 228
12.11 Expert questions 228
12.12 Common mistakes and pitfalls 230
12.13 References 230

In memory of our mother ***Nevin Örücü****...*

14 September 2025

1. Introduction and workflow

Scope: main window · Code: `main.py` (start-up), `main_dialog.py` (main window, tab locks, log), `threads.py` (background jobs), `run_config.py` (seed and run record), `tabs/tab_data.py` (seed and output folder fields)

1.1 What HABITUS is

HABITUS is a desktop program for species distribution modelling (SDM) that requires no coding. It takes the coordinates of the locations where a species occurs (presence records, CSV) and environmental raster layers on the same grid; from these it builds a habitat suitability model, evaluates the model by cross-validation, produces maps for current and future climate, computes the change in area between two maps and gathers all steps into an HTML report (Elith and Leathwick, 2009).

Computations are done in Python: with the scikit-learn, xgboost, lightgbm, catboost, pygam and elapid libraries and NumPy. No R installation is needed and no R packages are called. In this book, R packages (biomod2, dismo, ENMeval, blockCV, maxnet, ecospat) are mentioned only for comparison.

Two tasks remain outside the program. The first is the collection and cleaning of the presence records. The second, when no study boundary is given, is bringing the rasters onto a common grid: all continuous and categorical rasters must have the same dimensions, cell size, origin and coordinate reference system (CRS). HABITUS checks this on loading and names the file that differs, but does not resample it (Chapter 2).

> **Note:** The UI is in English. In this book, button, box and tab names are given in `code` format exactly as they appear in the UI.

1.2 How to use this book

The book follows the program's tabs from left to right. Every chapter has the same layout: the tab's place in the workflow, the screen layout, field groups and parameter tables, a four-part explanation for every parameter that affects the result (**What it does**, **When and how to set it**, **Effect on results**, **Caution**), outputs, expert questions, common mistakes and references.

- The default values and ranges in the parameter tables were read from the source code; the `Code` column gives the location in the form `file.py:line`.
- The screenshots and the numbers given in the text come from an example run: *Abies nordmanniana* (626 records), WorldClim 2.1 bioclimatic variables (2.5 minutes), future scenario MPI-ESM1-2-HR SSP5-8.5 2081-2100, algorithms GLM, GAM, RF and MAXENT. These numbers are there to illustrate the UI; they are not a conclusion about the ecology of the species.
- A first-time reader should read this chapter and then Chapters 2-6 in order. A reader looking for the meaning of a setting can go directly to the parameter table of the relevant chapter. The full list of output files, the run record and troubleshooting are in Chapter 12.

Distribution of chapters over the tabs:

Chapter	Tab	Topic
2-3	① `Data`	Records, rasters, study boundary, pseudo-absence and background
4	② `Variables`	VIF, correlation, advanced multicollinearity analysis, variable selection
5-6	③ `Models + Current Map`	Algorithms, cross-validation, threshold, ensemble, Tuning, current map

Chapter	Tab	Topic
7	④ Evaluation	Scores, ROC, omission, variable importance, response curves, Boyce, thresholds
8	⑤ Future Scenarios	Projection to future climate, extrapolation diagnostics
9	⑥ Range Change	Loss, gain and stable area
10	⑦ Validation	Accuracy against independent maps or points
11	⑧ Report	HTML report, ODMAP, batch run on a multi-species file
12	🗺 Map, ⓘ Help	Record map, help, output files, run record, troubleshooting

1.3 Installation and platforms

HABITUS runs on three desktop platforms. The packaged release carries its own Python interpreter and all libraries; no separate Python installation is needed.

Platform	Package	Installation
Windows	Installer (`HABITUS_Setup_v*.exe`) or portable ZIP	Run the installer; the program opens from the Start menu and the desktop shortcut. At the SmartScreen warning, choose `More info` and `Run anyway`
macOS	`.dmg` disk image (Apple Silicon and Intel)	Drag `HABITUS.app` into the `Applications` folder; on first launch, right-click and `Open`
Linux	`AppImage` or `tar.gz`	Give the AppImage file execute permission and run it; in the tar.gz archive the program is started with `run_habitus.sh`
From source	`main.py`	`python main.py` in a Python environment where the dependencies are installed

The minimum Python version required to run from source is 3.10 (`pyproject.toml:11`). If the optional algorithm libraries (xgboost, lightgbm, catboost, pygam, elapid) are not installed, the program still opens; the corresponding algorithms are marked as unavailable in the ③ Models tab (`main.py:365-367`). Detailed system requirements are in Chapter 12.

Self-test. If the packaged program is given the `--selftest` argument (`HABITUS.exe --selftest`, from source `python main.py --selftest`), no window opens; the program tests the coordinate database, a coordinate transformation, vector reading, writing and reading a GeoTIFF, which algorithms can be run, and Qt, and prints a report (`main.py:195-197`, `selftest.py`). If `--require-algorithms GLM,MAXENT` is added, the test is counted as failed when any of the named algorithms is missing (`selftest.py:368-376`). This is the quickest way to make sure an installation works.

1.4 Start-up

When the program starts, a splash screen appears first. Meanwhile the scientific libraries are loaded one by one and the splash screen shows the name of the library being loaded: plotting (matplotlib), tables (pandas), SciPy, raster reading (rasterio), vector layers, machine learning, XGBoost, LightGBM, CatBoost, MaxEnt (elapid) and GAM (`main.py:223-235`). On a cold start this may take a few seconds; loading is deliberately done before the window so that the screen does not stay blank.

At start-up the program also does the following:

- It checks the coordinate transformation database (`proj.db`). If an old database left over from another installation (e.g. PostGIS) is defined in the environment, it selects the program's own copy (`main.py:88-163`).
- On Linux and macOS it resets the character and numeric locale to the neutral C locale, so that under a Turkish locale the raster library does not go wrong in upper/lower case comparisons. The UI is not affected; this step is not applied on Windows (`main.py:293-315`).
- It opens the crash log (Chapter 12).
- It starts a check for a new release in the background; the result appears on the `ⓘ Help` > `About` page (`main_dialog.py:130-148`).

The initial size of the main window is 95% of the screen width and 92% of its height, at most 1600 x 1000 logical pixels (`main_dialog.py:113-121`). The window can be resized; every tab is inside a scroll area, and on a narrow screen the content scrolls (`main_dialog.py:183-193`).

1.5 Layout of the main window

The main window consists of four regions (`main_dialog.py:150-217`):

1. **Title bar.** On the left the program name and subtitle, on the right the `⟳ New Analysis` button (§1.5.1).
2. **Tab bar.** Eight numbered workflow tabs plus `🗺 Map` and `ⓘ Help` (§1.6).
3. **Tab content.** The screen of the selected tab; described in Chapters 2-12.
4. **Bottom status bar.** Progress bar and log panel (§1.7).

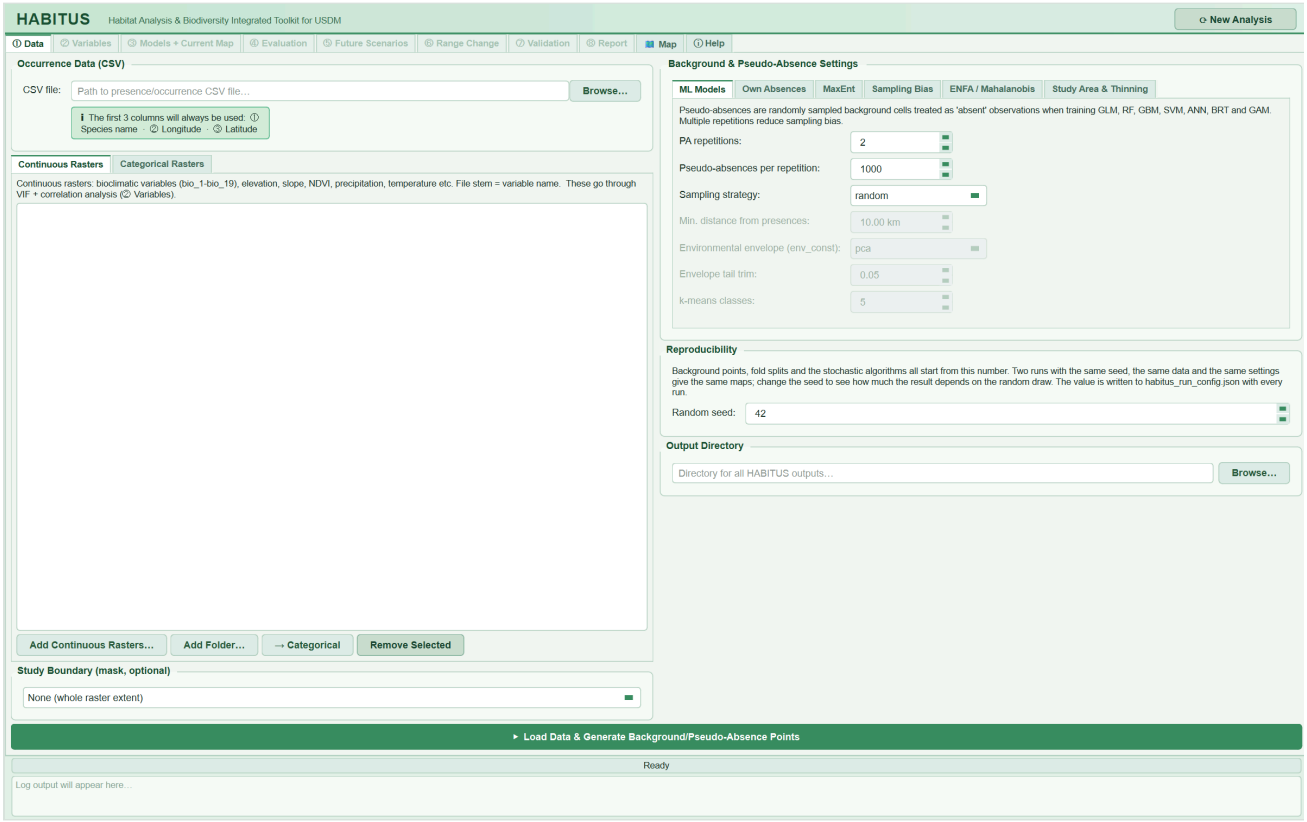


Figure 1.1. The main window when the program is first opened: the title bar at the top with the New Analysis button on the right, below it the ten-tab bar (only ① Data, 🗺 Map and ⓘ Help enabled), the ① Data tab in the middle, and at the bottom the progress bar reading Ready and the empty log panel.

Name in the UI	What it does	Default	Range / options	Code
`⟳ New Analysis`	Asks for confirmation; deletes all data, models and results from memory, locks the tabs, returns to `① Data`	—	button; `Yes` / `No` in the confirmation dialog (default `No`)	`main_dialog.py:159-163`, `241-249`, `303-419`
Tab bar	Workflow steps; tabs after an unfinished step are greyed out and cannot be clicked	`① Data`, `🗺 Map`, `ⓘ Help` open	ten tabs	`main_dialog.py:186-199`
Progress bar	Shows the step and percentage of the running job; green when the job finishes, red on error	`Ready`, 0%	0-100%	`main_dialog.py:205-208`, `659-703`
Log panel	Writes every message with its time; the same lines also go to the session log file	empty (`Log output will appear here…`)	read-only	`main_dialog.py:209-216`, `790-801`
`Random seed` (`① Data` > `Reproducibility`)	Starting number of all random operations	42	0-2147483647	`tabs/tab_data.py:242-249`
`Output Directory` (`① Data`)	Folder to which all outputs are written	empty; filled automatically when a CSV is selected	any folder	`tabs/tab_data.py:253-259`, `1125-1162`
`Output Directory (for saving map files)` (`③ Models`)	Folder to which models and maps are written; if empty, uses the folder in ① Data	automatic from ① Data	any folder	`tabs/tab_models.py:679-686`, `1338-1348`, `1470-1477`

1.5.1 `⟳ New Analysis`

- **What it does.** If `Yes` is answered to the question "This will clear all loaded data, models, and results. Proceed?", the program stops running background jobs (waiting at most 1.5 seconds), resets the shared state, clears the three map viewers, locks all tabs from `② Variables` to `⑧ Report` and returns to `① Data`. In `① Data` the presence CSV path, the raster lists and the `Output Directory` field are emptied; the session log is closed (`main_dialog.py:303-419`).
- **When and how to set it.** Used when switching to another species or to an entirely different data set. It is not needed to change just one setting for the same species and run again; re-running the relevant step is enough (§1.6.2).
- **Effect on results.** Everything in memory is deleted. The settings in the tabs (seed, number of pseudo-absences, algorithm choices, etc.), however, are **not** deleted; the new analysis starts with the previous settings. Because `Output Directory` is emptied, a new analysis folder is created when a new CSV is selected (§1.9).
- **Caution.** Files on disk are not deleted. The folder of the previous analysis remains as it is.

1.6 Tab bar and lock scheme

The tabs follow the order of the workflow. When the program opens, only `① Data`, `🗺 Map` and `ⓘ Help` are enabled; the seven tabs from `② Variables` to `⑧ Report` are locked (`main_dialog.py:198-199`). When a step finishes successfully, the next tab is unlocked and in most cases the program switches to that tab.

Completed step	Tabs unlocked	Where the program switches	Code
`① Data`: `Load Data & Generate Background/Pseudo-Absence Points`	`② Variables`	`② Variables`	`tabs/tab_data.py:1759-1760`
`② Variables`: confirming the variable selection	`③ Models + Current Map`	`③ Models`, `Algorithm Settings` sub-tab	`tabs/tab_vif.py:1206-1208`
`③ Models`: `Run Models + Generate Current Distribution Map`	`④ Evaluation`, `⑤ Future Scenarios`, `⑥ Range Change`, `⑦ Validation`, `⑧ Report` (together)	`Current Distribution Map` sub-tab within `③ Models`	`tabs/tab_models.py:1848-1852`
`⑤ Future Scenarios`: projection	(already open) `⑥ Range Change`, `④ Evaluation`	tab does not change	`tabs/tab_projection.py:672-673`

After `③ Models` has finished, the order `④`-`⑧` is not mandatory: for example, `⑦ Validation` or `⑧ Report` can be used without running `⑤ Future Scenarios`. `⑥ Range Change` needs two binary maps; if there is no future map, only a comparison between current maps is possible (Chapter 9). When the other species of a multi-species file are processed with the batch run in `⑧ Report`, all tabs are unlocked (`batch_species.py:215-217`).

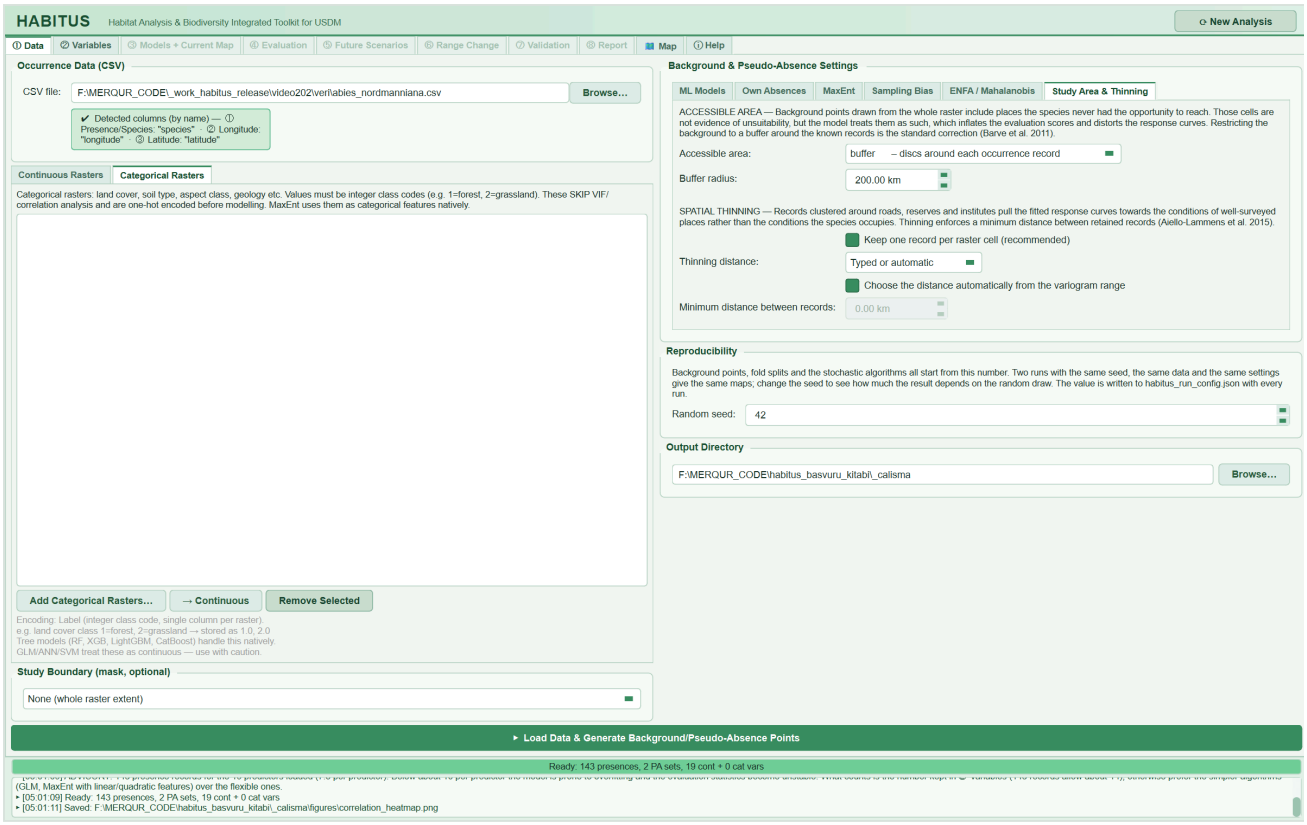


Figure 1.2. The main window after the ① Data step has finished: the ② Variables tab is unlocked, the other numbered tabs are still greyed out; the progress bar is green and reads "Ready: 143 presences, 2 PA sets, 19 cont + 0 cat vars"; time-stamped messages are visible in the log panel (example run).

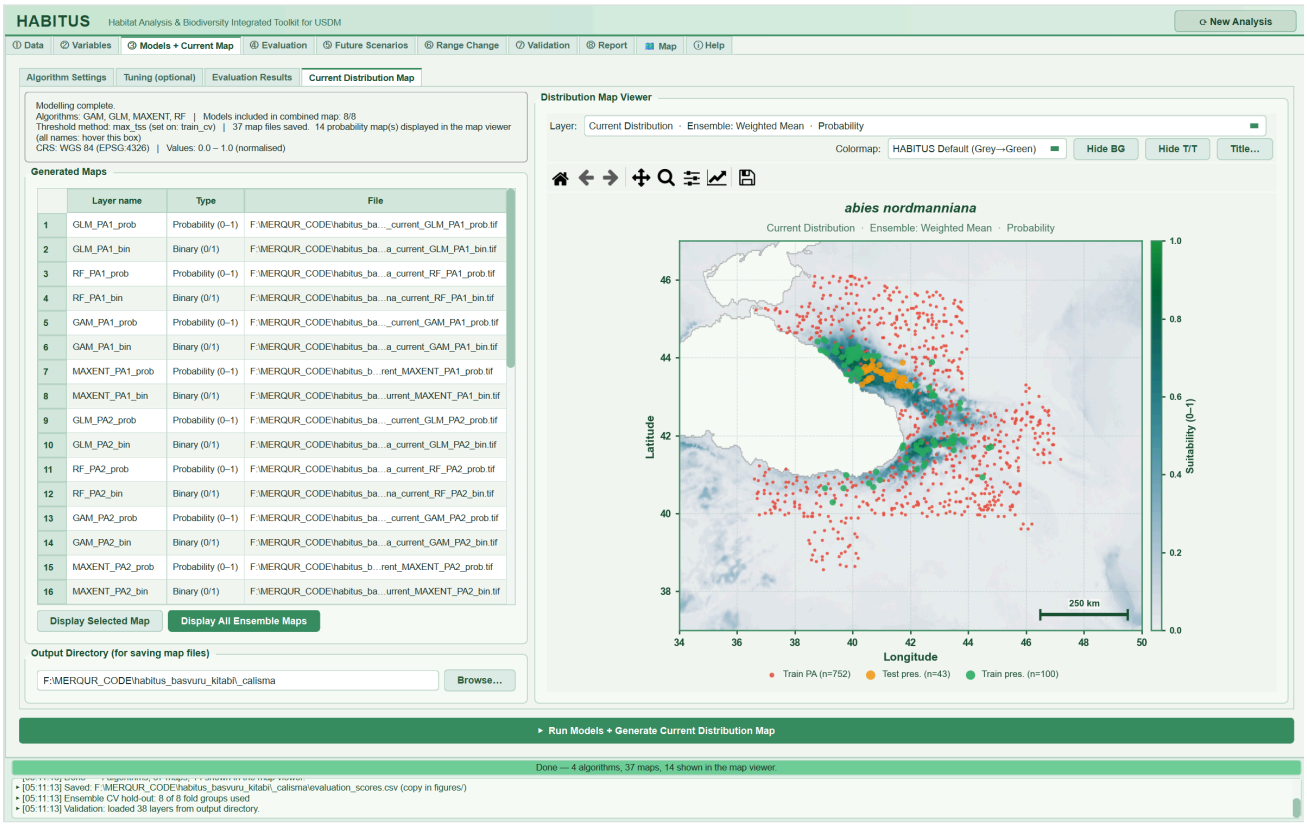


Figure 1.3. The main window after the ③ Models step has finished: all tabs from ④ Evaluation to ⑧ Report are unlocked; the progress bar reads "Done — 4 algorithms, 37 maps, 14 shown in the map viewer"; the Output Directory (for saving map files) field has been filled automatically with the folder from ① Data (example run).

1.6.1 Tab bar summary

The whole workflow is in the following table. The "Main output" column gives the principal files written to the output folder; the full catalogue is in Chapter 12.

Tab	What it does	Main output	Chapter
① Data	Reads and filters the records (one record per cell, thinning), loads the rasters and checks the grid, defines the accessible area, generates pseudo-absence and MaxEnt background points	`accessible_area.tif`, optional `background_maxent.csv`; training tables in memory	2-3
② Variables	Ranks variables by VIF and correlation; the user selects and confirms continuous and categorical variables; optional condition number, PCA, LASSO, Ridge	correlation heat maps, VIF plot, `high_correlation_pairs.csv`, `variable_priority_ranking.csv` in `figures/`	4
③ Models + Current Map	Trains the selected algorithms with cross-validation, sets the threshold and builds the ensemble, writes the current-climate map; optional Tuning	probability, binary and uncertainty rasters in the `current/` folder; `evaluation_scores.csv`; `occurrence_train.csv`, `occurrence_test.csv`; `habitus_run_config.json`	5-6
④ Evaluation	Scores, ROC, omission and calibration, variable importance, response curves, Boyce, threshold tables	`figures/evaluation_*.png`, `figures/classification_thresholds.csv`	7
⑤ Future Scenarios	Applies the models to future climate rasters; extrapolation maps (ExDet, optional MESS)	probability, binary, NT1, NT2, MIC rasters in the `<scenario>/` folder; `projection_options.json`	8
⑥ Range Change	Compares two binary maps: loss, gain, stable	`<species>_<analysis name>_SRC.tif` and area table	9
⑦ Validation	Compares the model with an independent reference map or points	reclassified rasters in the `validation/` folder; optional report and point file	10
⑧ Report	Gathers all steps into a scientific HTML report; exports the ODMAP protocol; runs the other species of a multi-species file	`<species>_HABITUS_report_<time>.html`, `odmap_protocol.md/.csv`, `batch_summary.csv`	11
🗺 Map	Shows the inputs (records, excluded records, own absences, boundary) on an online basemap; study area figure for publication	`<species>_study_area.png` (optional)	12
ⓘ Help	English user guide and update check	—	12

1.6.2 Re-running a step deletes the later results

HABITUS does not keep the results that follow a step with the old input when the input of that step changes. There are two deletion paths:

- **If ① Data is re-run** (when data have been loaded before), the program, without asking for confirmation, stops running model, projection and change jobs, deletes the variable selection, models, ensemble, projections, change results, Tuning results, evaluation plots and validation results held in memory, and locks all tabs from ② Variables to ⑧ Report. "Data inputs changed — all previous results cleared. Re-running data formatting…" is written to the log (tabs/tab_data.py:1396-1397, main_dialog.py:421-526).
- **If the selection in ② Variables is re-confirmed** and the models have already been run, the "Re-confirm Variable Selection" dialog asks: "Confirming now will discard all existing models, maps, projections, and evaluation results so everything is recomputed from scratch." If Yes is answered, the models and all later results are deleted; ① Data, ② Variables and ③ Models stay open, ④-⑧ are locked. "Variable selection changed — downstream results cleared. Re-run Models to continue." is written to the log (tabs/tab_vif.py:1170-1185, main_dialog.py:528-594).

When ③ Models is re-run, the tabs are not locked; the new results replace the old ones.

> **Caution:** Deletion applies only to the program's memory. Files in the output folder are not deleted; a new run overwrites files with the same name. Rasters left from the first run by an algorithm not selected in the second run, and old PNGs in figures/, remain in the folder. Because ⑧ Report reads all PNGs in figures/ and ⑦ Validation reads the layers in the output folder (tabs/tab_report.py:578-587), using a new output folder for every run with changed settings is the safest course.

1.7 Bottom status bar: progress bar and log

Progress bar

While a long job is running, the bar shows the current step and the percentage (e.g. "Reading raster 3 of 19: bio_11.tif 12%"). The bar only moves forward during the job; it reaches 100% only when the job has really finished. On completion the bar turns green and summarises the result (e.g. "Done — 4 algorithms, 37 maps, 14 shown in the map viewer"); on error it turns red and reads "Failed — see the message" (main_dialog.py:659-703, 755-788).

While a job is running, the button that started it is disabled and its text changes; a second click does not start a second job. The button is re-enabled when the job ends successfully, with an error or by cancellation (main_dialog.py:707-788). The ⑦ Validation and ⑧ Report jobs run in the UI thread; during this time a small "HABITUS" window shows the current step and the cursor changes to the wait cursor (main_dialog.py:712-753).

Log panel

Every message of the program appears in the log panel in the form ▸ [HH:MM:ss] message and the panel automatically scrolls to the last line (main_dialog.py:790-793). The log contains: the progress messages of every step, warnings (⚠, WARNING:), informational notes that affect interpretation (ADVISORY:), the path of every saved file (Saved: …) and the last line of errors (ERROR: …).

From the moment the output folder is known, the same lines are also written to the **session log** file (main_dialog.py:612-645, 794-801):

- The file name is `<species>_<YYYYMMDD_HHMMSS>.log`. If the species name is not known at that moment (e.g. if the output folder was selected manually), the name becomes `habitus_<YYYYMMDD_HHMMSS>.log`. The log file of the example run is named `habitus_20261121_050108.log`.
- The file begins with "HABITUS Analysis Log", the species name and the start time; `New Analysis` or closing the window closes the file with a "Closed" line.
- If the output folder changes (e.g. the next species in a multi-species file), the old log is closed and a new one is opened in that folder.
- When an error occurs, the full error message is appended to the log under a `--- ERROR ---` header; the error dialog also shows the path of the log file (`main_dialog.py:806-827`).

The error dialog ("HABITUS Error") shows the last line of the error as its title and, for recognised errors, a plain suggestion. `Show details` opens the full text, `Copy full message` copies all of it to the clipboard (`error_dialog.py:30-56`, `80-182`). When reporting a problem, this text and the session log should be sent together.

1.8 Progress, cancellation and closing the window

Data preparation, modelling, projection, range change, Tuning, raster folder scanning and the update check run in a background thread; the window does not freeze meanwhile and other tabs can be browsed (`threads.py:1-23`).

Cancellation is cooperative: when a stop is requested, a flag of the job is set and the job stops at its next progress report; a cancelled job reports neither a result nor an error (`main_dialog.py:37-74`, `threads.py:34-63`). There is no separate "stop" button for the `③ Models`, `① Data` and `⑤ Future Scenarios` jobs; a running job is stopped in these ways:

- `↻ New Analysis` (waits at most 1.5 seconds),
- re-running `① Data` or re-confirming the variable selection (stops the downstream jobs),
- closing the window.

The `Tuning (optional)` sub-tab has its own `Cancel` button (Chapter 6); the batch figure saving in `④ Evaluation` can also be cancelled from the progress dialog (Chapter 7).

If the window is to be closed while a job is running, the program asks: "Data formatting, Models still running — close anyway? The unfinished work is discarded." (with the names of the running jobs; default `No`). If `Yes` is answered, the jobs are cancelled and the program waits at most a few seconds (`main_dialog.py:284-301`, `596-608`).

> **Note:** Cancellation takes effect at a progress report. A single long computation (e.g. training a single model on a large data set) continues in the background until it finishes; its result does not reach the window, but it uses the processor for that time (`main_dialog.py:266-282`).

1.9 Output folder and automatic analysis folder

All outputs are written to a single folder: `① Data` > `Output Directory`.

- **Automatic folder.** If a presence CSV is selected while `Output Directory` is empty, the program creates a folder named `Documents/HABITUS/<species>_<YYYYMMDD_HHMMSS>` in the user's documents folder and writes it into the field; "Analysis folder auto-created: <path>" appears in the log. The species name is the most frequent value in the species column of the CSV (in a GBIF file the `species` column is recognised by its name); for a multi-species file the folder takes the name of the CSV file. If there is no documents folder, the home folder is used; characters other than letters, digits, dots and hyphens in the name are converted to underscores (`tabs/tab_data.py:1125-1162`, `main_dialog.py:831-853`).
- **Manually selected folder.** A folder selected with `Browse…` or typed in manually is used as it is; no automatic folder is created.
- **Multi-species file.** Each species is written to its own subfolder under the output folder (`tabs/tab_data.py:1115-1123`; Chapter 2).
- **`③ Models` folder.** The `Output Directory (for saving map files)` field in `③ Models` is filled automatically with the folder from ① Data. If the user enters another folder here, the modelling outputs and all subsequent steps (and the session log) move to that folder (`tabs/tab_models.py:1338-1348`, `1470-1477`).

1.9.1 `Output Directory`

- **What it does.** Sets where raster maps, tables, figures, the run record, the session log and the report are written. If it is empty, `Load Data & Generate Background/Pseudo-Absence Points` does not run and shows the message "Please specify an output directory." (`tabs/tab_data.py:1370-1371`).
- **When and how to set it.** A separate folder is recommended for each analysis; the automatic folder provides this by itself with a time stamp. If comparative runs with different settings are to be made for the same species, each run should be given a new folder.
- **Effect on results.** It does not change the computed numbers. But writing to the same folder a second time overwrites `habitus_run_config.json`, `evaluation_scores.csv` and the rasters, and files left over from the previous run may appear in the report (§1.6.2).
- **Caution.** If `evaluation_scores.csv` has been edited manually, it is backed up as `evaluation_scores.<time>.bak.csv` before being overwritten (`sdm_core.py:6764-6776`); other files are not backed up. If a file in the output folder is open in another program (e.g. a GIS program or a spreadsheet), writing may fail with "Permission denied".

1.10 Random seed and reproducibility

The `Random seed` in the `① Data` > `Reproducibility` group is the single starting number of the whole analysis. The description in the UI: "Background points, fold splits and the stochastic algorithms all start from this number. Two runs with the same seed, the same data and the same settings give the same maps; change the seed to see how much the result depends on the random draw." (`tabs/tab_data.py:234-249`).

1.10.1 `Random seed`

- **What it does.** When `① Data` is run, the seed is applied before anything random is done (`tabs/tab_data.py:1505-1510`, `sdm_core.py:115-119`). NumPy's global random number generator and Python's `random` module are initialised with this number; if the `PYTHONHASHSEED` environment variable is not defined, it is set to this number (`run_config.py:31-50`). Background and pseudo-absence sampling, the assignment of cross-validation folds and all stochastic algorithms (RF, GBM, BRT, MaxEnt, etc.) are derived from the same value. When modelling has finished, the seed is written to the `random_seed` field of the `habitus_run_config.json` file (`tabs/tab_models.py:2071`, `run_config.py:141`).
- **When and how to set it.** For publication, the default 42 or any fixed value is sufficient; what matters is that the value is reported. To see how much the result depends on the random draw, the same analysis can be repeated with several different seeds (e.g. 1, 2, 3) and the scores and maps compared.
- **Effect on results.** Two runs with the same data, the same settings and the same seed give the same pseudo-absence points, the same folds and the same maps. If the seed changes, the points and folds change; the scores and maps vary by the amount of this variability. In species with few records this variation can be marked.
- **Caution.** The seed is read when `① Data` is run. After changing the seed, re-running only `③ Models` does not apply the new seed: `① Data` must be re-run (which also deletes the later results, §1.6.2). Equality at bit level is not guaranteed; in multi-threaded computations (e.g. RF) the order of summation can produce small differences in the last digit.

1.11 Example data

A ready-made example data package (`habitus_sample.zip`) for trying out the program can be downloaded from the program's release page. The package contains the presence records of one species, the 19 bioclimatic variables, additional layers such as elevation and NDVI, categorical layers (aspect, slope), several climate scenarios and periods, and a reference distribution map for validation (`README.md`, "Sample Data"). The `ornek_veri` folder in the source tree additionally contains *Pinus brutia* records together with two **synthetic** absence files generated by rules for trying out the `Own Absences` sub-tab, and a calibration area polygon; the synthetic files are not field data and must not be used in an analysis intended for publication (`ornek_veri/OKUBENI.md`).

The screenshots in this book, however, were taken with the *Abies nordmanniana* data (§1.2).

Shortest path for a first try. (1) In `① Data`, select the CSV with `Browse…`; the output folder is created automatically. (2) Add the bioclimatic folder with `Add Folder…`. (3) Leave the other settings at their defaults and press `Load Data & Generate Background/Pseudo-Absence Points`. (4) In `② Variables`, confirm the suggested variables. (5) In `③ Models`, select a few algorithms and press `Run Models + Generate Current Distribution Map`. (6) In `⑧ Report`, generate the report.

1.12 Expert questions

Q1. Do two runs with the same data and the same settings really give the same map? Answer: By design, yes. The single seed is applied at the start of `① Data` to NumPy's global generator, to Python's `random` module and (if not defined) to `PYTHONHASHSEED` (`run_config.py:31-50, tabs/tab_data.py:1510`); the algorithms take the same value through their own `random_state` parameter (`run_config.py:38-40`). Equality at bit level is not guaranteed: in multi-threaded computations the order of summation can make a difference in the last digit. No bit-by-bit comparison of two runs was made for this book. In practice, the result obtained with the same release and the same library versions should be identical; the library versions are written to `habitus_run_config.json` > `environment` (`run_config.py:61-74`).

Q2. What happens if I change the seed and re-run only the models? Answer: The new seed is not applied. The seed is read only when `① Data` is run (`tabs/tab_data.py:1505-1510`); `③ Models` uses the value stored in the `formatter` object and also writes that value to the run record (`tabs/tab_models.py:1535`, `2071`). For a seed sensitivity test, `① Data` must be re-run for each seed.

Q3. Does the program replace the R packages; does it give the same result as biomod2? Answer: HABITUS does not call R; the algorithms are written with Python libraries (`main.py:223-235`). The method definitions have been kept close to their counterparts in biomod2, dismo, ENMeval, maxnet and blockCV, but the defaults and implementations are not identical; the differences at algorithm level are described in Chapters 5-7. Instead of expecting "the same result", one should check whether the methods are defined in the same way. It is known that the choice of algorithm can change the result substantially (Qiao et al., 2015); this is why multiple algorithms and an ensemble are recommended.

Q4. Which results does re-running a step invalidate? Answer: Re-running `① Data` deletes all downstream results including the variable selection and locks `②`-`⑧` (`main_dialog.py:421-526`); re-confirming the variable selection deletes the models and everything after them and locks `④`-`⑧` (`main_dialog.py:528-594`). This prevents a result produced with the old input from being mixed with the new input. Deletion is in memory; files on disk remain (§1.6.2).

Q5. Why do five tabs open at once when `③ Models` finishes; can a projection be made without evaluation? Answer: Yes. When `③ Models` finishes, `④`-`⑧` are unlocked together (`tabs/tab_models.py:1848-1852`). The program does not treat evaluation as a mandatory intermediate step, because the evaluation scores are already computed by cross-validation during modelling and written to `evaluation_scores.csv` (`tabs/tab_models.py:1841-1846`). `④ Evaluation` produces the plots of these scores and additional diagnostics. Scientifically, evaluating the model before projection is of course necessary; the program does not enforce this and leaves it to the user.

Q6. How is a long computation stopped; does a stopped job leave partial files? Answer: There is no separate stop button for the model, data and projection jobs; `New Analysis`, re-runs that delete downstream results, or closing the window cancel the job (`main_dialog.py:266-282`). Cancellation takes effect at the next progress report and the job reports no result (`main_dialog.py:54-74`). Files written to disk up to that moment are not deleted; the folder of an interrupted projection may contain some rasters. For this reason the folder of a cancelled run should not be used for a new run.

Q7. Which records of the study can be submitted to a reviewer? Answer: Three files together: `habitus_run_config.json` (settings, those left at default, seed, library versions; `run_config.py:137-152`), the session log (`<species>_<time>.log`; step-by-step messages, warnings and errors; `main_dialog.py:612-645`) and the ODMAP protocol produced by `⑧ Report` (Zurell et al., 2020; Chapter 11). Details are in Chapter 12.

Q8. Are the `ADVISORY:` lines in the log panel errors? Answer: No. These are informational notes that affect the interpretation of the result; for example, that sampling bias has not been corrected, that the cell size is

coarse for the local habitat, or that the number of records per predictor is small. They are written to the log when ① Data finishes; if there is a note other than the default sampling bias note, a warning dialog also opens (tabs/tab_data.py:1743-1755). The notes are also written to habitus_run_config.json > notes; the example run has four notes.

Q9. What happens if the program runs out of memory with large rasters? Answer: The error dialog shows the suggestion "The analysis ran out of memory. Reduce the raster extent or resolution, or lower the number of background points." (error_dialog.py:39-41). Large projections are done in strips (sdm_core.py:7512-7526; Chapters 8 and 12). The measure that reduces memory most is clipping the rasters to the study region (① Data > Study Boundary, Chapter 2).

1.13 Common mistakes and pitfalls

- **Changing the seed and re-running only the models.** The seed is applied in ① Data; for a new seed, ① Data must be re-run.
- **Writing runs with different settings to the same folder.** Files are overwritten, and PNGs and rasters left from the old run may enter the report and the validation lists. Use a new folder for each run; selecting a CSV after New Analysis creates a new automatic folder.
- **Assuming that New Analysis resets the settings.** Only the data, results, file paths and the folder field are cleared; the seed, pseudo-absence and algorithm settings remain.
- **Entering another path in the folder field in ③ Models.** All subsequent outputs and the session log go to that folder; the ① Data outputs (accessible_area.tif, correlation figures) remain in the first folder.
- **Assuming that the window has frozen.** Long jobs run in the background; if the progress bar and the log are being updated, the job is still running.
- **Closing the error dialog and losing the text.** Copy full message copies the full text; the text is also in the session log.
- **Using the system Python when running from source.** If the required libraries are missing, the program stops with an error of the type "No module named 'rasterio'"; an environment in which the dependencies are installed must be used.

1.14 References

Elith, J., Leathwick, J.R., 2009. Species distribution models: ecological explanation and prediction across space and time. Annual Review of Ecology, Evolution, and Systematics 40, 677-697. https://doi.org/10.1146/annurev.ecolsys.110308.120159

Örücü, Ö. K., & Örücü, S. (2026). HABITUS: Habitat Analysis and Biodiversity Integrated Toolkit for Unified Species Distribution Modelling. Ecological Perspective, [Technical Report]. https://doi.org/10.53463/ecopers.20260435

Qiao, H., Soberón, J., Peterson, A.T., 2015. No silver bullets in correlative ecological niche modelling: insights from testing among many potential algorithms for niche estimation. Methods in Ecology and Evolution 6, 1126-1136. https://doi.org/10.1111/2041-210X.12397

Zurell, D., Franklin, J., König, C., Bouchet, P.J., Dormann, C.F., Elith, J., et al., 2020. A standard protocol for reporting species distribution models. Ecography 43, 1261-1277. https://doi.org/10.1111/ecog.04960

2. Data I: records, layers and study boundary (① Data)

Tab: ① Data · Code: `tabs/tab_data.py` (UI), `sdm_inputs.py` (CSV reading), `sdm_mask.py` (folder scanning, clipping to the boundary), `sdm_core.py` `DataFormatter` (loading and record filtering)

2.1 What this tab does

① Data is the first and mandatory step of the workflow. It reads the species records (presence points), the predictor raster layers and an optional study boundary; it filters the records; it then generates the pseudo-absence sets for the machine learning models, the background points for MaxEnt and a separate background sample for ENFA. This chapter covers the "data" half of the tab: the record file, the layers, the boundary, the seed, the output folder and every operation that `Load Data` performs on the records. The background, pseudo-absence, accessible area and thinning settings are in Chapter 3.

- **Input:** an occurrence CSV (species name, longitude, latitude), at least one continuous raster, optional categorical rasters and a boundary layer.
- **Output:** the in-memory `DataFormatter` object (predictor arrays, filtered presences, pseudo-absence sets, backgrounds), counts written to the log and to the run record (`habitus_run_config.json`); if a boundary is given, clipped layers (`masked_inputs/current/`); if an accessible area is defined, `accessible_area.tif`.
- **Lock order:** when the step finishes successfully, the ② Variables tab is unlocked and the program switches to it (`tabs/tab_data.py:1759-1760`). If ① is run again, all results of the later tabs are reset (`tabs/tab_data.py:1396-1397`).

2.2 Screen layout

The tab consists of two columns (`tabs/tab_data.py:78-82`). The left column holds the record file, the raster lists and the study boundary; the right column holds the background/pseudo-absence settings, reproducibility and the output folder. At the bottom is the green button that starts the whole step, with the progress bar and the session log below it.

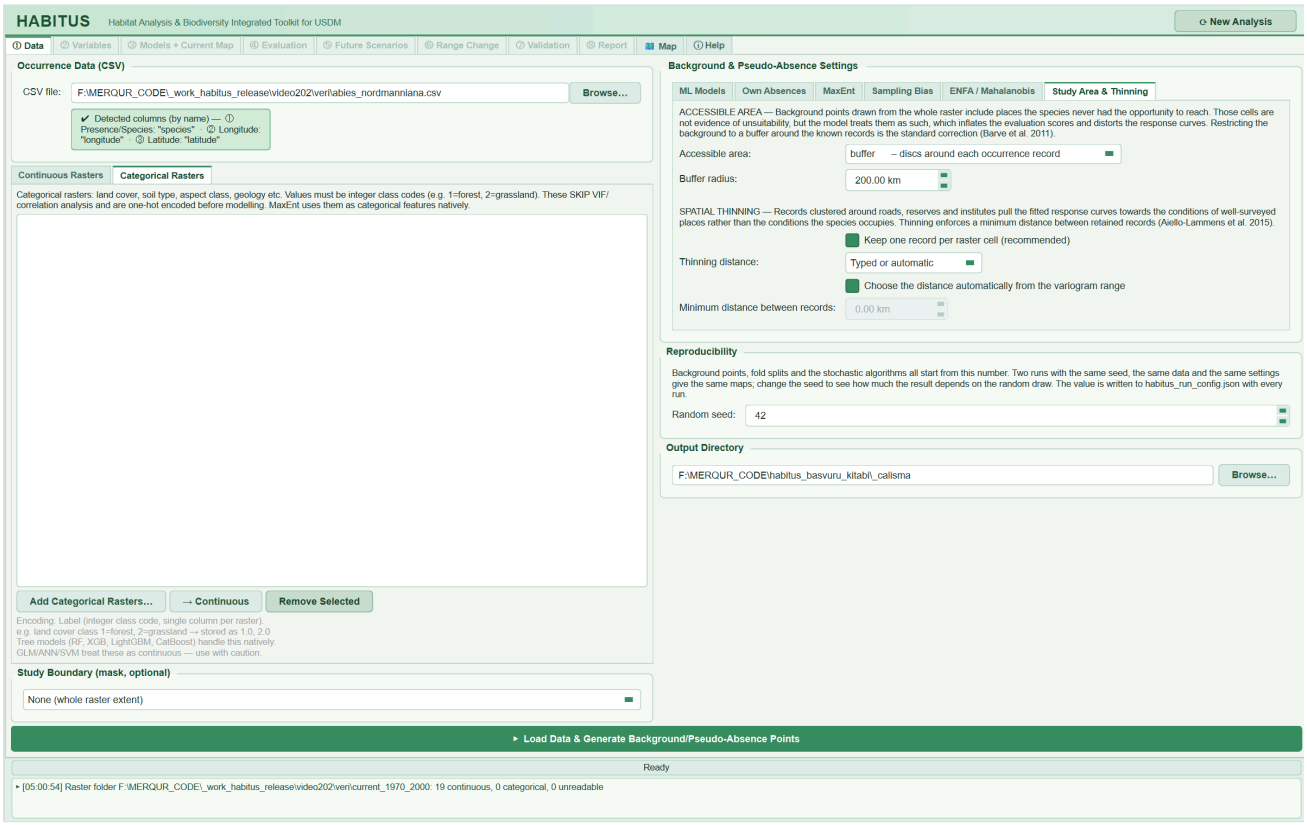


Figure 2.1. The ① Data tab with sample data loaded, before the button is pressed. Left: Occurrence Data (CSV), the Continuous/Categorical Rasters sub-tabs and Study Boundary; right: Background & Pseudo-Absence Settings (here the Study Area & Thinning sub-tab is open), Reproducibility and Output Directory; bottom: the Load Data button and the log.

Region	Name in the UI	In this book
Top left	`Occurrence Data (CSV)`	§2.3
Middle left	`Continuous Rasters`, `Categorical Rasters`	§2.4
Bottom left	`Study Boundary (mask, optional)`	§2.5
Top right	`Background & Pseudo-Absence Settings` (6 sub-tabs)	Chapter 3
Middle right	`Reproducibility`	§2.6
Bottom right	`Output Directory`	§2.7
Bottom bar	`Load Data & Generate Background/Pseudo-Absence Points`	§2.8, §2.9

2.3 Species records: Occurrence Data (CSV)

This group reads the table holding the coordinates of the places where the species was observed. When a file is selected, the program inspects the first rows, identifies the columns, states in a green information box which column will be used as what and, if the file contains more than one species, opens a species list.

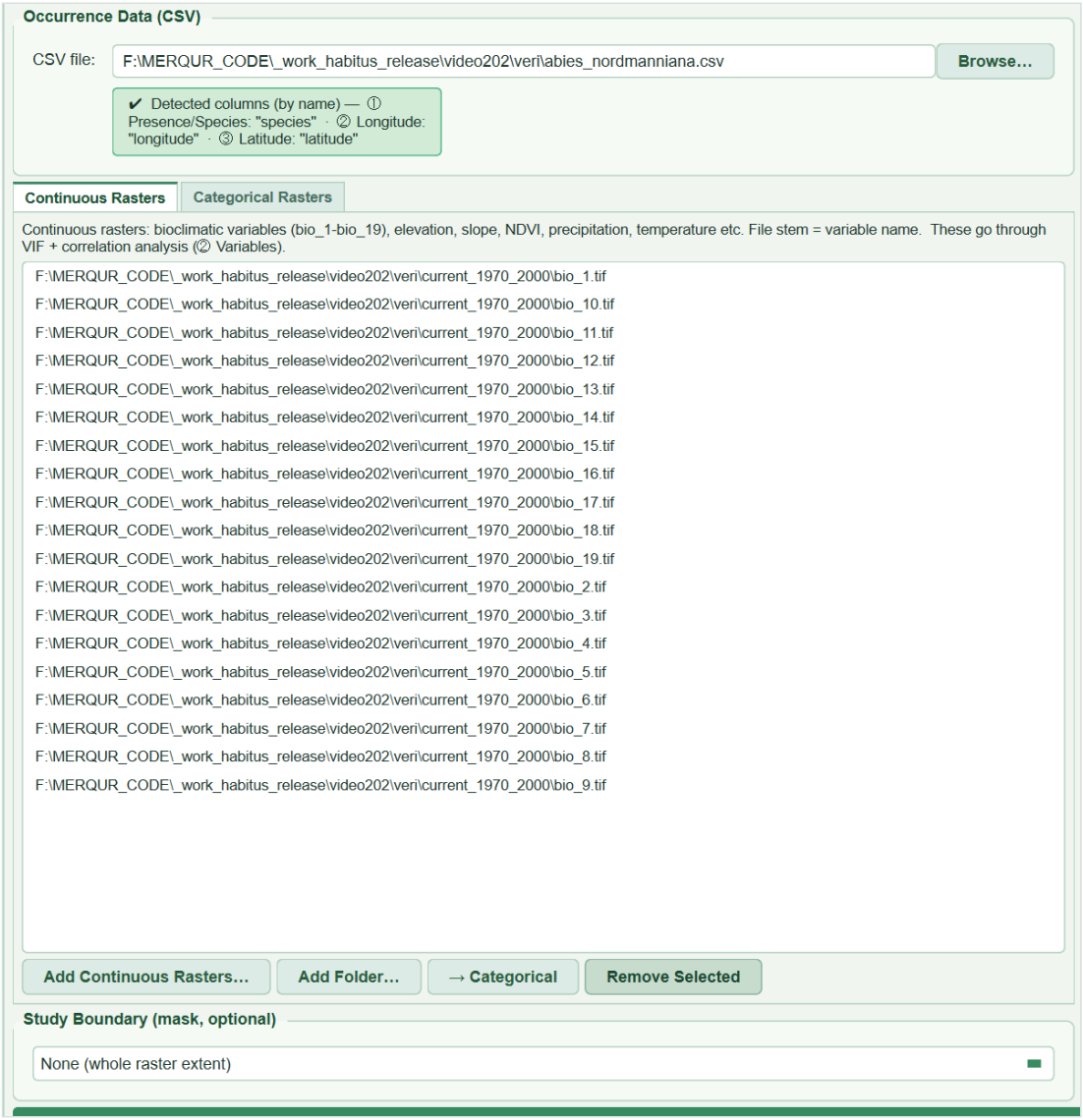


Figure 2.2. Left column: the selected record file and the "Detected columns (by name)" information box (species, longitude, latitude found by name); below it the 19 WorldClim bioclimatic layers in the Continuous Rasters list and the Add Continuous Rasters..., Add Folder..., → Categorical, Remove Selected buttons.

Name in the UI	What it does	Default	Range / options	Code
`CSV file:` + `Browse…`	Selects the record table; on selection scans the columns and species and suggests an output folder	empty	`.csv`, `.tsv`, `.txt`	`tabs/tab_data.py: 92-96`, `972-979`
Information box	Shows the detected species, longitude and latitude columns and the columns that will be ignored	—	—	`tabs/tab_data.py: 99-108`, `1008-1023`

Name in the UI	What it does	Default	Range / options	Code
`Species:` list	In a multi-species file, one checkbox per species (with its record count)	all ticked	`All` / `None`	`tabs/tab_data.py: 112-149`, `1045-1072`
`Analyse now:`	Selects the species for which the ①–⑧ workflow will now run	first ticked species	ticked species	`tabs/tab_data.py: 132-140`, `1085-1095`

2.3.1 File format, delimiter and decimal mark

- **What it does.** The program reads every CSV with the same routine (`sdm_inputs.py:20-65`). The first 50 non-empty lines are taken from the first 64 KB of the file; the count of commas, semicolons and tabs is computed for each line, and the character that occurs the same number of times in most lines is taken as the delimiter. If the delimiter is a semicolon or a tab and the body contains numbers of the form `36,52` while there is no dot decimal at all, the decimal mark is taken to be the comma. The file is read as UTF-8; a leading byte order mark (BOM) is discarded.
- **When and how to set it.** There is no setting; a `;`-delimited, comma-decimal file saved by a spreadsheet with a Turkish, German or Spanish regional setting is read as is. The safest format is still comma delimiter and dot decimal.
- **Effect on results.** A wrongly detected delimiter collapses the whole table into a single column; in that case the program issues the warning "CSV has only 1 column(s)" and no coordinates are read.
- **Caution.** In a comma-delimited file, if coordinates are written with a comma inside quotes, such as `"36,52"`, the decimal comma is not detected (detection is done only for `;` and tab) and the coordinate column remains text. Any coordinate cell that cannot be converted to a number stops the run with the row number: "column '…' holds non-numeric coordinates in N row(s) (row 5: '36°31′'…)" (`sdm_inputs.py:100-123`). Empty coordinate cells, however, do not stop it; those records are dropped with the reason "missing coordinate" and a warning is written (`sdm_core.py:2944-2953`). A non-UTF-8 file is rejected with the message "Re-save it as 'CSV UTF-8'" (`sdm_inputs.py:90-93`).

2.3.2 Column detection (by name, by position, GBIF)

- **What it does.** `occurrence_columns` first looks at the names (`sdm_inputs.py:128-166`). Case-insensitively, `longitude, decimallongitude, lon, long, lng, x` are searched for longitude and `latitude, decimallatitude, lat, y` for latitude. If both are found, the species column is the first of the names `species, scientificname, acceptedscientificname, verbatimscientificname, taxon`; if none of these exists, the first non-coordinate column is taken. If the coordinate names cannot be found, the first three columns are read by position: ① species, ② longitude, ③ latitude. The suffix "(by name)" in the information box shows that the columns were found by name. If the file has more than three columns, the extra ones are ignored and an information window lists them one by one ("Extra Columns Detected", `tabs/tab_data.py:1030-1041`).
- **When and how to set it.** A Darwin Core table downloaded from GBIF (`gbifID, …, species, …, decimalLongitude, decimalLatitude`) can be used without editing. In your own table, naming the columns `species, longitude, latitude` is the clearest route; there is no separate field in the UI for selecting columns.
- **Effect on results.** Swapping longitude and latitude throws most records outside the raster. The program tries to detect this situation (§2.8.6).
- **Caution.** When no names are found, the order is strictly "species, longitude, latitude"; a file in the order "species, latitude, longitude" is read the wrong way round. If only columns named `x` and `y` exist, they are also taken as coordinates. The initial information text in the UI ("The first 3 columns will always be used") appears before a file is selected; once a file is selected, the columns actually used are written, and this is what counts.

2.3.3 Multi-species file and `Analyse now`

- **What it does.** If the species column holds more than one name, the `Species:` row opens; each species is listed as a checkbox with its record count (`tabs/tab_data.py:1045-1072`). If all values in the column are numbers (e.g. a 0/1 presence code), they are not treated as species names and the list does not open. The `Analyse now:` box selects one of the ticked species. When `Load Data` is pressed, only that species' records are taken, written as `occurrences_used.csv` to a subfolder named after the species inside the output folder, and the workflow continues with this file (`tabs/tab_data.py:1500-1506`).
- **When and how to set it.** To model many species with the same settings: first complete ①–⑧ for one species, then `Run Selected Species` in ⑧ Report applies the same settings to the other ticked species (Chapter 11).
- **Effect on results.** Each species is modelled separately, with its own records; no information is shared between species. Subfolder names are made file-system safe and de-duplicated case-insensitively ("Quercus robur" and "quercus robur" go to separate folders; `tabs/tab_data.py:34-60`).
- **Caution.** If no species is ticked, the run stops with the message "The occurrence file holds several species. Tick at least one and choose it under 'Analyse now'." (`tabs/tab_data.py:1373-1378`). Leading and trailing spaces in species names are trimmed; "Abies nordmanniana" and "Abies nordmanniana" (two spaces) are still counted as two different species.

2.3.4 Species name and suggested output folder

The species name used in reports and file names is determined as follows (`sdm_core.py:1566-1593`): if the first column is neither a coordinate nor the species column and is not numeric, the most frequent text in it; otherwise the most frequent text in the species column; failing that, the name of the CSV file (underscores and hyphens converted to spaces). In the sample run the name is "abies nordmanniana". If `Output Directory` is empty, a folder `Documents/HABITUS/<species>_YYYYMMDD_HHMMSS` is created as soon as the file is selected and written into the field (`tabs/tab_data.py:1125-1160`, `main_dialog.py:832-853`); for a multi-species file the folder name is taken from the CSV name.

2.4 Predictor layers: Continuous Rasters and Categorical Rasters

Predictors are kept in two separate lists. **Continuous** layers are measured quantities: temperature, precipitation, elevation, slope, NDVI. **Categorical** layers are class codes: land cover, soil type, geology. This distinction affects every subsequent step: continuous layers enter the VIF and correlation analysis in ② Variables, while categorical layers are excluded from that analysis and are passed to the models as classes.

Name in the UI	What it does	Default	Range / options	Code
`Add Continuous Rasters…`	Adds the selected files to the continuous list (the same path is not added twice)	—	`.tif .tiff .asc .img .nc .vrt`	`tabs/tab_data.py:911-912`, `1183-1191`
`Add Folder…`	Reads all rasters in the folder and classifies each as continuous/categorical	—	folder	`tabs/tab_data.py:913-919`, `1205-1290`
`→ Categorical` / `→ Continuous`	Moves the selected files to the other list	—	—	`tabs/tab_data.py:920-922`, `948-950`, `1197-1203`
`Remove Selected`	Removes the selected files from the list	—	—	`tabs/tab_data.py:923-924`, `1193-1195`
`Add Categorical Rasters…`	Adds the selected files to the categorical list	—	same extensions	`tabs/tab_data.py:946-947`
Encoding note	States that the categorical encoding is label (integer class code)	label	label only	`tabs/tab_data.py:959-968`, `sdm_core.py:1428-1440`

2.4.1 Adding files individually and variable names

- **What it does.** The name of each file without its extension becomes the variable name. The name is cleaned by converting spaces, parentheses, hyphens and dots to underscores: `bio_ (1)` → `bio_1`, `tr-elev` → `tr_elev` (`sdm_core.py:166-186`). Whatever order the layers were added to the list in, the program sorts the variables in "natural" order (`bio_2 < bio_3 < bio_12`; `sdm_core.py:189-200`, `1617-1620`); thus the results do not depend on the order of addition.
- **When and how to set it.** Because layers are matched by name in future scenarios (⑤), current and future files must have the same names (`bio_1.tif` ↔ `bio_1.tif`).
- **Effect on results.** Only the **first band** of each layer is read (`sdm_core.py:1608`); the other bands of a multi-band GeoTIFF or NetCDF are not used. Continuous layers are held in memory as float32 if the source type can be carried losslessly, otherwise as float64 (`sdm_core.py:1367-1379`); values are preserved bit for bit, and all computations are done in float64.
- **Caution.** The layer's NoData value must be defined in the file; if it is not, sea or empty cells (e.g. -3.4e38) are taken as real values. Two files that give the same name after cleaning (`bio 1.tif` and `bio_1.tif`) are not checked when no boundary is used; the one read later replaces the earlier one. When a boundary is used, two files with the same name give an error: "Two predictor rasters have the same name" (`sdm_mask.py:337-345`).

2.4.2 `Add Folder…`: continuous/categorical classification and its rationale

- **What it does.** `scan_raster_folder` sorts the `.tif .tiff .asc .img .nc .vrt` files in the folder by name and classifies each with `classify_raster` (`sdm_mask.py:54-141`). The rules, in order, are:
  1. A file whose name looks like a bioclimatic layer (`bio1`, `bio_12`, `wc2.1_30s_bio_5`) and which carries no colour table is taken as **continuous** without reading its values (so the gigabytes of text of a 30 arc-second global `.asc` are not read).
  2. Other files are read decimated: downsampled by nearest neighbour so that the long side is at most about 1000 cells.
  3. If there are no valid cells, the file is left continuous and flagged as suspicious.
  4. If there is a colour table (palette), **categorical**.
  5. If there are decimal values, **continuous**.
  6. If all values are integers and the number of distinct values is at most 30 (`MAX_CATEGORICAL_CLASSES`), **categorical**; if there are at least 12 distinct values and the name contains no word suggesting classes, or the name suggests a continuous variable (`elev`, `slope`, `ndvi`, `temp`, `prec` …), it is flagged as suspicious.
  7. If there are more than 30 distinct integers, **continuous**; suspicious if the name contains `landcover`, `soil`, `geolog`, `class`, `zone` ….
- **When and how to set it.** This is the fastest route for a user who keeps bioclimatic and other layers in one folder. The result window ("Rasters added") lists each file with the letter `N` (continuous) or `C` (categorical) and its rationale; rows marked ⚠ should be checked by hand, and a misclassified file should be moved with `→ Categorical` / `→ Continuous`.
- **Effect on results.** In the sample data all 19 files were taken as continuous from their names (log: "19 continuous, 0 categorical, 0 unreadable"). A layer left in the wrong list is misrepresented in the model: a temperature layer left in the categorical list does not enter the VIF analysis and is treated as classes.
- **Caution.** Ordinal integer scales such as a 1–5 productivity score cannot be distinguished from class codes by their values and are taken as categorical without being flagged (`sdm_mask.py:48-51`). Files that cannot be opened are reported with a "✕ … could not be opened" row and are not added to the list.

2.4.3 How categorical layers enter the model

- **What it does.** Categorical layers are read converted to integer (int32), NoData cells are marked with `-9999`, and the class list of each layer is extracted (`sdm_core.py:1653-1677`). At the sampled points each layer appears as a single column holding the class code itself (label encoding). The code is not used as a number: the first step of the GLM, GLMNET, SVM and ANN pipeline expands each class into a separate 0/1 column (`_CatPrep`, `sdm_core.py:3364-3384`); GAM uses a factor term and MaxEnt the categorical feature class; tree methods (RF, GBM, BRT, XGBoost, LightGBM, CatBoost, CART) split on the code; BIOCLIM, Domain, ENFA and Mahalanobis leave categorical layers out (`sdm_core.py:1781-1787`).
- **When and how to set it.** For genuine class layers such as land cover. For a species with few records, a layer with many classes produces classes with no or very few records in the training data; the program reports these with a "sparse classes" warning (`sdm_core.py:3232-3246`) and offers the option of excluding the layer in ② Variables.
- **Effect on results.** Cells where a categorical layer is NoData are removed from all sampling (`_valid_cell_mask`, `sdm_core.py:2309-2330`); a land cover layer with narrower coverage than the continuous layers shrinks the study area. The numbers of records and cells lost for this reason alone are reported to ② Variables (`sdm_core.py:3073-3078`).
- **Caution.** Decimal values are truncated when converted to integer (2.7 → 2). A file in the categorical list must also be on the same grid as the continuous layers (§2.4.5). At least one continuous layer is mandatory; running with only categorical layers stops with the message "Please add at least one environmental raster." (`tabs/tab_data.py:1368-1369`).

2.4.4 Converting large ASCII grids to GeoTIFF

- **What it does.** If the total size of the `.asc` files added with `Add Folder…` exceeds 200 MB (`_ASC_CONVERT_MB`), the program offers to convert them once to compressed, tiled (DEFLATE) GeoTIFF copies ("Convert ASCII grids to GeoTIFF?"; `tabs/tab_data.py:1292-1330`, `sdm_mask.py:144-178`). The copies are written with the same names to a `geotiff/` subfolder inside the folder, and the paths in the list are replaced with them; the `.asc` files are not touched. A copy that is newer than its source is not regenerated.
- **When and how to set it.** "Yes" is recommended for global `.asc` layers: because ASCII text is parsed from scratch at every read, clipping, sampling and projection each take minutes; GeoTIFF is read window by window.
- **Effect on results.** Values and the coordinate system remain the same; only the run time is shortened.
- **Caution.** The offer appears only when adding by folder; it is not asked for `.asc` files added individually.

2.4.5 Grid alignment and coordinate system

- **What it does.** The program reads all layers cell by cell into a single table; therefore, when no boundary is used, every layer must have the same **number of rows and columns, cell size, origin and coordinate system** as the first layer (the first name in natural order). A difference of 1e-6 units in cell size and origin is tolerated (`sdm_core.py:1683-1744`). Each non-matching layer is listed with its name and the difference: "Environmental rasters are not on the same grid. • bio_12: cell size …".
- **When and how to set it.** Layers from different sources should be aligned beforehand (QGIS: Raster > Align Rasters). Alternatively, if a study boundary is given, the program brings the layers to a common grid itself (§2.5.2).
- **Effect on results.** The coordinate system can be geographic (degrees) or projected (metres). Every distance-based decision (thinning, buffer, minimum distance, block size) is made in kilometres using great-circle (haversine) distance; on a projected grid the coordinates are first converted to longitude/latitude (`sdm_spatial.py:113-141`). If the length unit of a projected system cannot be read, the run stops ("The length unit of the raster coordinate system could not be determined"; `sdm_spatial.py:93-110`).
- **Caution. The coordinates of the occurrence CSV must be in the same coordinate system as the rasters.** The program does not reproject the CSV; it converts the coordinates to cells directly with the raster's transform matrix (`sdm_core.py:1763-1765`). If a raster in UTM is used with a CSV in degrees, all records fall outside the raster. When the rasters are geographic, a warning is given for coordinates whose latitude exceeds ±90 or whose longitude exceeds ±180 (`sdm_inputs.py:179-196`). If the cell size at the equator is about 4 km or larger, a resolution warning is written (`sdm_core.py:1751-1761`); in the sample run (2.5′, about 4.6 km) this warning appeared.

2.5 Study boundary: Study Boundary (mask, optional)

This group clips all rasters with a polygon or with a rectangle in degrees. Global layers can thus be used without being cut beforehand, and every cell outside the boundary becomes NoData.

Name in the UI	What it does	Default	Range / options	Code
Mode box	Selects the type of boundary	`None (whole raster extent)`	`None (whole raster extent)` / `Polygon layer (SHP, GPKG, KML, GeoJSON)` / `Corner coordinates (degrees)`	`tabs/tab_data.py:173-181`
File field + `Browse…` / `Clear`	Polygon layer (visible only in the second mode); selecting a file switches the mode to polygon automatically	empty	`.shp .gpkg .geojson .json .kml .gml`	`tabs/tab_data.py:161-172`, `1332-1339`
`West` / `South` / `East` / `North`	Corners of the rectangle, WGS84 degrees (third mode only)	25 / 35 / 45 / 43	longitude −180…180, latitude −90…90; 4 decimals	`tabs/tab_data.py:186-197`

2.5.1 Modes and the scope of clipping

- **What it does.** When a boundary is given, `Load Data` first of all clips all continuous and categorical layers (`tabs/tab_data.py:1511-1524`). Every subsequent step (record filtering, pseudo-absences, background, cross-validation folds, the current map, the future layers in ⑤ and the ⑥ range change) sees only the cells inside the boundary, because a cell that is NoData in one layer is excluded everywhere. Corner coordinates are taken as an EPSG:4326 rectangle and converted to the raster's system if needed (`sdm_mask.py:193-201`).
- **When and how to set it.** When the model and maps must be restricted to a country, a catchment or a region. The boundary differs from the **accessible area** described in Chapter 3: the boundary clips all layers and maps; the accessible area only narrows the area from which background and pseudo-absences are drawn, and the map is still produced for the whole boundary.
- **Effect on results.** Records falling outside the boundary are dropped with the reason "outside raster extent" or "falls on NoData cell". Because the background is drawn only from inside the boundary, narrowing the boundary affects the evaluation scores in a way similar to narrowing the accessible area (Chapter 3).
- **Caution.** The check "West must be smaller than East and South smaller than North" is performed (`tabs/tab_data.py:1488-1491`). In polygon mode, if there is no file, the run stops with the message "The study boundary layer was not found". If the layer has no polygons ("has no polygon features"), if the boundary does not overlap the rasters at all, or if a layer has no values inside the boundary, the run stops and names the layer (`sdm_mask.py:213-216`, `369-383`, `399-403`). If the boundary layer or the rasters do not declare a coordinate system, the two are assumed to be the same and this is written to the report with a warning (`sdm_mask.py:228-236`).

2.5.2 How clipping is done: single window, common grid, `masked_inputs`

- **What it does.** `mask_rasters` (`sdm_mask.py:304-418`) computes, on the first layer of the list, **a single window** corresponding to the bounding box of the boundary; all layers are written to the grid of this window. A layer with the same cell size and alignment is read through the window (values unchanged); a layer with a different resolution, a shifted origin or another coordinate system is resampled to this grid by **nearest neighbour** (`sdm_mask.py:273-301`). Cells outside the polygon are then set to NoData. The clipped copies are written as GeoTIFF with their original names under `<output folder>/masked_inputs/current/`; the progress bar shows "Rasters clipped to the study boundary: R x C cells.".
- **When and how to set it.** For using together layers that were cut to different extents or downloaded one cell wider (common in GCM sets), the boundary mode is a practical way of aligning them.
- **Effect on results.** Nearest neighbour produces no new values and preserves class codes; but when a fine-resolution continuous layer is brought down to a coarse grid, the value of a single cell is taken, not the mean. The future layers in ⑤ are clipped to the same grid, so the current and future maps overlap cell by cell.
- **Caution.** The grid is determined by the first layer of the list (first the continuous, then the categorical list, in order of addition). Putting the layer with the finest resolution first prevents loss of information. For a multi-species file the `masked_inputs` folder is also in the main output folder; the species share the same clipped copies.

2.6 Reproducibility: Reproducibility

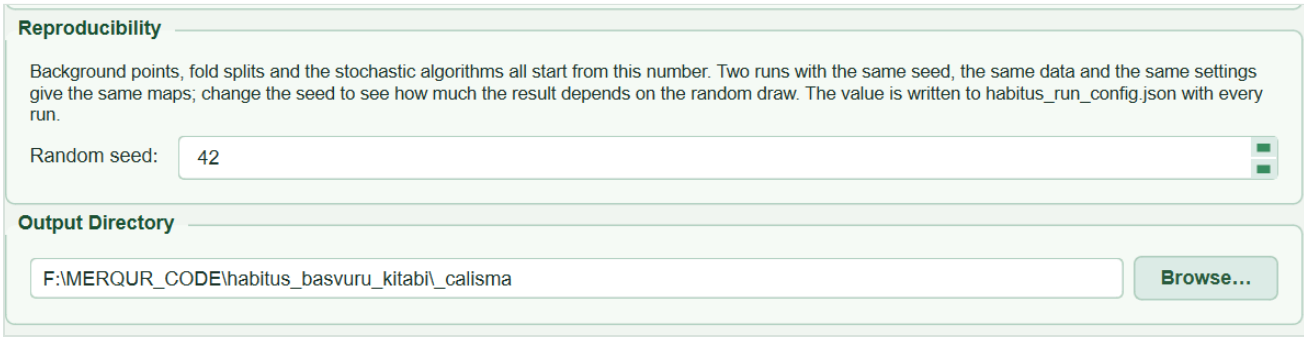


Figure 2.3. The Reproducibility group (Random seed 42) and the Output Directory field; the explanatory text states what the seed is applied to and that it is written to the habitus_run_config.json file.

Name in the UI	What it does	Default	Range / options	Code
`Random seed:`	Starting number for all random operations	42	0 – 2147483647	`tabs/tab_data.py:242-249`, `run_config.py:28`

2.6.1 Random seed

- **What it does.** When the data step starts, `set_run_seed` seeds NumPy's global generator, Python's `random` module and the `PYTHONHASHSEED` environment variable with this number (`tabs/tab_data.py:1510`, `run_config.py:31-50`). `DataFormatter` also builds its own generator (`numpy.random.default_rng(seed)`) from this number; pseudo-absences, background draws, the thinning order, the variogram subsample and k-means derive from it (`sdm_core.py:1503-1505`, `2301-2302`, `2536`). The `random_state` of the models is also given from the same number (Chapter 5).
- **When and how to set it.** In a study to be published, the value is reported; it is written to the `random_seed` field of the `habitus_run_config.json` file at every run. To see how much the result depends on the random draws, several different seeds are tried with the same settings.
- **Effect on results.** The same data, the same settings and the same seed are expected to give the same maps. When the seed changes, the pseudo-absence and background points, the folds and the stochastic algorithms change; the fluctuation in the scores and maps shows the sensitivity of the data to these draws.
- **Caution.** `PYTHONHASHSEED` is set only if it is not already defined (`setdefault`), and setting it after the Python interpreter has started does not affect hash ordering. Whether multi-threaded libraries (e.g. some gradient boosting implementations) give bit-for-bit identical results on different machines is not guaranteed in the code and has not been tested; reproducibility on the same machine with the same installation is the aim.

2.7 Output folder: Output Directory

Name in the UI	What it does	Default	Range / options	Code
Folder field + `Browse…`	Folder to which all outputs are written	when a CSV is selected, `Documents/HABITUS/<species>_<time>`	any folder	`tabs/tab_data.py:253-260`, `1125-1160`

It does not affect the results; if left empty, the run stops with the message "Please specify an output directory.". In a multi-species file each species is written to the `<output folder>/<species>` subfolder (`tabs/tab_data.py:1115-1123`); the session log is also moved to this folder at the start of the run, so in a batch run each species' log is in its own folder (`tabs/tab_data.py:1398-1408`).

2.8 The Load Data button: order of operations and record filtering

The `▶ Load Data & Generate Background/Pseudo-Absence Points` button runs the whole step in the background, without freezing the UI (`tabs/tab_data.py:1363-1574`). The order of operations matters for understanding which record was dropped and why:

1. **Pre-check:** does the CSV exist, is there at least one continuous raster, is the output folder filled in, do the target-group CSV and the bias raster (if given) exist, is the boundary valid.
2. If there was a previous data step, all results of the later tabs are reset.
3. In a multi-species file, only the records of the selected species are separated into `occurrences_used.csv`.
4. The seed is set (§2.6).
5. If a boundary is given, all layers are clipped (§2.5).
6. `load_data`: the CSV and the layers are read, the species name is determined, grid alignment is checked (`sdm_core.py:1549-1681`).
7. `generate_pa_datasets`: coordinates are converted to numbers, records are filtered (this chapter), thinning and the accessible area are applied, pseudo-absence sets and backgrounds are drawn (Chapter 3), and the collinearity pre-computation is performed (`sdm_core.py:2911-3249`).
8. The accessible area is written as `accessible_area.tif`; if requested, the MaxEnt background is saved to CSV.

2.8.1 Steps of record filtering

Presence records are filtered in the following order (`sdm_core.py:2931-2968`, `1824-2037`). Every dropped record enters the `dropped_points` list with its reason; the list appears in the warning window, the log, the report and `habitus_run_config.json` (`points_dropped_by_reason`).

Order	Reason (text in the UI)	Rule	Can it be turned off	Code
1	`missing coordinate`	Longitude or latitude cell is empty	no	`sdm_core.py:2944-2953`
2	`duplicate coordinate`	Identical to an earlier record when the coordinates are rounded to 8 decimals	no	`sdm_core.py:1844-1856`
3	`outside raster extent`	Cell row/column is outside the grid	no	`sdm_core.py:1864-1876`
4	`falls on NoData cell`	NaN in any continuous layer or NoData in any categorical layer	no	`sdm_core.py:1878-1905`
5	`same raster cell as another record`	A record earlier in file order exists in the same predictor cell	yes: `Keep one record per raster cell`	`sdm_core.py:1913-1936`
6	`spatially thinned`	A retained record exists within the thinning distance	yes (Chapter 3)	`sdm_core.py:2960-2968`

2.8.2 Duplicate coordinates, outside the raster and NoData

- **What it does.** An exactly identical coordinate (up to a rounding difference of about 1 mm) is kept only where it first occurs. Of the remaining records, those falling outside the grid and those falling on a cell with no value in some layer are dropped. Records dropped because of a categorical layer are counted separately (`cat_nodata_loss_presence`) and reported to ② Variables.
- **When and how to set it.** These three filters always run; there is no setting.
- **Effect on results.** At coarse resolution some coastal records may fall on a sea cell and be caught by NoData. The share of dropped records determines the number of presences used and hence the prevalence in Chapter 3.
- **Caution.** If more than 20% (`LOSS_WARN_FRACTION`) of the non-duplicate records fall outside the raster or on NoData, the program swaps longitude and latitude and tries again; if at least half of the records fall inside in the swapped state, it gives the warning "The lon/lat columns are probably swapped", otherwise the warning asking you to check the coordinate system and the columns (`sdm_inputs.py:199-221`, `sdm_core.py:1969-1990`). When all records are lost and one layer is NoData at every record inside the extent, the culprit is that layer, not the coordinates; the message names the layer (`sdm_core.py:2014-2029`).

2.8.3 One record per cell

- **What it does.** Records falling in the same predictor cell carry the same predictor values; each extra record adds weight to that cell without adding information. With the option on, the first record in file order is kept in each cell and the others are dropped with the reason "same raster cell as another record" (`sdm_core.py:1913-1936`). With the option off, these records are still counted and a warning is written (`sdm_core.py:1962-1967`). The checkbox itself is in the Study Area & Thinning sub-tab (Chapter 3, `tabs/tab_data.py:722-731`); its default is on.
- **When and how to set it.** It should almost always stay on. Turning it off comes into question only for species with very few records concentrated in a few cells; even then the records repeat the same cell values.
- **Effect on results.** In the sample run, 405 of the 626 records were dropped because they fell in a cell that already had a record, leaving 221 records (session log). Had it been off, a few intensively observed areas (e.g. hundreds of records in the same valley) would have dominated both the model curves and the cross-validation folds.
- **Caution.** The rule is applied before thinning. Which record is kept depends on file order; even if the records in the same cell have different coordinates, their predictor values are the same, so there is no difference for the model, but thinning and spatial blocks use the exact location of the record. The external test set (⑦) is exempt from this rule.

2.8.4 At least five presences

- **What it does.** If at least 5 records (`MIN_PRESENCES`, `sdm_core.py:1269`) do not remain after the filters, the run stops. The message varies with the cause (`sdm_core.py:1997-2035`): if the loss is due only to the one-per-cell rule, "The N presence record(s) fall in only K predictor cell(s)…"; if there is no loss at all, "Only K usable presence record(s). At least 5 are required…"; if the losses come from outside-raster/NoData, the coordinate system warning.
- **When and how to set it.** It cannot be changed. The threshold rests on the smallest samples in published studies comparing SDM methods (Hernandez et al. 2006; Pearson et al. 2007).
- **Effect on results.** With 5–24 records the run continues but warnings appear (§2.8.5).
- **Caution.** This limit is checked **before** thinning. If fewer than 10 records remain after thinning, only a warning is given ("Thinning at … km leaves only N records"; `sdm_core.py:2547-2552`); the run does not stop.

2.8.5 Warnings on record count and the record/predictor ratio

- **What it does.** `_check_sample_size` (`sdm_core.py:2843-2885`) compares the number of presences remaining after thinning (n) with the number of loaded predictors (k: continuous + categorical) and can produce three warnings:
  - $n < 10$: "Only N usable presence record(s). No species distribution method produces a trustworthy model at this sample size…"
  - $n < 25$ (`JACKKNIFE_RECOMMENDED_MAX`, `sdm_spatial.py:594`): selecting jackknife validation (leaving out one presence at a time) in ③ Models is recommended (Pearson et al. 2007).
  - $n \geq 10$ and $n/k < 10$: "N presence records for the K predictors loaded (R per predictor)…". The text states that the number of predictors to be kept in ② Variables is the number that really matters and that n records can support about $\lfloor n/10 \rfloor$ predictors.
- **When and how to set it.** The rule of 10 records per predictor is a common guideline for logistic-type models; flexible learners (RF, boosting, MaxEnt with hinge features) require more (Wisz et al. 2008).
- **Effect on results.** In the sample run, with 143 records and 19 layers the ratio is 7.5; the message says "143 records allow about 14". When 8 variables are kept in ② Variables (ratio 17.9), the same check is repeated and the warning disappears (`note_selected_predictors`, `sdm_core.py:2889-2909`; `tabs/tab_vif.py:1213`).
- **Caution.** The warnings do not change the result; they are written to the report as "advisory". The k in this section counts all layers loaded in ① Data; for the number after selection, the message in ② is authoritative.

2.8.6 Messages after the data step

When the step finishes, the following windows appear in order (`tabs/tab_data.py:1576-1761`):

1. **"Invalid Occurrence Points Detected"**: if at least one record was dropped; one heading per reason and a list of coordinates (under the cell and thinning headings the first 15 records, the rest as "… and N more"), ending with "✓ N valid point(s) will be used for modelling." and a CRS reminder. The same text is written to the log.
2. **"Points worth knowing before you interpret this run"**: warnings that affect the interpretation of the result (resolution, record/predictor ratio, accessible area, thinning, absence records…). If the only warning is the sampling bias reminder, this window does not open and the warning is written only to the log with the prefix "ADVISORY:".
3. **"Data ready!"** summary: number of presences used/loaded, breakdown of dropped records by reason, the pseudo-absence or true absence row, number of MaxEnt background points, the accessible area and the share it covers, the thinning distance and how it was chosen, the autocorrelation range, prevalence, the note that sampling bias has not been corrected, the seed, and the names of the continuous and categorical variables.

The summary of the sample run, compiled from the session log, is as follows: 626 records loaded; 405 dropped as same cell and 78 by thinning; 143 presences used; 2 sets × 1000 pseudo-absences; 10,000 MaxEnt background points; accessible area buffer, 200 km, 40% of the raster extent; thinning 4.7 km (automatic); autocorrelation range 151.5 km; prevalence 0.125.

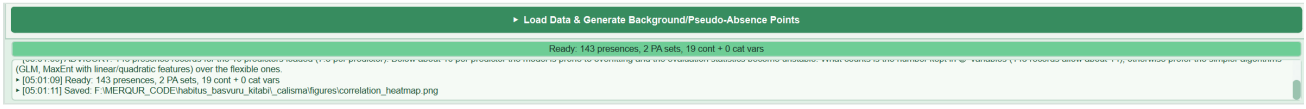


Figure 2.4. The bottom bar after the data step has finished: "Ready: 143 presences, 2 PA sets, 19 cont + 0 cat vars" in the progress bar, and in the log the end of the record/predictor warning and the line for the correlation heat map saved on moving to ② Variables.

2.9 Outputs and what passes to the next step

- **In memory (for ② onwards):** predictor arrays, variable names (`var_names`, `cat_var_names`), pseudo-absence sets (`pa_datasets`), MaxEnt and ENFA backgrounds, filtered presences, the autocorrelation range (used as the spatial block size in ③), the collinearity pre-ranking (`collinearity_df`, the starting table of ② Variables).
- **On disk:** `masked_inputs/current/*.tif` (if a boundary is given), `occurrences_used.csv` (in a multi-species file), `accessible_area.tif` (if an accessible area is defined; 1 = inside, 0 = outside), `background_maxent.csv` (if requested), the session log `habitus_<time>.log`.
- **Run record:** the `data` section of `habitus_run_config.json` contains the CSV used, the species name, the number of presences used, the breakdown of dropped records by reason, the raster lists, the grid information (`raster_grid`: reference layer, rows-columns, cell size, km at the equator, CRS) and the prevalence; the `study_boundary` section contains the boundary, the number of polygons and the clipped grid. This record is written when ③ Models is run.

2.10 Expert questions

Q1. Are the coordinates transformed to the raster's coordinate system, or are they expected to be in the same system? Answer: They are expected to be in the same system. Records are converted directly to row-column with the raster's transform matrix (`sdm_core.py:1763-1765`, `1866`); there is no coordinate system field or reprojection for the CSV. Reprojection is done only for the boundary and calibration polygons (`sdm_mask.py:219-227`, `sdm_spatial.py:1194-1225`). With projected rasters the CSV must also be in metres; with geographic rasters values outside ±90/±180 produce a warning.

Q2. Which operation in R does the one-record-per-cell rule correspond to, and why is it applied before thinning? Answer: It is de-duplication by cell identity on the predictor grid; it resembles selecting one point per grid cell with `gridSample` in dismo, or filtering with `duplicated` on cell numbers. HABITUS keeps the first record in file order in each cell (`sdm_core.py:1919-1936`). The reason for applying it first is that records in the same cell are copies of each other in terms of predictors, and thinning would waste time eliminating these copies in random order; moreover, distance-based thinning is meaningless at the sub-cell scale. If a thinning distance smaller than the cell is entered, the program reports it (`sdm_core.py:2526-2534`).

Q3. How precisely are duplicate coordinates compared? Answer: Longitude and latitude are rounded to 8 decimals and compared (`sdm_core.py:1848`); in degrees this is about 1 mm. That is, only a truly identical point counts as a duplicate; records entered a few metres apart pass this filter and are left to the one-per-cell rule and thinning.

Q4. How is it detected that longitude and latitude are swapped? Answer: If more than 20% of the non-duplicate records fall outside the raster or on NoData, the records are placed on the grid once more with latitude and longitude swapped; if the share falling inside in this state is at least 0.5 and greater than the actual share inside, the message says that the columns are most likely reversed (`sdm_core.py:1972-1990`, `sdm_inputs.py:207-216`). The program does not swap the columns automatically; the decision is left to the user.

Q5. What is the minimum number of records, where does this number come from, and what is recommended with few records? Answer: At least 5 records after filtering are mandatory (`sdm_core.py:1269`, `1993-2013`); the rationale is the studies in which SDM methods were tested with small samples (Hernandez et al. 2006; Pearson et al. 2007). Warnings are written that below 10 no reliable model can be produced and below 25 jackknife validation should be selected (`sdm_core.py:2864-2875`). For species with very few records, the ESM (ensemble of small models) option in ③ Models is described separately (Chapter 5).

Q6. With which number of predictors is the "10 records / predictor" rule applied? Answer: With the number of all layers loaded in ① Data (`sdm_core.py:2858`); this number is usually larger than the number in the final model. When the selection is confirmed in ② Variables, the same check is repeated with the selected number and the new message replaces the previous one (`sdm_core.py:2889-2909`). The rule is advisory; it does not stop the run. Source: Wisz et al. (2008) and the events-per-variable tradition in logistic regression.

Q7. Can layers of different resolution or extent be used together? Answer: Without a boundary, no: the number of rows and columns, cell size, origin and CRS must be the same, otherwise the run names the non-matching layer and stops (`sdm_core.py:1698-1744`). With a boundary, yes: all layers are brought by nearest neighbour to the common grid defined by the cells of the first layer of the list (`sdm_mask.py:283-301`). The program does not aggregate by averaging; to bring a fine layer down to a coarse grid, aggregating it beforehand with a GIS tool is more appropriate.

Q8. Why is a categorical layer not used as a number in linear models, and which algorithms leave it out? Answer: The layer is carried in a single column with its class code, but the first step of the GLM, GLMNET,

SVM and ANN pipeline (`_CatPrep`) produces a separate 0/1 column for each class seen in training; the code is never used as a number that is squared, scaled or multiplied (`sdm_core.py:3364-3384`). GAM uses a factor term and MaxEnt a categorical feature; tree methods split on the code; BIOCLIM, Domain, ENFA and Mahalanobis leave the layer out (`sdm_core.py:1781-1787`). If a class not seen in training is encountered in the projection area, all 0/1 columns of that variable become zero.

Q9. Do two runs with the same seed give bit-for-bit identical results? Answer: That is the aim: the seed is applied to all generators at the start of the data step (`tabs/tab_data.py:1510`, `run_config.py:31-50`), and the background draw, thinning, variogram subsample, k-means and folds come from their own seeded generators. The order in which layers are added does not change the result either, because the variables are sorted in natural order (`sdm_core.py:1617-1620`). Bit-for-bit equality across different operating systems or library versions is not guaranteed in the code and has not been tested; for this reason the run record also writes the library versions (`habitus_run_config.json`, `environment`).

Q10. What is the difference between the study boundary and the accessible area; can one be used in place of the other? Answer: The boundary clips all layers: cells outside become NoData and the map also ends at the boundary (`sdm_mask.py:304-418`). The accessible area only restricts the cells from which background and pseudo-absences are drawn; the map is still produced for the whole boundary or raster extent (`sdm_core.py:2181-2203`). To define the M area in the sense of Barve et al. (2011), the accessible area should be used; using the boundary as M also reduces the model's projection area to M.

Q11. How is the memory requirement determined? Answer: Each continuous layer is read fully into memory; sources that fit losslessly into float32 (float32, 16-bit integers) are held as float32, the others as float64 (`sdm_core.py:1367-1379`). According to the code comment, 19 global layers at 2.5′ resolution occupy 2.6 GB. The way to reduce memory is to give a study boundary; clipped layers cover only the bounding box of the boundary.

2.11 Common mistakes and pitfalls

- Rasters in UTM, CSV in degrees: all records become "outside raster extent"; the message says to check the coordinate system. Convert the CSV to the raster's system.
- No column names and the order is "species, latitude, longitude": reading by position reverses them; the "probably swapped" warning appears. Name the columns `longitude` and `latitude`.
- Degrees-minutes-seconds or a quoted decimal comma in the coordinates: the run stops with the message "holds non-numeric coordinates".
- Layers cut to different extents: "Environmental rasters are not on the same grid." Align them or give a study boundary.
- Layer with no defined NoData value: sea cells are taken as real values; coastal records are not filtered out and the background falls into the sea. Define the layer's NoData value.
- Multi-band file: only the first band is read.
- A continuous layer left in the categorical list (e.g. temperature stored as integers): check the ⚠ mark and move it with `→ Continuous`.
- The first layer of the boundary at coarse resolution: all layers are brought down to that grid.
- No species ticked in a multi-species file: the run stops.

2.12 References

Barve, N., Barve, V., Jiménez-Valverde, A., Lira-Noriega, A., Maher, S.P., Peterson, A.T., Soberón, J., Villalobos, F., 2011. The crucial role of the accessible area in ecological niche modeling and species distribution modeling. Ecological Modelling 222, 1810–1819. https://doi.org/10.1016/j.ecolmodel.2011.02.011

Hernandez, P.A., Graham, C.H., Master, L.L., Albert, D.L., 2006. The effect of sample size and species characteristics on performance of different species distribution modeling methods. Ecography 29, 773–785. https://doi.org/10.1111/j.0906-7590.2006.04700.x

Pearson, R.G., Raxworthy, C.J., Nakamura, M., Peterson, A.T., 2007. Predicting species distributions from small numbers of occurrence records: a test case using cryptic geckos in Madagascar. Journal of Biogeography 34, 102–117. https://doi.org/10.1111/j.1365-2699.2006.01594.x

Wisz, M.S., Hijmans, R.J., Li, J., Peterson, A.T., Graham, C.H., Guisan, A., NCEAS Predicting Species Distributions Working Group, 2008. Effects of sample size on the performance of species distribution models. Diversity and Distributions 14, 763–773. https://doi.org/10.1111/j.1472-4642.2008.00482.x

3. Data II: background, pseudo-absences, accessible area and thinning (① Data)

Tab: ① Data › Background & Pseudo-Absence Settings · Code: `tabs/tab_data.py` (UI), `sdm_core.py` `DataFormatter.generate_pa_datasets`, `_sample_background` (computation), `sdm_sampling.py` (strategies, bias surface), `sdm_spatial.py` (accessible area, thinning, variogram)

3.1 What this chapter covers

Most species records report presence only: we know where the species was seen, not where it is absent. For a model to discriminate presence it needs "comparison" points. HABITUS generates these points for three different purposes, as three separate sets:

- **Pseudo-absences:** points passed with the label "absent" to two-class (presence/absence) algorithms such as GLM, GAM, RF, GBM, BRT, SVM, ANN, XGBoost, LightGBM, CatBoost, GLMNET and CART. Several independent sets (repetitions) can be drawn.
- **MaxEnt background:** the sample MaxEnt uses to describe the "available environment". It is not treated as absence; it is drawn once and used in every MaxEnt fit.
- **ENFA background:** a separate sample ENFA uses to compute the global environmental mean and covariance.

Where these three sets are drawn from is set by the accessible area, **with what weight** by the sampling bias surface, and **relative to which records** by thinning. All of them are on the six sub-tabs of the `Background & Pseudo-Absence Settings` group at the upper right of ① Data, and they are applied together with the `Load Data` button (Section 2.8).

Processing order (`sdm_core.py:2955-3207`): record cleaning → thinning → accessible area mask (from the thinned records) → user's absence records → bias surface → strategy constraints (envelope, k-means) → pseudo-absence sets → MaxEnt background → ENFA background.

3.2 Screen layout

Sub-tab	Tooltip text	Which set	In this book
`ML Models`	ML Models (Pseudo-Absences)	pseudo-absences	§3.3
`Own Absences`	My Own Absence Data	true absences instead of / in addition to pseudo-absences	§3.4
`MaxEnt`	MaxEnt (Background Points)	MaxEnt background	§3.5
`Sampling Bias`	Sampling Bias (Target Group)	weights of all three sets	§3.6
`ENFA / Mahalanobis`	ENFA / Mahalanobis (Presence-Only)	ENFA background	§3.7
`Study Area & Thinning`	Study Area & Thinning	area of all three sets; presence records	§3.8, §3.9

The sub-tab tooltips are defined in `tabs/tab_data.py:214-219`. The screenshots in this chapter show the settings of the example run (*Abies nordmanniana*).

3.3 Pseudo-absences: ML Models

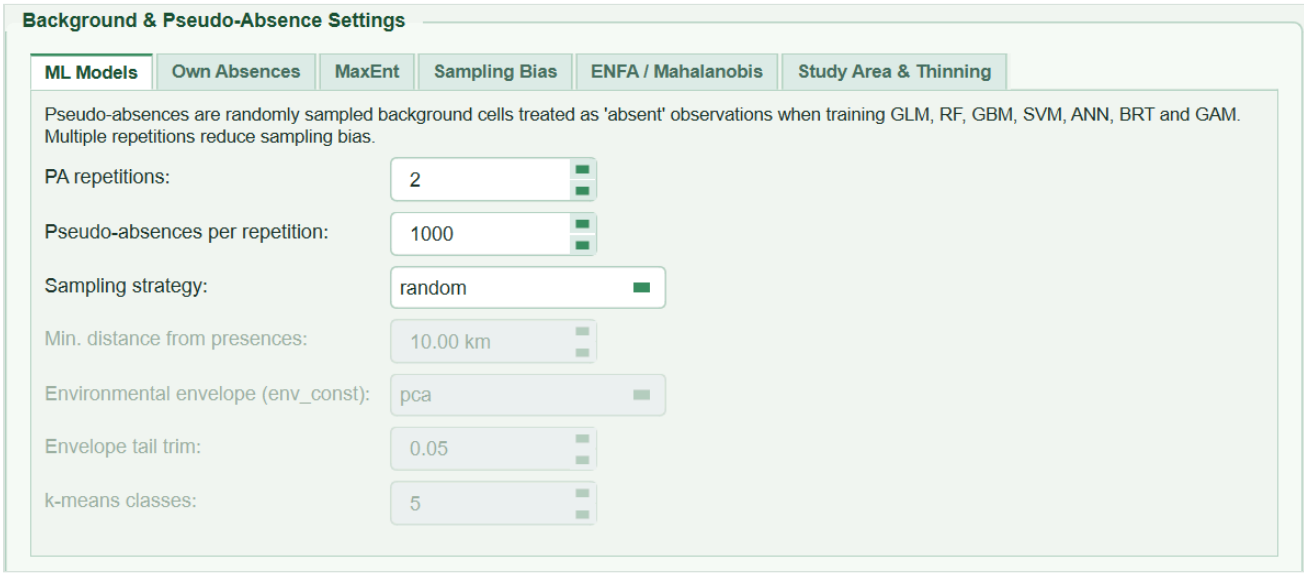


Figure 3.1. ML Models sub-tab: PA repetitions 2, Pseudo-absences per repetition 1000, Sampling strategy random. The Min. distance from presences, Environmental envelope (env_const), Envelope tail trim and k-means classes fields are greyed out because the random strategy does not use them.

Name in the UI	What it does	Default	Range / options	Code
`PA repetitions:`	Number of independent pseudo-absence sets	2	1 – 10	`tabs/tab_data.py: 293-297`
`Pseudo-absences per repetition:`	Number of pseudo-absences in each set	1000	100 – 100,000 (step 500)	`tabs/tab_data.py: 299-304`
`Sampling strategy:`	Which cells the pseudo-absences are drawn from	`random`	`random`, `disk`, `sre`, `kmeans`, `env_const`, `geo_const`, `geo_env_const`, `geo_env_km_const`	`tabs/tab_data.py: 306-326`, `sdm_sampling.py:47-53`
`Min. distance from presences:`	Minimum distance to presences (km)	10 km	0 – 500 km	`tabs/tab_data.py: 328-333`
`Environmental envelope (env_const):`	How the envelope is built	`pca`	`pca`, `bioclim`	`tabs/tab_data.py: 335-342`
`Envelope tail trim:`	Proportion trimmed from each end of the envelope	0.05	0.00 – 0.45	`tabs/tab_data.py: 343-348`
`k-means classes:`	Number of clusters in environmental space	5	2 – 30	`tabs/tab_data.py: 349-353`

Fields are enabled only for the strategy that uses them (`tabs/tab_data.py:557-573`): minimum distance for `disk` and `geo_*`; envelope and tail trim for `env_const`, `geo_env_const`, `geo_env_km_const`; k-means for `kmeans`, `geo_env_km_const`. If `replace` is selected in `Own Absences`, all fields are greyed out (§3.4).

3.3.1 PA repetitions

- **What it does.** With the same presences and a separately drawn set of `n` pseudo-absences each, `PA repetitions` training sets are built (`sdm_core.py:3035-3082`). The presence side of the sets is identical; only the absence side changes. ③ Models fits a separate model for each set (`PA1`, `PA2` in the log), and the ensemble combines the models from all sets (Chapters 5–6).
- **When and how to set it.** The UI tooltip suggests 2–5. When working with few pseudo-absences (e.g. as many as there are presences), the number of repetitions should be increased; Barbet-Massin et al. (2012) recommend averaging 10 repetitions with as many pseudo-absences as presences for classification trees.
- **Effect on results.** Repetition turns a result that depends on the luck of a single draw into an average; the differences between sets show the uncertainty arising from the choice of pseudo-absences. Run time grows roughly in proportion to the number of repetitions.
- **Caution.** In `Own Absences` `replace` mode the number of repetitions is reduced to 1, because there is no draw left to repeat (`sdm_core.py:3031-3033`); the report also states 1.

3.3.2 Pseudo-absences per repetition (number) and prevalence

- **What it does.** The number of pseudo-absences drawn in each set. Cells are drawn without replacement (`rng.choice(..., replace=False)`, `sdm_core.py:2301-2302`); if there are fewer candidate cells than requested, all are taken and a warning is written.
- **When and how to set it.** The tooltip suggests the range 1000–10,000 and no fewer than the number of presences. Barbet-Massin et al. (2012) recommended a large number (e.g. 10,000) with equal weighting of presences and absences for regression methods (GLM, GAM), and many repetitions with as many pseudo-absences as presences for tree methods. In HABITUS, class weighting is set separately in ③ Models (Chapter 5).
- **Effect on results. Prevalence** (the proportion of presences in the training set) is determined by this number: $p = n_{\text{presence}} / (n_{\text{presence}} + n_{\text{absence}})$ (`sdm_core.py:3087-3089`). In the example run it is $143/(143 + 1000) = 0.125$. Prevalence directly changes the level of the raw output of unweighted models, and hence the threshold values: increasing the number of pseudo-absences lowers the raw probabilities. AUC is threshold-independent but sensitive to where the pseudo-absences come from (§3.3.3, §3.8).
- **Caution.** If the number of pseudo-absences exceeds the number of valid cells in the accessible area, all cells in the area are used and the warning "The accessible area holds N usable cells, fewer than the M points requested" is written (`sdm_core.py:2190-2203`). A point that falls on NoData in a predictor after sampling is dropped and counted ("PA1: N sampled point(s) fell on NoData…", `sdm_core.py:2816-2841`); because NoData cells of categorical layers are excluded from sampling in advance, this is rare.

3.3.3 Sampling strategy (eight strategies)

All strategies start from the same candidate set: cells that have a value in every continuous and categorical layer (`_valid_cell_mask`, `sdm_core.py:2309-2330`), restricted to those inside the accessible area if one is defined (`sdm_core.py:2181-2203`). The strategy-specific constraint is then applied (`sdm_core.py:2205-2297`), and the draw is made with bias and class weights, if any.

Strategy	What it does	Fields used	R equivalent	Code
`random`	Equal-probability draw from candidate cells	—	biomod2 `random`, flexsdm `random`	`sdm_core.py:2291-2305`
`disk`	Excludes cells within `Min. distance` of each presence, draws randomly from the rest	minimum distance	biomod2 `disk` (minimum distance only)	`sdm_core.py:2243-2258`
`sre`	Draws from cells **outside** the bioclimatic envelope of the presences; the envelope is the 5%–95% range of each variable at the presences	(fixed 0.05)	biomod2 `sre`	`sdm_core.py:2079-2119`, `3005-3022`
`kmeans`	k-means in scaled environmental space; draws from classes containing no presences, with equal total weight per class	k	sdm `eRandom`, flexsdm `kmeans`	`sdm_sampling.py:245-288`
`env_const`	Draws from cells outside the environmental envelope of the presences	envelope, tail trim	flexsdm `env_const`	`sdm_sampling.py:219-242`
`geo_const`	Same rule as `disk`	minimum distance	flexsdm `geo_const`	`sdm_sampling.py:26-27`
`geo_env_const`	Minimum distance **and** outside the envelope	minimum distance, envelope, tail trim	flexsdm `geo_env_const`	`sdm_core.py:2997-3003`
`geo_env_km_const`	Minimum distance, outside the envelope and presence-free k-means classes	all	flexsdm `geo_env_km_const`	`sdm_core.py:2399-2429`

- **What it does.** The strategy determines "what" the model compares presences against. `random` represents the environmental variety of the accessible area as it is. The other strategies move pseudo-absences away from presences geographically (`disk`, `geo_const`) or environmentally (`sre`, `env_const`, `kmeans`); the aim is to reduce the labelling as "absent" of cells that are actually suitable but have no records (false absences).
- **When and how to set it.** Barbet-Massin et al. (2012) recommended random selection for regression methods, and geographically and environmentally distant selection for classification trees and discriminant methods. The default `random`, used together with an accessible area, is the option with the fewest assumptions. For an intensively sampled species with a high risk of false absences around the records, `disk` (e.g. 10–50 km) is a reasonable middle way.
- **Effect on results.** Every constraint that moves pseudo-absences away from presences makes discrimination easier: AUC and TSS **rise**, but part of this rise is a product of the sampling design rather than of the model; response curves are cut off outside the presence range, and the suitable area on the map expands (VanDerWal et al. 2009; Barbet-Massin et al. 2012). The outside-envelope strategies (`sre`, `env_const`) do this most strongly: the model tends to learn the inside of the envelope as "suitable" and the outside as "unsuitable", and simply reproduces the envelope.

- **Caution.** The envelope and k-means strategies are computed with **all** continuous layers loaded in ① Data; because the selection in ② Variables is made later, unselected layers also affect where the pseudo-absences are placed. In `sre`, a cell must be within range in **all** variables to count as inside the envelope; with 19 layers almost every cell falls outside in at least one variable, and the constraint weakens. If no cells remain outside the envelope, the strategy falls back to a random draw and a warning is written ("no cells outside the envelope found. Falling back to random sampling."; `sdm_core.py:2206-2212`). In strategies other than `random`, `disk` and `sre`, if there are fewer candidate cells than requested, fewer points are drawn ("leaves only N candidate cells"; `sdm_core.py:2294-2297`).

3.3.4 Min. distance from presences

- **What it does.** In `disk`, `geo_const`, `geo_env_const` and `geo_env_km_const`, the great-circle (haversine) distance from the centre of each candidate cell to each presence is computed; cells whose distance is not greater than this value are excluded (`sdm_core.py:2243-2258`). On a projected grid, the coordinates are first converted to longitude/latitude.
- **When and how to set it.** A distance on the order of the location error of the records and the local spread of the species (a few km to a few tens of km) is a typical choice. Barbet-Massin et al. (2012) tested a distance of about 2° for classification methods; in a small study area this removes most candidates.
- **Effect on results.** As the distance grows, pseudo-absences move away from the presences and their environmental conditions; AUC/TSS rise, and the model draws a "wider" suitable area.
- **Caution.** With a distance of 0, the `geo_*` strategies exclude nothing and report this as a warning (`sdm_core.py:2999-3003`). A very large distance stops with the error "No background cells remain after min_dist=… km. Reduce the minimum distance." The computation scans all candidate cells for each presence; on a global grid and with thousands of records it can take a long time.

3.3.5 Environmental envelope and Envelope tail trim

- **What it does.** In the `env_const` family, the environmental envelope of the presences is built in one of two ways (`sdm_core.py:2407-2420`):
  - `pca`: principal components (SVD) are computed on the standardised continuous variables in all valid cells; the $[q,\ 1-q]$ quantile range of the presences on the first two components forms the envelope. A cell is outside the envelope if it lies outside this range on at least one of the two components (`sdm_sampling.py:219-242`).
  - `bioclim`: each variable separately; a cell is inside the envelope if it lies within the $[q,\ 1-q]$ range in all variables (the same computation as `sre`, but q is adjustable; `sdm_core.py:2410-2412`).
  - `Envelope tail trim` is q: 0 gives the full minimum–maximum range, 0.05 the 5%–95% range.
- **When and how to set it.** `pca` is the closest to flexsdm's `env_const` method. If location errors or outliers are suspected in the records, q is increased (the envelope narrows).
- **Effect on results.** As q increases, the envelope narrows, the outside-envelope candidate area grows and the pseudo-absences approach presence conditions; as q decreases, the envelope widens, the pseudo-absences move environmentally away from the presences and discrimination becomes easier.
- **Caution.** The principal components are built from all valid cells, not restricted to the accessible area; the envelope itself is then intersected with the accessible area. The proportion of outside-envelope cells is written to the run record (`frac_cells_outside_envelope`). The `sre` strategy does not use these two fields; its trim is always 0.05.

3.3.6 k-means classes

- **What it does.** Valid cells are split into `k` classes in standardised environmental space (scikit-learn `KMeans`, `n_init=3`; if there are more than 50,000 cells, the model is fitted on a random subsample of 50,000 cells and all cells are classified afterwards). Presences are assigned to the nearest class centre. Classes without presences are the source of pseudo-absences, and each receives the same total weight (stratified draw). If every class contains presences, the class weight becomes $1/(1 + n_{\text{presence}})$ and a warning is written (`sdm_sampling.py:245-288`).
- **When and how to set it.** The default is 5. In an environmentally diverse study area, increasing the number of classes makes it easier to find presence-free classes.
- **Effect on results.** Equal class weights lead to over-representation of small but environmentally distinct classes among the pseudo-absences; environmental variety is balanced, but the geographic distribution changes.
- **Caution.** The classes are built from all valid cells; if all presence-free classes fall outside the accessible area, the run stops with the message "No candidate cell has a positive sampling weight" (`sdm_core.py:2281-2285`). The k-means seed is derived from the run seed.

3.4 Your own absence records: Own Absences

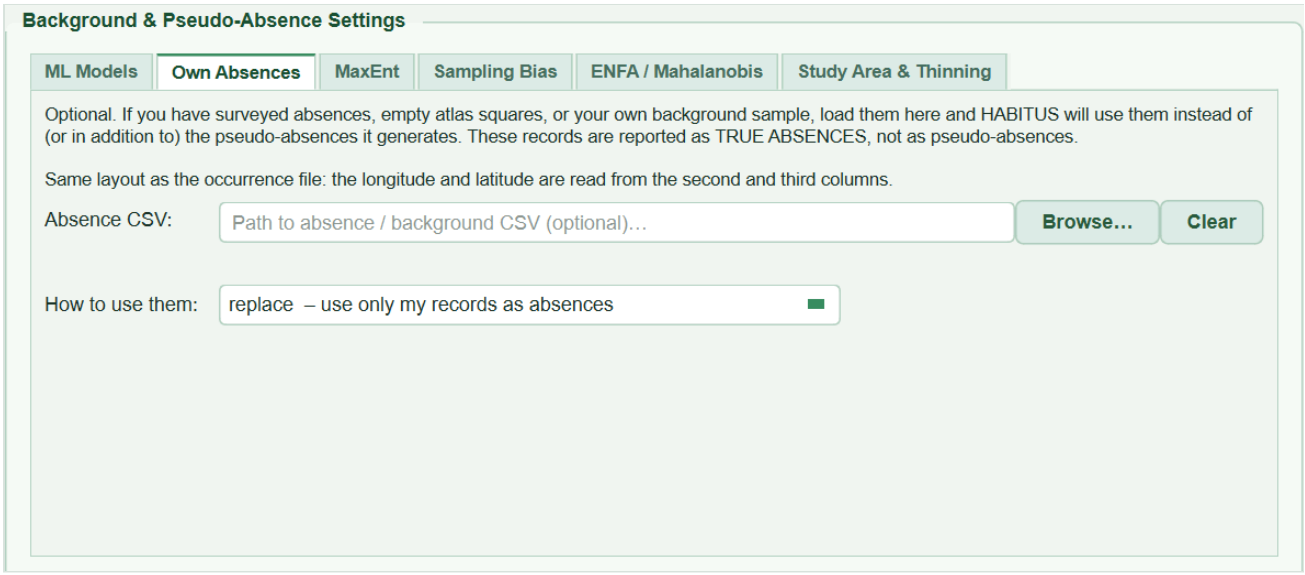


Figure 3.2. Own Absences sub-tab: description text, empty Absence CSV field (Browse… and Clear) and the default mode "replace – use only my records as absences".

Name in the UI	What it does	Default	Range / options	Code
`Absence CSV:` + `Browse…` / `Clear`	Observed absence (or own background) records; once selected, the record count and the columns to be read are previewed	empty	CSV	`tabs/tab_data.py:478-491`, `1162-1181`
`How to use them:`	Whether the records replace or supplement the pseudo-absences	`replace`	`replace – use only my records as absences` / `augment – my records plus generated pseudo-absences`	`tabs/tab_data.py:493-507`

3.4.1 Absence CSV and columns

- **What it does.** The file is read with the same reader as the records file (including delimiter and decimal detection). Coordinate columns are looked up in this order (`sdm_core.py:2729-2754`): the longitude/latitude column names of the records file, if they also exist in this file; otherwise `longitude/lat…` columns found by name; otherwise the 2nd and 3rd columns; if there are only two columns, the 1st and 2nd. Rows with empty coordinates are dropped and counted; those falling outside the raster or on NoData are dropped with the reason "absence record outside extent or on NoData", and if the loss exceeds 20%, an explanatory warning is given as for presences (`sdm_core.py:2770-2797`).
- **When and how to set it.** For points where the species was not found in a field survey, empty atlas squares, or a target-group background compiled with the same method. These records are reported in the report and summary as **"TRUE absences"**, separately from pseudo-absences.
- **Effect on results.** A model trained with true absences obtains the information "where the species does not occur" from observation rather than by guesswork; the meaning of the evaluation statistics also changes (discrimination against true absences rather than pseudo-absences).
- **Caution.** Absence records are not reduced to one per cell, not thinned, and not checked for falling in the same cell as a presence. If a file is given but cannot be used, the program does not silently fall back to pseudo-absences; it stops ("All N user-supplied absence records fall outside the raster extent…", `sdm_core.py:2791-2797`).

3.4.2 replace and augment modes

- **What it does.**
  - `replace`: The absence side consists only of your records. No pseudo-absences are generated; all fields in `ML Models` are greyed out, the number of repetitions drops to 1, and the minimum distance and accessible area are not applied to your records (`tabs/tab_data.py:535-546`, `sdm_core.py:3031-3040`).
  - `augment`: Your records appear unchanged in each set, and `Pseudo-absences per repetition` pseudo-absences are drawn on top (`sdm_core.py:3060-3062`). The differences between sets come only from the generated part.
- **When and how to set it.** `replace` if the records come from a real and comprehensive absence survey; `augment` if they are few, or a partial background rather than a complete survey.
- **Effect on results.** In `replace`, prevalence is your presence/absence ratio. In `augment`, the absence side grows and prevalence falls.
- **Caution.** In `replace` mode, if fewer than 20 absence records are supplied, the warning "Only N true absence records were supplied and pseudo-absences were switched off…" is given and `augment` is recommended (`sdm_core.py:2807-2813`). Absence records enter only the training sets of the two-class algorithms: MaxEnt continues to use its own background, and ENFA its own background (§3.5, §3.7).

3.5 MaxEnt background: MaxEnt

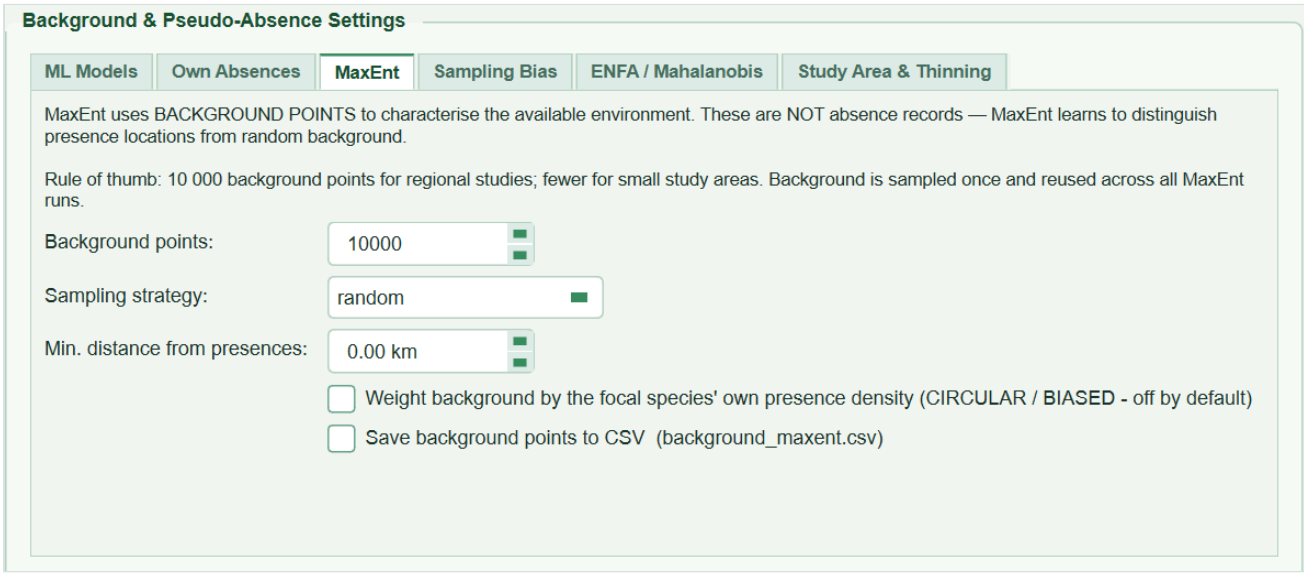


Figure 3.3. MaxEnt sub-tab: Background points 10000, Sampling strategy random, Min. distance from presences 0.00 km; the boxes for weighting by the focal species' own record density (CIRCULAR / BIASED) and for saving the background to CSV are unchecked.

Name in the UI	What it does	Default	Range / options	Code
`Background points:`	Number of MaxEnt background points	10000	100 – 200,000 (step 1000)	`tabs/tab_data.py:789-794`
`Sampling strategy:`	Draw rule	`random`	`random`, `disk`	`tabs/tab_data.py:796-801`
`Min. distance from presences:`	Minimum distance to presences in `disk`	0 km	0 – 500 km	`tabs/tab_data.py:803-806`
`Weight background by the focal species' own presence density (CIRCULAR / BIASED - off by default)`	Weights the background by the record density of the focal species	off	on / off	`tabs/tab_data.py:808-822`, `sdm_core.py:3102-3142`
`Save background points to CSV (background_maxent.csv)`	Writes the background coordinates and predictor values	off	on / off	`tabs/tab_data.py:824-830`, `1549-1563`

3.5.1 Background points and strategy

- **What it does.** The background is drawn once, separately from the pseudo-absences (`sdm_core.py:3144-3166`); the accessible area and (if selected) target-group bias are applied. MaxEnt uses the whole of this background in every fit, including the cross-validation folds (`sdm_core.py:4595-4603`). `disk` is effective only if `Min. distance` is greater than 0; at 0 it is identical to `random`.
- **When and how to set it.** 10,000 is Maxent's common default (Phillips et al. 2006). If a small accessible area has fewer valid cells than this, all cells are used; the background is then a complete census of the area (Phillips and Dudík 2008).
- **Effect on results.** The size of the background determines how well the environmental distribution of the area is represented; the gain above 10,000 is usually small, while run time increases. The **location** of the background (accessible area) is far more influential than its size (§3.8).
- **Caution.** The background does not exclude presence cells; as in Maxent's definition, presences are also part of the available environment. The summary line "Background (MaxEnt): N points" gives the count after points dropped because of NoData have been subtracted.

3.5.2 Weighting by the focal species' own density (circular)

- **What it does.** When on, each valid cell is given as weight the sum of Gaussian kernels computed from its distances to all presences (bandwidth 0.5 units), and the background is drawn with this weight (`sdm_core.py:3119-3136`). The background thus piles up where presences are dense.
- **When and how to set it.** Not recommended; the box's name and tooltip say so explicitly. The focal species' own records carry both the bias and the species' true distribution; the weight cannot separate the two, and the model learns to ignore the environment of the best-sampled areas. A valid correction requires an independent bias surface (Phillips et al. 2009): the `Sampling Bias` sub-tab.
- **Effect on results.** The contrast between presence and background decreases; AUC falls and response curves flatten.
- **Caution.** When switched on, the warning "MaxEnt background weighted by the focal species' own presence density: this is circular and biased…" is written to the log in every run. If a surface is given in `Sampling Bias` and applied to MaxEnt, this box is ignored (`sdm_core.py:3104-3107`). The bandwidth is in raster coordinate units (0.5°, about 55 km, on a geographic grid).

3.6 Sampling bias: Sampling Bias

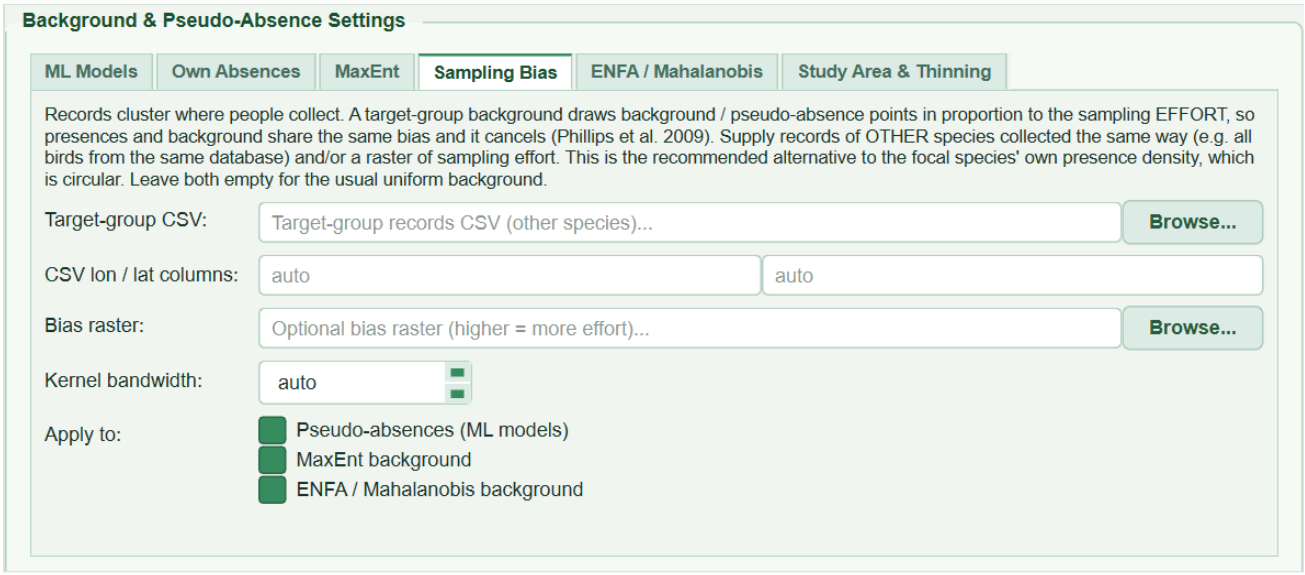


Figure 3.4. Sampling Bias sub-tab: Target-group CSV and Bias raster fields empty, CSV lon / lat columns "auto", Kernel bandwidth "auto"; under Apply to, Pseudo-absences (ML models), MaxEnt background and ENFA / Mahalanobis background are checked.

Name in the UI	What it does	Default	Range / options	Code
`Target-group CSV:`	Records of **other** species collected with the same method	empty (off)	CSV	`tabs/tab_data.py:402-405`
`CSV lon / lat columns:`	Coordinate columns in the target-group CSV	`auto`	column name	`tabs/tab_data.py:407-409`, `sdm_sampling.py:77-93`
`Bias raster:`	Sampling effort raster (large value = high effort)	empty (off)	`.tif .tiff .img .asc`	`tabs/tab_data.py:411-416`, `sdm_sampling.py:161-198`
`Kernel bandwidth:`	Gaussian smoothing (sigma) of the target-group density	`auto` (0)	0 – 5000 km	`tabs/tab_data.py:418-423`, `sdm_sampling.py:103-158`
`Apply to:`	Sets the surface is applied to	all three checked	`Pseudo-absences (ML models)`, `MaxEnt background`, `ENFA / Mahalanobis background`	`tabs/tab_data.py:425-441`, `sdm_core.py:2991-2993`

3.6.1 Target-group surface and bias raster

- **What it does.** Records concentrate where people collect them (around roads, villages, stations). The target-group approach (Phillips et al. 2009) draws the background and pseudo-absences in proportion to the record density of other species that carry the trace of the same effort; because presence and background carry the same bias, the bias cancels out. HABITUS builds a weight grid from two sources (`sdm_core.py:2332-2397`):
  - **Target-group CSV:** records are counted into cells and smoothed with a Gaussian filter; the grid is divided by its maximum value, and cells with zero density are left a floor weight of 10^{-6} (`sdm_sampling.py:103-158`).
  - **Bias raster:** read directly if on the same grid, otherwise resampled bilinearly; NaN, NoData and negative values become 0, and the grid is divided by its maximum value (`sdm_sampling.py:161-198`).
  - If both are given, their product is used. The draw probability is proportional to the cell weight; the weight is combined with the accessible area and strategy constraints (multiplied by the k-means class weight; `sdm_core.py:2272-2289`).
- **When and how to set it.** When sampling effort is clearly unbalanced. The target group should consist of species collected from the same database and with the same method as the focal species (e.g. all woody plant records of the same herbarium). The focal species' own records should not be placed in the target group; this would be circular.
- **Effect on results.** The background shifts towards accessible places; the model confuses "easy to reach" with "suitable" less often. AUC typically falls (discrimination becomes harder), but the map follows environmental conditions rather than roads and settlements (Kramer-Schadt et al. 2013).
- **Caution.** Target-group coordinates must be in the same coordinate system as the raster; if none fall on the grid, the run stops. With fewer than 20 target-group records a warning is written that the density will be very coarse, and when the surface is concentrated in less than 0.1% of the cells, a warning that most of the background will come from there (`sdm_core.py:2352-2393`). A faulty file stops the run; the program does not silently fall back to uniform sampling. If the bias raster does not declare a coordinate system, it is assumed to be in the same system as the predictors and a warning is given; if its grid differs and it has no CRS, it cannot be aligned and an error is raised. When no surface is given, the note "Sampling bias: not corrected" is added to the report (`sdm_core.py:1358-1363`, `2988-2990`).

3.6.2 Kernel bandwidth

- **What it does.** The standard deviation (km) of the Gaussian smoothing. On a geographic grid, km is converted to degrees, corrected in the longitude direction by the cosine of the records' mean latitude. On a projected grid, km is converted using the metre equivalent of the map unit. With `auto` (0), the normal reference rule is used for each axis: $h = 1.06\ \min(s,\ \mathrm{IQR}/1.34)\, n^{-1/5}$ (MASS `bandwidth.nrd`/4, as in `kde2d`). Sigma is never smaller than one cell (`sdm_sampling.py:136-154`).
- **When and how to set it.** Widened if the target-group records are sparse; kept at the level of a few cells if effort is concentrated right around roads.
- **Effect on results.** A small bandwidth sticks the surface to the record points, and the background comes almost only from recorded cells; a large bandwidth flattens the surface and brings it closer to uniform sampling.
- **Caution.** The bandwidth used is written to the run record (`auto (nrd)` or km).

3.6.3 Apply to

The surface is applied only to the checked sets (`sdm_core.py:2991-2993`). Checking all three ensures that all algorithms see the same bias correction and makes comparisons between algorithms consistent. Checking only one is useful for testing the effect of the correction in a single family of algorithms.

3.7 ENFA background: ENFA / Mahalanobis

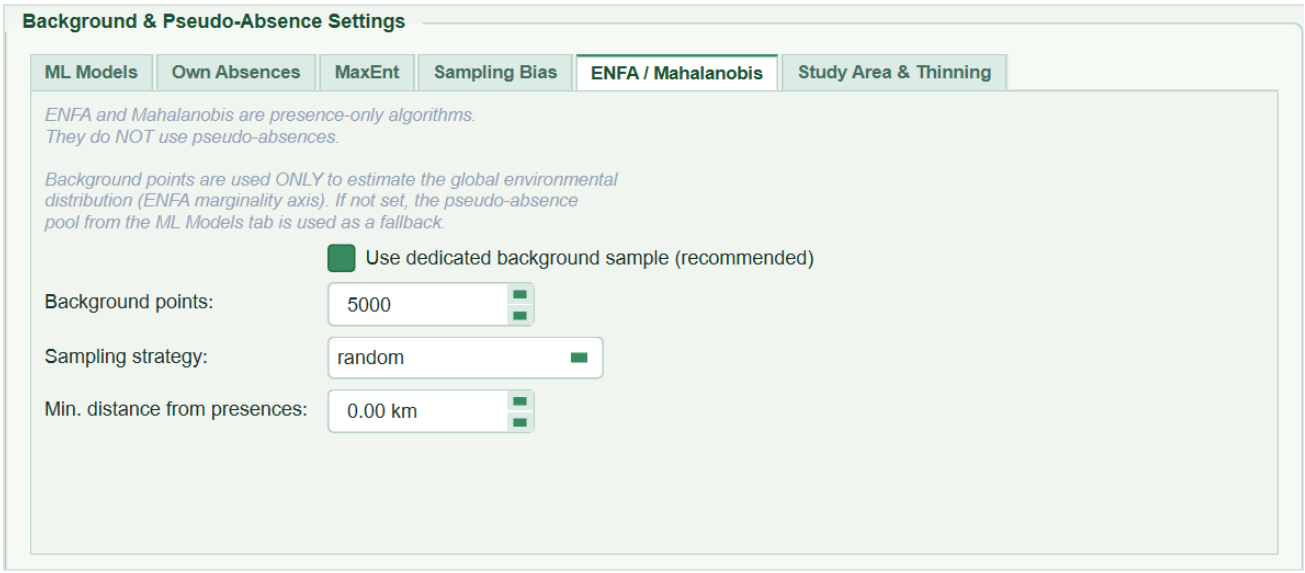


Figure 3.5. ENFA / Mahalanobis sub-tab: Use dedicated background sample (recommended) checked, Background points 5000, Sampling strategy random, Min. distance from presences 0.00 km.

Name in the UI	What it does	Default	Range / options	Code
`Use dedicated background sample (recommended)`	Draws a separate background for ENFA	on	on / off	`tabs/tab_data.py:858-864`
`Background points:`	Number of background points	5000	100 – 100,000 (step 1000)	`tabs/tab_data.py:866-872`
`Sampling strategy:`	Draw rule	`random`	`random`, `disk`	`tabs/tab_data.py:874-879`
`Min. distance from presences:`	Minimum distance in `disk`	0 km	0 – 500 km	`tabs/tab_data.py:881-885`

- **What it does.** ENFA compares the species' conditions with the "global" conditions of the study area: the global mean and covariance are computed from this background (`sdm_core.py:4079-4092`). The background is drawn within the accessible area and (if checked) with the bias surface (`sdm_core.py:3176-3199`). If the box is off, ENFA computes the global statistics from the training rows (presences + pseudo-absences).
- **When and how to set it.** Leaving it on is recommended: the global environment should be estimated from a uniform sample of the area, not from a sample distorted by the pseudo-absence strategy (e.g. `disk`, `sre`). The tooltip considers 5000–10,000 points sufficient.
- **Effect on results.** ENFA's marginality (deviation of the species' mean from the global mean) and specialisation axes are measured against this sample; narrowing the accessible area reduces marginality.

- **Caution.** In the code only ENFA uses this background; the Mahalanobis distance method uses only the mean and covariance of the presences (`sdm_core.py:4207-4230`), and the settings of this tab do not change it. If the background columns do not exactly match the variables in the model, ENFA falls back to the training rows. Selecting `disk` in the strategy box only activates the minimum distance.

3.8 Accessible area: Study Area & Thinning (upper half)

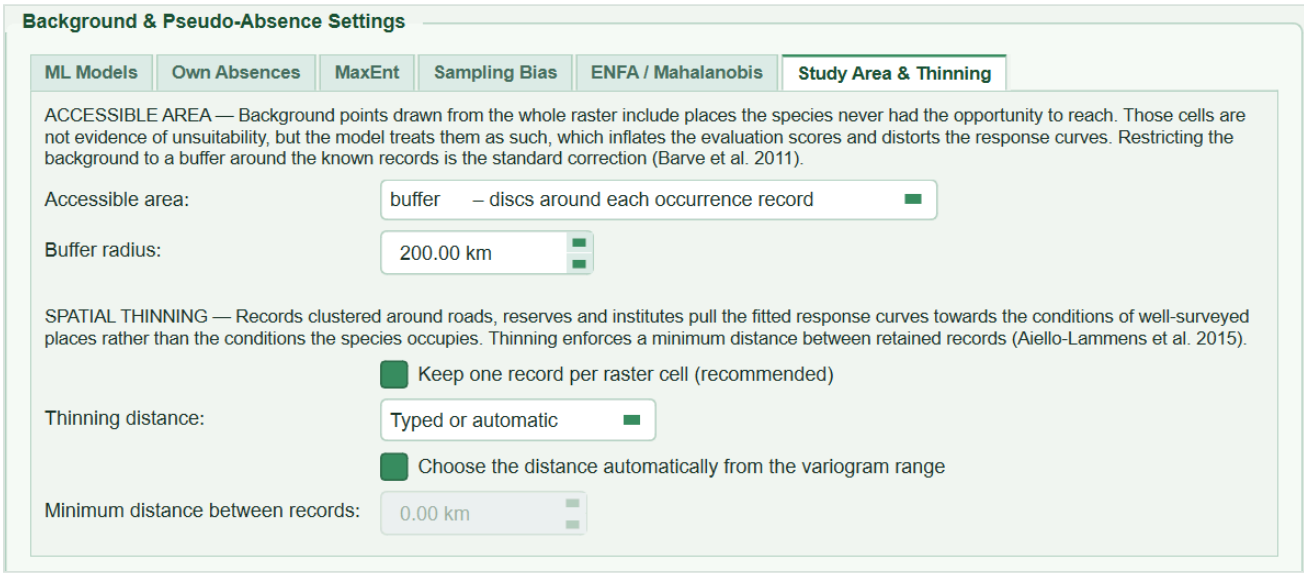


Figure 3.6. Study Area & Thinning sub-tab with the settings of the example run: Accessible area "buffer – discs around each occurrence record", Buffer radius 200.00 km; below, Keep one record per raster cell checked, Thinning distance "Typed or automatic", Choose the distance automatically from the variogram range checked (hence Minimum distance between records greyed out).

Name in the UI	What it does	Default	Range / options	Code
`Accessible area:`	Method for building the background/pseudo-absence area	`buffer`	`buffer – discs around each occurrence record`, `mcp_buffer – buffered minimum convex polygon`, `polygon – my own calibration area layer`, `none – sample the whole raster extent`	`tabs/tab_data.py:644-666`
`Buffer radius:`	Buffer radius; in `polygon`, the buffer added around the polygons	200 km	0 – 5000 km (step 50)	`tabs/tab_data.py:668-674`, `609-618`
`Calibration layer:` + `Browse…`	Calibration polygon layer (visible only in `polygon`)	empty	`.shp .gpkg .geojson .json .kml .gml`	`tabs/tab_data.py:677-691`
`Keep only the polygons containing an occurrence record`	Keeps only polygons that contain records (only in `polygon`)	on	on / off	`tabs/tab_data.py:693-701`

3.8.1 Why an accessible area

A species does not occur in places it has never been able to reach, even if they are suitable. These cells are not evidence of "unsuitable"; but when they enter the model as background or pseudo-absences, the model learns them as such (Barve et al. 2011). The accessible area (M in the BAM framework) restricts the area from which the background is drawn to the region the species can reach. Keeping the extent wide adds cells with conditions very different from the presences to the background; discrimination becomes easier and AUC rises artificially (Lobo et al. 2008; VanDerWal et al. 2009). The mask is applied to all three sets (pseudo-absences, MaxEnt, ENFA); it is not applied to the user's absence records. The map is still produced for the whole extent (or the study boundary).

3.8.2 Accessible area methods

- **What it does.**
  - `buffer`: All cells within `Buffer radius` of the cell of each presence (`sdm_spatial.py:943-1062`). Distance is computed with an anisotropic Euclidean distance transform (`distance_transform_edt`) that converts cell size to km; on a geographic grid, the cell width is reduced by the cosine of the records' mean latitude.
  - `mcp_buffer`: The convex hull (minimum convex polygon) of the presence cells is filled, then the same buffer is added. If the records lie on a line (river, coast), the hull is the line segment between the two most distant records.
  - `polygon`: The user's calibration layer (§3.8.4).
  - `none`: No constraint; all valid cells.
- **When and how to set it.** Rojas-Soto et al. (2024) compared seven calibration-area methods on 31 species and recommended the method that uses biogeographic units (ecoregion, basin, biogeographic province); its counterpart in HABITUS is `polygon` + `Keep only the polygons containing an occurrence record`. Buffer methods are safe and simple but cannot represent barriers (mountain ranges, sea). `mcp_buffer` suits species with continuous ranges; for fragmented ranges the hull also covers the gaps. `none` should be chosen only if the rasters have already been clipped to the area of interest.
- **Effect on results.** As the area narrows, the background resembles presence conditions; AUC/TSS fall but become more honest, and response curves describe the gradient within M. In the example run, the 200 km buffer kept 40.0% of the raster extent.
- **Caution.** The buffer is built from the thinned records (`sdm_core.py:2971`). In `buffer` and `mcp_buffer`, a radius of 0 applies no constraint. If the buffer covers more than 97% of the extent, the warning "The accessible-area buffer covers almost the whole raster extent, so it is not constraining anything." is given (`sdm_core.py:2694-2699`). If the area has fewer cells than the number of points requested, all cells are used, a warning is written, and the draw is not extended to the whole extent (`sdm_core.py:2166-2203`). If the mask cannot be built, the program does not fall back to the whole extent; it stops.

3.8.3 Buffer radius

- **What it does.** In `buffer` and `mcp_buffer`, the radius of M; in `polygon`, the buffer added around the selected polygons (`sdm_spatial.py:1260-1299`).
- **When and how to set it.** As the UI tooltip says, it should be chosen from the species' dispersal ability and the age of the range, not from the size of the raster. Tens of km for species with limited dispersal, hundreds of km for long-distance dispersers.
- **Effect on results.** As the radius grows, the background becomes environmentally more varied and AUC rises; prevalence does not change (the number of points is the same), but the pseudo-absences move away from the presences.
- **Caution.** When switching to `polygon` mode, the field keeps the default value of 200 km; to use the polygons as drawn, the value must be reduced to **0** (tooltip: "Leave it at zero to use the polygon boundaries exactly as drawn"). The example run was run with `buffer` 200 km.

3.8.4 Calibration layer and "containing a record"

- **What it does.** The layer is read with pyogrio; only polygons are used. If its coordinate system differs from the rasters, it is reprojected; if the layer declares no system, it is assumed to be the same and a warning is given (`sdm_spatial.py:1194-1225`). If `Keep only the polygons containing an occurrence record` is on, only polygons in which at least one presence falls are kept (biogeographic unit method); if off, all polygons in the layer are used. An optional buffer is added and the polygons are burned into the grid with `all_touched=True` (every cell touched by a polygon is inside) (`sdm_spatial.py:1234-1321`). The summary states "Calibration area: K of N polygons… X% of the raster extent".
- **When and how to set it.** Loading an entire ecoregion layer without clipping and leaving the box on is the most practical way. If the layer contains only your study area, the box is switched off.
- **Effect on results.** M follows biogeographic boundaries; barriers that buffer methods cannot cross are taken into account.
- **Caution.** If no polygon contains a record, the run stops. With the box off, if no presence lies inside the polygons the warning "Calibration polygon: NONE of the N presence records lies inside the polygon(s)…" is given, and if fewer than 80% of presences lie inside, the warning "only K of N presence records… lie inside" (`sdm_core.py:2644-2672`). If `polygon` is selected and no layer is given, the run stops.

3.8.5 accessible_area.tif

If an accessible area is defined, the mask is written to the species' output folder as `accessible_area.tif` (1 = inside, 0 = outside, compressed uint8; `tabs/tab_data.py:1536-1548`, `sdm_spatial.py:1065-1075`). ⑥ Range Change uses this file to report change within this area separately (Chapter 9). If `none` is selected, the file is not written.

3.9 Thinning the records: Study Area & Thinning (lower half)

Name in the UI	What it does	Default	Range / options	Code
`Keep one record per raster cell (recommended)`	One record in each predictor cell	on	on / off	`tabs/tab_data.py:722-731` (Section 2.8.3)
`Thinning distance:`	Source of the thinning distance	`Typed or automatic`	`Typed or automatic`, `One cell`, `10 km (Boria 2014)`	`tabs/tab_data.py:733-743`
`Choose the distance automatically from the variogram range`	Chooses the distance from the variogram range (only in `Typed or automatic`)	off	on / off	`tabs/tab_data.py:745-751`
`Minimum distance between records:`	Manually entered distance (only in `Typed or automatic` with automatic off)	0 km (no thinning)	0 – 500 km	`tabs/tab_data.py:753-765`

3.9.1 How thinning works

- **What it does.** Records clustered around roads, protected areas and research institutions pull the response curves towards the conditions of well-sampled places rather than the conditions in which the species lives (Aiello-Lammens et al. 2015). Thinning enforces a minimum distance between any two retained records. The algorithm is greedy: records are visited in a seeded random order, and a record closer than the distance to an already retained record is dropped; this is tried with three different orders and the result retaining the most records is chosen (`sdm_spatial.py:888-937`, `sdm_core.py:2536`). Distances are great-circle distances.
- **When and how to set it.** Thinning is applied after the one-per-cell rule (Section 2.8.1). It is switched on if sampling is clearly clustered; if the records are already sparse and regular, it can stay off.
- **Effect on results.** The number of records drops, prevalence drops, and the remaining records are distributed more evenly in geographic space. Boria et al. (2014) showed that thinning reduces overfitting and improves performance in independent evaluation. Dropped records are listed with the reason "spatially thinned"; the distance used and how it was chosen are written to the log, summary, report and run record (`thinning.how_chosen`).
- **Caution.** By default spThin selects the largest set from many random repetitions; HABITUS's 3 repetitions approach this result but do not guarantee the largest set. If fewer than 10 records remain after thinning, a warning is given. If the manually entered distance is smaller than one cell, it is reported that it only separates records in neighbouring cells (`sdm_core.py:2526-2534`).

3.9.2 Presets: One cell and 10 km

- **What it does.** `One cell` sets the distance equal to the edge of a predictor cell: on a geographic grid, the mean of cell height × 110.57 km and cell width × 111.32 km × cos(mean latitude) (`sdm_core.py:2554-2569`). `10 km` is the fixed distance used by Boria et al. (2014) and widely copied (`sdm_core.py:2478-2483`). When a preset is selected, the automatic box and the manually entered distance are greyed out.
- **When and how to set it.** `One cell` is the smallest meaningful distance, tied to predictor resolution, for separating records in neighbouring cells; with 2.5′ data it is about 4 km. `10 km` is a convention, not a rule for every species and resolution.
- **Effect on results.** Both are usually smaller than the automatic distance; they drop fewer records.
- **Caution.** The presets are not subject to the rule of retaining at least 50% of records.

3.9.3 Automatic distance: half the variogram range and the reduction rule

- **What it does.** In every run, even with thinning off, the program estimates the spatial autocorrelation range of the predictors from the record points (`sdm_spatial.py:166-281`): at most 1500 records are subsampled; for each predictor an omnidirectional empirical semivariogram is computed in 15 bins up to half the maximum pairwise distance; the sill is the median semivariance of the outer third of the bins, and the range is the distance at which 95% of this sill is first reached; the median across variables is taken. With fewer than 20 points or no usable variogram, the range is taken as one tenth of the maximum distance (`fallback_extent`). With automatic thinning on, the distance is half the range; if this distance drops more than half of the records (`THIN_MIN_RETAIN = 0.5`), the distance is halved up to 8 times, and the first distance that retains at least 50% is used (`sdm_core.py:2484-2522`, `2571-2594`).
- **When and how to set it.** If there is no species-specific justification for a distance, it is a defensible starting point; because the same range is used as the spatial block size in ③ Models, thinning and block size are consistent with each other (the logic of Roberts et al. 2017 and blockCV [Valavi et al. 2019]).
- **Effect on results.** In the example run the range was found to be 151.5 km; since its half, 75.7 km, would have dropped more than half of the records, the distance was reduced to 37.9 → 18.9 → 9.5 → 4.7 km; at 4.7 km, 143 of 221 records (65%) were retained (run record: "auto (half the variogram range, reduced to keep 65% of records)").
- **Caution.** In climate layers dominated by a regional trend, the range comes out very large and the reduction rule brings the distance down almost to the cell size; in this case the distance no longer removes autocorrelation, only clustering. If every candidate distance drops more than half of the records, no thinning is done and a warning is given; if the range cannot be computed at all, no thinning is done either and the message "Type a thinning distance in ① Data" is shown (`sdm_core.py:2495-2522`). There is no 50% retention limit for a manually entered distance.

3.10 Outputs and what passes to the next step

- `pa_datasets`: `PA repetitions` sets, each consisting of (predictor table, 0/1 labels, coordinates); passed to ③ Models.
- `background_X` and its coordinates: the MaxEnt background; if requested, `background_maxent.csv` (columns `longitude, latitude, <variables>`).
- `background_ponly`: the ENFA background.
- `accessible_area.tif` and accessible area information; thinning information; autocorrelation range (block size in ③).
- The `background`, `study_area` and `thinning` sections of the run record: absence source (`pseudo`, `user`, `user+pseudo`), strategy and its details (candidate cell count, outside-envelope proportion, k-means class counts), background counts, target-group information, accessible area method and proportion, calibration polygons, thinning distance and how it was chosen. In the example run, settings left at their defaults are also listed under `defaults_left_unchanged`.

3.11 Expert questions

Q1. Is the `random` pseudo-absence draw in HABITUS the same as `random` in biomod2? Answer: The rule is the same: an equal-probability draw without replacement from valid cells (`sdm_core.py:2301-2302`). There are two differences. First, HABITUS by default restricts the draw to a 200 km accessible area; biomod2 has no such default and uses the whole extent. Second, pseudo-absences are drawn from cell centres, and within a set a cell is selected only once. Presence cells are not excluded in `random`; a pseudo-absence may fall in a presence cell.

Q2. Why do strategies that move pseudo-absences away from presences inflate the scores? Answer: AUC and TSS measure the model's ability to discriminate presence from absence. When pseudo-absences are moved geographically (`disk`) or environmentally (`sre`, `env_const`) away from presences, discrimination becomes easier by construction of the data; the measured skill comes more from the sampling design than from the model (VanDerWal et al. 2009; Barbet-Massin et al. 2012). When comparing strategies on the same data, it is not the scores but the performance obtained with independent validation data (⑦ Validation) that should be compared.

Q3. Why is the accessible area buffer 200 km, and where does this value come from? Answer: 200 km is a starting value (`tabs/tab_data.py:669`, `sdm_core.py:1400`); in the code it has no species-specific justification. The UI tooltip asks for the radius to be chosen from the species' dispersal ability and the age of the range. Barve et al. (2011) stress that M must be defined specifically for the species and the question; Rojas-Soto et al. (2024) recommend a calibration area based on biogeographic units (`polygon` in HABITUS). The value is written to the record in every run and should be reported.

Q4. Is the buffer distance computed geodesically? Answer: No, it is computed approximately. `buffer` and `mcp_buffer` use an anisotropic Euclidean distance transform that treats cell height as constant and cell width as reduced by the cosine of the records' mean latitude (`sdm_spatial.py:1022-1062`). In areas with a small latitudinal span the error is small; for a range extending thousands of km north–south, the east–west width of the buffer deviates at the extremes. Thinning and minimum distance, by contrast, are computed with haversine.

Q5. Do the user's absence records also enter MaxEnt? Answer: No. `Own Absences` records enter only the absence side of `pa_datasets` (`sdm_core.py:3035-3082`). MaxEnt uses its own background (`sdm_core.py:4595-4598`); the background is not treated as absence. ENFA also uses its own background. There is no path in the code for a user who wants to give true absences to MaxEnt as "background".

Q6. How is the bandwidth of the target-group surface chosen; what is its R equivalent? Answer: With `auto`, $1.06 \min(s, \mathrm{IQR}/1.34)\, n^{-1/5}$ is used for each axis; this is one quarter of the MASS `bandwidth.nrd` value and identical to the sigma that `kde2d` uses internally (`sdm_sampling.py:143-151`). The density is obtained by smoothing the cell counts with a Gaussian filter; mass beyond the boundary is lost (`mode="constant"`). The surface is scaled to its maximum value and floored at 10^{-6}, so that cells with no target records can still be selected, albeit with very low probability. Kramer-Schadt et al. (2013) and Phillips et al. (2009) give the rationale for this approach.

Q7. Why is the automatic thinning distance half the variogram range, and why is it reduced? Answer: The range is the distance at which autocorrelation in the predictors vanishes; half of it is a common starting point for reducing repeated environmental information between records, and because the same range determines the block size, the two decisions are consistent (`sdm_core.py:2433-2442`). In climate layers with a strong regional trend the range comes out at hundreds of km and its half drops most of the records; since the aim of thinning is to remove redundancy, not the data set, the distance is halved until at least 50% of the records remain (`sdm_core.py:2571-2594`). In the example run it was reduced from 75.7 km to 4.7 km.

Q8. Does HABITUS's thinning give the same result as spThin? Answer: The rule is the same (minimum distance between any two retained records, great-circle distance); the algorithm differs. spThin `thin()` performs many random repetitions and selects the set retaining the most records (Aiello-Lammens et al. 2015); HABITUS performs a greedy selection with 3 repetitions (`sdm_spatial.py:888-937`), and a single repetition in the automatic distance search. The number of retained records may be slightly lower than spThin's best; exact equality has not been tested.

Q9. What is the difference between `sre` and `env_const` (bioclim)? Answer: Both use the same computation (`_compute_sre_mask`): the $[q,\ 1-q]$ range of the presences in each variable; a cell within range in all variables is inside the envelope (`sdm_core.py:2079-2119`). In `sre`, q is fixed at 0.05 (`sdm_core.py:3010-3011`); in `env_const` + `bioclim` it is set with `Envelope tail trim`. Variables with values at fewer than 4 presences are not included in the envelope. `env_const` + `pca`, by contrast, works on two principal components and prevents the envelope from becoming meaningless as the number of variables grows.

Q10. Do the background and pseudo-absences all come from the same seed; are the repetitions independent of each other? Answer: `DataFormatter` sets up a single seeded generator and all draws are made from it in sequence (`sdm_core.py:1505`). Because each pseudo-absence set is drawn from the next state of the generator, the sets differ from each other and are reproduced with the same seed in the same order. Changing the number of repetitions also changes the subsequent MaxEnt and ENFA backgrounds, because the generator state has advanced.

Q11. What happens if the accessible area is small and has fewer cells than the requested number of background points? Answer: All valid cells in the area are used, fewer points than requested are drawn and a warning is written (`sdm_core.py:2190-2203`); how many cells were found for which request is added to the run record (`cells_short_of_request`). The background is then a complete census of the area (Phillips and Dudík 2008). The draw is not extended to the whole extent; this preserves the M defined by the user.

Q12. Which settings on this tab affect the result most? Answer: No systematic sensitivity analysis has been carried out with the example data. From the structure of the code and the literature, the following order is expected: (1) accessible area method and radius (determines the environmental coverage of the background; Barve et al. 2011), (2) pseudo-absence strategy (determines the difficulty of discrimination; Barbet-Massin et al. 2012), (3) bias correction (determines whether the map follows bias or environment; Phillips et al. 2009), (4) thinning and the one-per-cell rule (determine the number of records used; from 626 to 143 in the example run). Pseudo-absence and background counts, within a reasonable range, are less influential.

3.12 Common mistakes and pitfalls

- Leaving `Buffer radius` at 200 km in `polygon` mode: the polygons are expanded by 200 km. Enter 0 to use the polygons as they are.
- Using a buffer even when the rasters are already clipped to the study area is not redundant; however, if the buffer covers more than 97% of the extent, the program reports this.
- Using the `geo_const` or `disk` strategy with 0 km: nothing is excluded.
- Supplying the focal species' own records as the `Target-group CSV`, or switching on the circular box in MaxEnt: the bias is not corrected but reinforced.
- Supplying the target-group CSV in a coordinate system different from the rasters: "None of the N target-group points falls inside the raster extent."
- `replace` with few true absences: all threshold-dependent statistics become unstable; use `augment`.
- Comparing AUC/TSS across strategies and choosing the highest: the score difference comes from the design (Q2).
- Assuming that automatic thinning removes autocorrelation for every species: the reduction rule can make the distance very small; read the distance used from the summary.
- Assuming that the ENFA / Mahalanobis tab also changes Mahalanobis: only ENFA uses the background.

3.13 References

Aiello-Lammens, M.E., Boria, R.A., Radosavljevic, A., Vilela, B., Anderson, R.P., 2015. spThin: an R package for spatial thinning of species occurrence records for use in ecological niche models. Ecography 38, 541–545. https://doi.org/10.1111/ecog.01132

Barbet-Massin, M., Jiguet, F., Albert, C.H., Thuiller, W., 2012. Selecting pseudo-absences for species distribution models: how, where and how many? Methods in Ecology and Evolution 3, 327–338. https://doi.org/10.1111/j.2041-210X.2011.00172.x

Barve, N., Barve, V., Jiménez-Valverde, A., Lira-Noriega, A., Maher, S.P., Peterson, A.T., Soberón, J., Villalobos, F., 2011. The crucial role of the accessible area in ecological niche modeling and species distribution modeling. Ecological Modelling 222, 1810–1819. https://doi.org/10.1016/j.ecolmodel.2011.02.011

Boria, R.A., Olson, L.E., Goodman, S.M., Anderson, R.P., 2014. Spatial filtering to reduce sampling bias can improve the performance of ecological niche models. Ecological Modelling 275, 73–77. https://doi.org/10.1016/j.ecolmodel.2013.12.012

Kramer-Schadt, S., Niedballa, J., Pilgrim, J.D., Schröder, B., Lindenborn, J., Reinfelder, V., Stillfried, M., Heckmann, I., Scharf, A.K., Augeri, D.M., Cheyne, S.M., Hearn, A.J., Ross, J., Macdonald, D.W., Mathai, J., Eaton, J., Marshall, A.J., Semiadi, G., Rustam, R., Bernard, H., Alfred, R., Samejima, H., Duckworth, J.W., Breitenmoser-Wuersten, C., Belant, J.L., Hofer, H., Wilting, A., 2013. The importance of correcting for sampling bias in MaxEnt species distribution models. Diversity and Distributions 19, 1366–1379. https://doi.org/10.1111/ddi.12096

Lobo, J.M., Jiménez-Valverde, A., Real, R., 2008. AUC: a misleading measure of the performance of predictive distribution models. Global Ecology and Biogeography 17, 145–151. https://doi.org/10.1111/j.1466-8238.2007.00358.x

Phillips, S.J., Anderson, R.P., Schapire, R.E., 2006. Maximum entropy modeling of species geographic distributions. Ecological Modelling 190, 231–259. https://doi.org/10.1016/j.ecolmodel.2005.03.026

Phillips, S.J., Dudík, M., 2008. Modeling of species distributions with Maxent: new extensions and a comprehensive evaluation. Ecography 31, 161–175. https://doi.org/10.1111/j.0906-7590.2008.5203.x

Phillips, S.J., Dudík, M., Elith, J., Graham, C.H., Lehmann, A., Leathwick, J., Ferrier, S., 2009. Sample selection bias and presence-only distribution models: implications for background and pseudo-absence data. Ecological Applications 19, 181–197. https://doi.org/10.1890/07-2153.1

Roberts, D.R., Bahn, V., Ciuti, S., Boyce, M.S., Elith, J., Guillera-Arroita, G., Hauenstein, S., Lahoz-Monfort, J.J., Schröder, B., Thuiller, W., Warton, D.I., Wintle, B.A., Hartig, F., Dormann, C.F., 2017. Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography 40, 913–929. https://doi.org/10.1111/ecog.02881

Rojas-Soto, O., Forero-Rodríguez, J.S., Galindo-Cruz, A., Mota-Vargas, C., Parra-Henao, K.D., Peña-Peniche, A., Piña-Torres, J., Rojas-Herrera, K., Sánchez-Rodríguez, J.D., Toro-Cardona, F.A., Trinidad-Domínguez, C.D., 2024. Calibration areas in ecological niche and species distribution modelling: Unravelling approaches and concepts. Journal of Biogeography 51, 1416–1428. https://doi.org/10.1111/jbi.14834

Valavi, R., Elith, J., Lahoz-Monfort, J.J., Guillera-Arroita, G., 2019. blockCV: An R package for generating spatially or environmentally separated folds for k-fold cross-validation of species distribution models. Methods in Ecology and Evolution 10, 225–232. https://doi.org/10.1111/2041-210X.13107

VanDerWal, J., Shoo, L.P., Graham, C., Williams, S.E., 2009. Selecting pseudo-absence data for presence-only distribution modeling: How far should you stray from what you know? Ecological Modelling 220, 589–594. https://doi.org/10.1016/j.ecolmodel.2008.11.010

4. Variables (② Variables)

Tab: ② Variables · Code: `tabs/tab_vif.py` (VIF + Correlation), `tabs/tab_vif_advanced.py` (Advanced Analysis), `sdm_core.py` `compute_vif`, `compute_correlation_matrix`, `variable_priority_ranking` (computation)

4.1 What this tab does

② Variables shows the collinearity among the continuous predictors loaded in the ① Data step (collinearity: a predictor being close to a linear combination of the other predictors) and lets the user select the variables that will enter the model. The selection is made by the user. The program only ranks, flags and offers a starting suggestion (`Recommended`); no variable leaves the model without confirmation.

- **Input:** the pseudo-absence (PA) sets produced by ① Data (`state["pa_datasets"]`) and the names of the continuous variables (`formatter.var_names`). Categorical rasters are listed in a separate section; they do not enter the VIF or the correlation.
- **Output:** the selected continuous variables (`state["var_names"]`), the selected and excluded categorical rasters (`state["cat_selected"]`, `state["cat_excluded"]`), and four files written under `figures/` (§4.8).
- **Lock order:** the tab opens once ① Data is complete. The `Confirm Variable Selection → Proceed to ③ Models` button unlocks the ③ Models tab and switches to it (`tabs/tab_vif.py:1206-1208`).

The tab has two sub-tabs: `VIF + Correlation` (where the selection is made) and `Advanced Analysis` (complementary diagnostics with the condition number, principal components, LASSO and Ridge). The second sub-tab does not make a selection; it only provides information.

4.2 Screen layout

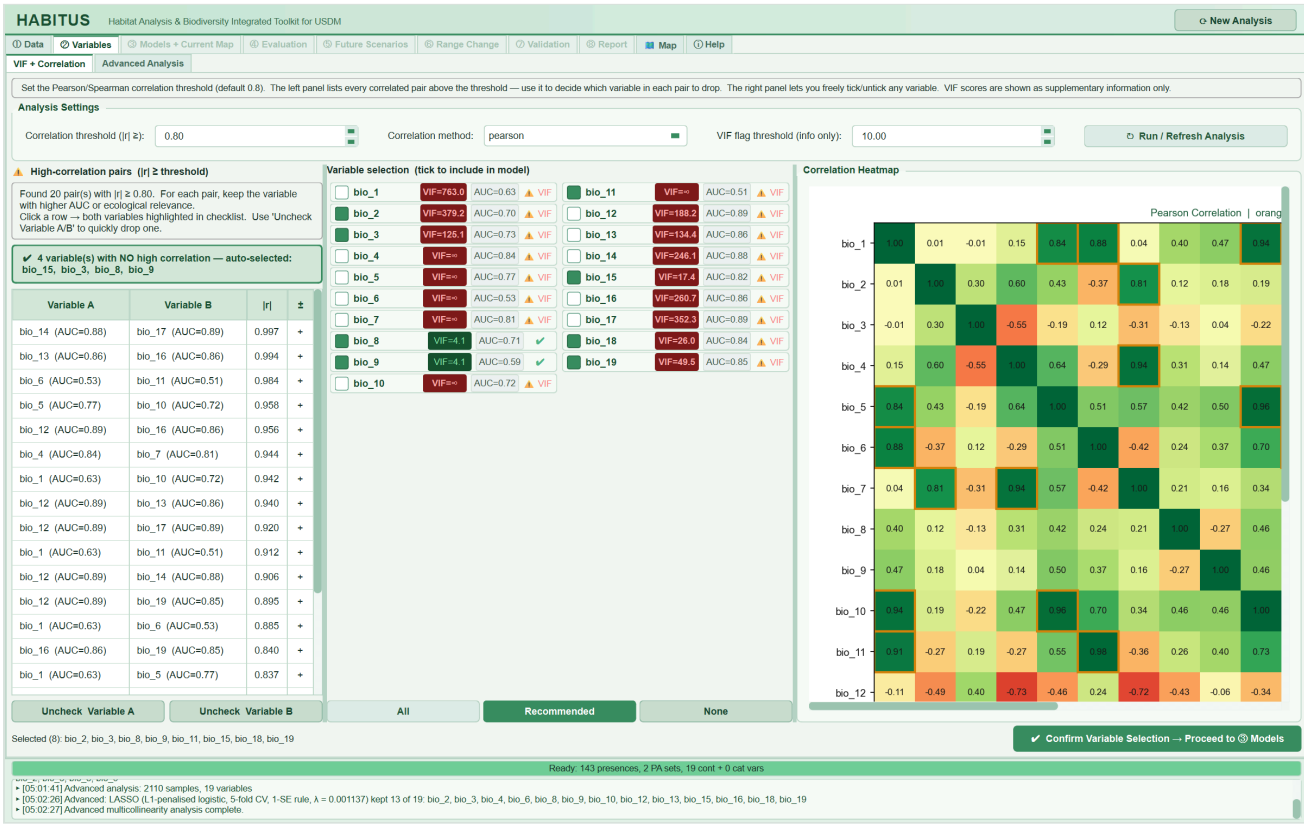


Figure 4.1. VIF + Correlation sub-tab: the Analysis Settings strip at the top, highly correlated pairs on the left, the variable list (with VIF and AUC badges) in the middle, the correlation heatmap on the right, and the selection summary and Confirm button at the bottom.

Figure 4.1 is taken from the example run (*Abies nordmanniana*, 143 presences used, 2 PA sets × 1000 points, 19 WorldClim bioclimatic variables). Regions:

Region	What it contains	Subsection
Top information strip and `Analysis Settings`	Correlation threshold, correlation method, VIF threshold, `Run / Refresh Analysis`	§4.4
Left panel `High-correlation pairs (|r| ≥ threshold)`	Table of pairs exceeding the threshold, "safe" variable box, `Uncheck Variable A/B`	§4.5
Middle panel `Variable selection (tick to include in model)`	For each continuous variable a checkbox, VIF badge, AUC badge and status icon; the categorical section below it; `All`, `Recommended`, `None`	§4.6
Right panel `Correlation Heatmap`	Correlation matrix with square cells; cells exceeding the threshold have an orange border	§4.5.3
Bottom strip	Selection summary ("Selected (n): ...") and `Confirm Variable Selection → Proceed to ③ Models`	§4.7
`Advanced Analysis` sub-tab	`Condition Number`, `PCA`, `LASSO (L1)`, `Ridge (L2)`	§4.9

4.3 Which sample the computations use

All numbers in this tab are computed on a **sample**, not on all raster cells. The sample is the PA sets produced by ① Data stacked on top of each other: each set contains all presences used plus the pseudo-absences of that set (`tabs/tab_vif.py:349-353`, `sdm_core.py:3221-3226`). In the example run this gives 2 × (143 + 1000) = 2286 rows. The MaxEnt background points (10,000) do not enter this pool.

The two sub-tabs prepare the pool differently:

- `VIF + Correlation` uses the pool as it is; it only drops rows with missing values (`dropna`). Each presence appears as many times as there are PA sets (twice in the example).
- `Advanced Analysis` drops duplicate rows: presences that are identical in every PA set are counted once, and pseudo-absences that happen to fall in the same cell in two sets are also deduplicated (`tabs/tab_vif_advanced.py:356-363`). In the example run 2110 rows remain (status line in Figure 4.2: "2110 samples × 19 variables").

> **Note:** VIF and correlation describe the relationships among the predictors themselves; they do not look at the presence/absence label. They do, however, depend on the points at which they are measured. If the accessible area (① Data) is kept large, the pseudo-absences spread over a wider climatic range and the correlations may change. The correlation of the same layers over the whole study area need not be the same as the correlation in this sample.

The univariate AUC (below), by contrast, does look at the label: presences are taken as 1 and pseudo-absences as 0.

4.4 Analysis settings (Analysis Settings)

This strip determines the thresholds with which the pairs table, the heatmap and the `Recommended` suggestion are built.

Name in the UI	What it does	Default	Range / options	Code
`Correlation threshold (|r| ≥):`	Lists pairs whose absolute correlation is equal to or greater than this value and frames them in the heatmap; used in the second step of `Recommended`	0.80	0.30–0.99, step 0.05	`tabs/tab_vif.py:123-130`, `sdm_core.py:435-445`
`Correlation method:`	Pearson (linear) or Spearman (rank) correlation	`pearson`	`pearson`, `spearman`	`tabs/tab_vif.py:132-136`, `sdm_core.py:300-302`
`VIF flag threshold (info only):`	Badge colouring, the `VIF_OK` flag and the stepwise VIF threshold of `Recommended`	10.00	2–50, step 1	`tabs/tab_vif.py:138-143`, `sdm_core.py:405`, `sdm_core.py:427-434`
`↻ Run / Refresh Analysis`	Recomputes the ranking, pairs, list and heatmap with the three values above	—	button	`tabs/tab_vif.py:339-376`

4.4.1 Correlation threshold (`Correlation threshold`)

- **What it does.** Writes every pair with $|r| \geq$ threshold in the upper triangle of the correlation matrix to the left table, sorts them by $|r|$ in descending order (`tabs/tab_vif.py:696-705`) and marks these cells with an orange border in the heatmap (`tabs/tab_vif.py:984-991`). Variables that appear in no pair are counted as "safe" (§4.5.2). Together with `Run / Refresh Analysis`, it also determines the correlation step of `Recommended` (§4.6.2).
- **When and how to set it.** The tooltip gives three common values: 0.70 (conservative), 0.80 (default), 0.90 (lenient). In simulation studies, Dormann et al. (2013) report that $|r| > 0.7$ is the approximate limit at which collinearity begins to seriously distort model estimation and prediction. If the model is to be transferred to another region or climate (future scenario), 0.70 is safer; if only the description of the current distribution is required, 0.80 is often sufficient.
- **Effect on results.** As the threshold decreases, the number of pairs increases, the number of safe variables decreases and `Recommended` retains fewer variables. In the example run, 20 pairs are listed with the 0.80 threshold (Figure 4.1; the strongest pair is bio_14–bio_17, $|r| = 0.997$).
- **Caution.** Changing the value in the box refreshes the pairs table and the heatmap immediately (`tabs/tab_vif.py:880-884`), but it does not refresh the `Recommended` suggestion or the `Corr_OK` flag; for these, `Run / Refresh Analysis` must be pressed. The suggestion shown when the tab first opens comes from the automatic pre-computation of the ① Data step, and this pre-computation uses a threshold of 0.70, the Pearson method and a VIF threshold of 10 (`sdm_core.py:3225-3227`, `sdm_core.py:305-309`), whereas the box shows 0.80. For a suggestion computed with the values in the UI, press `Run / Refresh Analysis` first.

4.4.2 Correlation method (`Correlation method`)

- **What it does.** Computes the Pearson or Spearman matrix with `pandas.DataFrame.corr(method=…)` (`sdm_core.py:300-302`). The pairs table, the heatmap, the mean absolute correlation (`Mean_Corr`) and the correlation step of `Recommended` use this matrix. VIF is not affected by this choice; VIF is always computed by linear regression.
- **When and how to set it.** Tooltip: "pearson: linear correlation (assumes normality)", "spearman: rank-based, robust to non-normality and outliers". Spearman is more appropriate for predictors with skewed distributions, such as precipitation variables, or with monotonic but non-linear relationships.
- **Effect on results.** Because Spearman also captures monotonic non-linear relationships, it usually lists more pairs. Since the VIF step does not change, the first stage of `Recommended` stays the same.
- **Caution.** Nothing is refreshed automatically when the method box is changed (no signal is connected to the box, `tabs/tab_vif.py:132-136`); `Run / Refresh Analysis` must be pressed.

4.4.3 VIF threshold (`VIF flag threshold (info only)`)

- **What it does.** It is used in three places: (1) the VIF badge in the variable list is red if the value exceeds the threshold and green otherwise (`tabs/tab_vif.py:443-462`); (2) the `VIF_OK` flag is true when $\mathrm{VIF} \leq$ threshold and $\mathrm{VIF} < 999$ (`sdm_core.py:405`); (3) after `Run / Refresh Analysis`, the stepwise VIF stage of `Recommended` removes variables down to this threshold (`sdm_core.py:427-434`, `tabs/tab_vif.py:357`).
- **When and how to set it.** 10 is the commonly used limit; the default of usdm::vifstep is also `th = 10`. O'Brien (2007) argues that such thresholds should be read not as rules but as signals to be weighed together with sample size and purpose. For models whose coefficients will be interpreted, such as GLM and GAM, a stricter threshold such as 5 may be considered.
- **Effect on results.** As the threshold decreases, `Recommended` retains fewer variables. It does not remove variables by itself: the ticks in the list do not change; the new suggestion is applied only when the `Recommended` button is pressed.
- **Caution.** The phrase "(info only)" in the UI and the sentence "does NOT force deselection" in the tooltip mean that the threshold does not forcibly remove any variable. The threshold nevertheless determines the `Recommended` suggestion. Changing the value in the box does not refresh anything on its own; `Run / Refresh Analysis` is required.

4.4.4 How VIF is computed

Each continuous variable x_i is regressed on all the other continuous variables by linear regression with an intercept (`sklearn.linear_model.LinearRegression`), and

$$\mathrm{VIF}_i = \frac{1}{1 - R_i^2}$$

is computed (`sdm_core.py:221-297`). Details:

- Only rows in which x_i and all the other variables are finite are used; if there are fewer than 10 such rows or the variable is constant, the result is `NaN` and the badge shows `VIF=–` (`sdm_core.py:269-282`).
- R^2 is clipped to the [0, 1] interval to guard against floating-point overflow. If $R^2 > 0.9999$ (true VIF above 10,000), the value is **capped at 999** and the badge shows `VIF=∞` (`sdm_core.py:286-295`, `tabs/tab_vif.py:447-451`). This is normal for WorldClim variables derived from one another (e.g. bio_7 = bio_5 − bio_6).
- At least two variables are required; with a single variable VIF is undefined and the ranking table writes VIF = 1 for each variable (`sdm_core.py:356-368`).

4.5 Highly correlated pairs and the heatmap

The left panel and the heatmap on the right are built from the same correlation matrix. The aim is to show pairs of variables that substitute for each other and to make it easier for the user to decide which member of each pair to keep.

Name in the UI	What it does	Default	Range / options	Code
Pairs table (`Variable A`, `Variable B`, `|r|`, `±`)	Each pair exceeding the threshold; univariate AUC next to the name; absolute r and sign	—	—	`tabs/tab_vif.py: 682-801`
Clicking a pair row	Highlights the frames of the two variables in the list in orange and scrolls them into view	—	—	`tabs/tab_vif.py: 827-865`
`Uncheck Variable A` / `Uncheck Variable B`	Unticks the first / second variable of the selected pair	—	button	`tabs/tab_vif.py: 209-221`, `867-878`
Safe variable box	Variables that appear in no pair ("auto-selected")	—	—	`tabs/tab_vif.py: 707-729`
`Correlation Heatmap`	Matrix with square cells; RdYlGn colour scale, −1 …+1	—	—	`tabs/tab_vif.py: 937-1009`

4.5.1 Columns of the pairs table

- `Variable A`, `Variable B`: the two variables of the pair, in their natural order in the matrix (bio_1 first, bio_10 after). The univariate AUC of the variable is shown in parentheses next to its name, and of the two variables the one with the lower AUC is shown in red and the one with the higher AUC in green (`tabs/tab_vif.py:752-772`).
- `|r|`: absolute correlation, three decimals. Colour: ≥ 0.95 red, ≥ 0.85 orange, below that neutral (`tabs/tab_vif.py:777-785`).
- `±`: sign of the correlation (`+` or `−`).

The information text above the panel gives the number of pairs ("Found 20 pair(s) with |r| ≥ 0.80"). If there are no pairs, it reads "No variable pairs exceed |r| ≥ … All variables can be safely used together."

> **Caution:** The AUC colouring of the pairs table and the advice "keep the variable with higher AUC or ecological relevance" in the information text are not the same as the criterion of `Recommended`: `Recommended` removes the member of a pair with the higher VIF and does not look at AUC (§4.6.2). The univariate AUC is for information only; a variable that discriminates well in this sample may be doing so because of its correlation with the true driver and may fail when transferred (`sdm_core.py:323-339`). Which member of a pair to keep should be decided primarily by the ecology of the species.

4.5.2 Safe variables

Variables that appear in no pair exceeding the threshold are listed in a green box under the heading "N variable(s) with NO high correlation — auto-selected". When the list is rebuilt, these variables always come ticked (`tabs/tab_vif.py:427-431`). In the example run, four variables are safe with the 0.80 threshold: bio_15, bio_3, bio_8, bio_9 (Figure 4.1).

> **Note:** "Safe" refers only to pairwise correlation. A variable may be close to a combination of several variables without being strongly related to any single one; VIF shows this. In the example, bio_3 is in no pair, but its VIF in the full set is 125.1. `Recommended` does not add the safe variables automatically; it only applies the result of the stepwise selection (`tabs/tab_vif.py:900-915`).

4.5.3 Correlation heatmap

The heatmap shows the matrix of all continuous variables computed with the selected method. Cells are square and of fixed size (0.5 inch for up to 25 variables, 0.4 inch for more); a large matrix is not compressed but navigated with a scroll bar (`tabs/tab_vif.py:953-963`). For up to 25 variables, cell values are printed with two decimals. Off-diagonal cells exceeding the threshold have an orange border; the title has the form "Pearson Correlation | orange border = |r| ≥ 0.80". At every redraw the figure is saved as `figures/correlation_heatmap.png` at 300 dpi (if an output folder is defined; `tabs/tab_vif.py:1006-1009`).

4.6 Variable list and bulk selection

The list in the middle panel is where the continuous variables that will enter the model are selected. If there are more than 12 variables, it is split into two columns (`tabs/tab_vif.py:522-528`).

Name in the UI	What it does	Default	Range / options	Code
Variable checkbox	A ticked variable enters the model	safe variables and the `Recommended` suggestion ticked	on / off	`tabs/tab_vif.py:423-438`
VIF badge	VIF in the full set; green ≤ threshold, red > threshold, `VIF=∞` capped, `VIF=–` could not be computed	—	—	`tabs/tab_vif.py:441-469`
AUC badge	Univariate AUC; neutral colour, information only	—	0.5–1	`tabs/tab_vif.py:471-499`, `sdm_core.py:380-393`
Status icon	`⚠ VIF` (VIF_OK false), `⚠ Corr` (Corr_OK false), `✓` (recommended), `–`	—	—	`tabs/tab_vif.py:501-517`
`All`	Ticks all continuous variables	—	button	`tabs/tab_vif.py:890-893`
`Recommended`	Applies the result of the stepwise selection and writes the removals to the log	—	button	`tabs/tab_vif.py:900-915`
`None`	Unticks all continuous variables	—	button	`tabs/tab_vif.py:895-898`

4.6.1 Priority table and its columns

`variable_priority_ranking` produces one row per variable; the list badges are read from this table, and on confirmation the table is written as `figures/variable_priority_ranking.csv` (`sdm_core.py:305-465`, `tabs/tab_vif.py:1083-1096`). Rows are sorted by `Priority_Score` in descending order.

Column	Meaning
`Variable`	Variable name
`VIF`	VIF in the full set (two decimals; 999 = capped)
`Mean_Corr`	Mean absolute correlation with the other variables (diagonal taken as 0)
`Univariate_AUC`	Ability of the variable on its own to separate presences from pseudo-absences; if below 0.5, $1 - \mathrm{AUC}$ is taken
`Priority_Score`	$0.5\,(1 - (\min(\mathrm{VIF}, 50) - 1)/49) + 0.5\,(1 - \mathrm{Mean_Corr})$, clipped to $[0, 1]$
`VIF_OK`	$\mathrm{VIF} \leq$ threshold and $\mathrm{VIF} < 999$
`Corr_OK`	`Mean_Corr` < correlation threshold
`Recommended`	Whether it survived the stepwise selection
`VIF_in_selection`	VIF within the recommended set (empty for variables not recommended)

`Priority_Score` is based only on collinearity; the univariate AUC sits in the table as information but does not enter the ranking as a weight (`sdm_core.py:395-401`). The rationale is explicit in the code: promoting a predictor that discriminates well in this sample would be a selection that ignores ecological meaning and transferability to another region or climate.

> **Note:** `Corr_OK` looks not at individual pairs but at the **mean** absolute correlation of the variable. In the example run, with the 0.70 threshold, `Corr_OK` is true for all 19 variables (the highest `Mean_Corr` is 0.533, for bio_12), whereas 20 pairs exceed 0.80. This is why the ⚠ `Corr` icon is rarely seen; pairs are shown by the left table.

4.6.2 Recommended selection (`Recommended`): stepwise VIF and correlated pairs

- **What it does.** The `Recommended` button applies the `Recommended` column of `variable_priority_ranking` to the list; it leaves no other variable ticked (`tabs/tab_vif.py:907-909`). This column is obtained through a two-stage procedure (`sdm_core.py:427-447`):

  1. **Stepwise VIF.** Start with all continuous variables. The VIFs of the remaining set are recomputed; if the largest VIF exceeds the threshold, that variable is removed. Stop when the largest VIF is equal to or below the threshold, or when a single variable remains.
  2. **Correlated pair.** Among the remaining variables, the strongest of the pairs with $|r| \geq$ correlation threshold is found; of the two members of the pair, the one with the larger VIF in the remaining set is removed (the first one if tied). This continues until no such pair remains.

  Each removed variable is written to the log in order, with its VIF or $|r|$ value at the moment of removal.

- **When and how to set it.** It is used as a starting point. The tooltip says so too: "A starting point: check it against what you know of the species." First set the thresholds, press `Run / Refresh Analysis`, then press `Recommended` and correct the result manually according to the ecology of the species (for example, keeping the ecologically more meaningful member of a pair).

- **Effect on results.** In the example run the stepwise selection removed 11 variables in this order: bio_4 (VIF 999), bio_5 (999), bio_1 (757.27), bio_7 (653.22), bio_17 (346.39), bio_16 (236.16), bio_6 (130.1), bio_12 (80.89), bio_10 (74.4), bio_14 (26.21), bio_13 (15.98). The remaining eight variables are bio_2, bio_3, bio_8, bio_9, bio_11, bio_15, bio_18, bio_19; the VIFs within this set range from 1.65 (bio_15) to 6.43 (bio_18), and since no pair exceeding the correlation threshold remained among them, the second stage removed nothing (`_calisma/figures/variable_priority_ranking.csv`, session log). In the full set only two variables (bio_8 and bio_9) have a VIF below 10; filtering by full-set VIF would therefore leave very few variables. The stepwise method avoids this problem.
- **Caution.** The stepwise method works not on the VIF in the full set but on the VIF recomputed at each step. A variable showing `VIF=∞` in the full set (bio_11 in the example) can therefore remain in the recommended set; the badge in the list describes the full set, the `VIF_in_selection` column the recommended set. When capped values (999) are tied, the first variable in natural order is removed (`v_now.idxmax()`, `sdm_core.py:430`); in this case the removal order may differ from that of a computation that does not cap VIF (§4.11, Q2).

Same result as usdm::vifstep

The stepwise VIF stage does what the `vifstep` function of the R package usdm (Naimi et al. 2014) does. For this book the two were compared on the same matrix: 19 WorldClim 2.1 bioclimatic variables at 5000 land cells drawn at random within the study extent (10° W–50° E, 30–46° N); `variable_priority_ranking(..., vif_threshold=10)` in HABITUS, `usdm::vifstep(d, th = 10)` in R (script: `habitus_dogrulama\H3_yazilim_tanitim\betikler\Y10_vif_usdm.py`). The two programs removed 11 variables **in the same order** (bio_5, bio_10, bio_11, bio_16, bio_7, bio_17, bio_1, bio_12, bio_3, bio_13, bio_18), and the VIFs of the remaining eight variables agree to the fourteenth decimal:

Variable	usdm::vifstep VIF	HABITUS VIF
bio_2	2.375571	2.375571
bio_4	2.524650	2.524650
bio_6	6.864633	6.864633
bio_8	2.787451	2.787451

Variable	usdm::vifstep VIF	HABITUS VIF
bio_9	6.130096	6.130096
bio_14	3.548789	3.548789
bio_15	3.131781	3.131781
bio_19	2.161437	2.161437

(Source: `habitus_dogrulama\H3_yazilim_tanitim\yazim\dogrulama_vif_usdm.csv`; the table is rounded to six decimals.) In the comparison the correlation threshold was kept at 0.99, i.e. the second stage was disabled; usdm has no counterpart of this stage (usdm::vifcor applies correlation first and with a different rule).

4.6.3 Univariate AUC

For each variable `sklearn.metrics.roc_auc_score(y, x)` is computed; if it is below 0.5, the relationship is treated as reversed and $1 - \mathrm{AUC}$ is written (`sdm_core.py:380-393`). If there are fewer than 10 valid rows or the computation fails, 0.5 is written. The badge is deliberately neutral in colour: a low AUC is not a reason to remove a variable (`tabs/tab_vif.py:471-499`). In the example run the highest values are for precipitation variables (bio_12 0.887; bio_17 0.886) and the lowest for temperature variables (bio_11 0.51; bio_6 0.533).

4.6.4 Categorical variable section

If the data contain categorical rasters, a fixed box titled `Categorical variables (not part of VIF)` opens below the list (`tabs/tab_vif.py:557-671`). Each categorical raster is one row:

Item	What it does	Default	Code
Checkbox	A ticked raster enters the model	ticked (or the previous confirmation, if any)	`tabs/tab_vif.py:602-616`
Class badge (`N classes`)	Number of classes; the tooltip gives "presences / records" per class (first PA set)	—	`tabs/tab_vif.py:618-639`
Class count text	"class:presences/records" values of the first four classes	—	`tabs/tab_vif.py:641-651`
Sparse class warning	Number of classes with fewer than 3 presences ("⚠ k classes <3 presences")	—	`tabs/tab_vif.py:653-664`, `sdm_core.py:1264`

Categorical rasters do not enter the VIF or the correlation; `All`, `Recommended` and `None` do not touch this box. The tooltip states how each algorithm uses a categorical variable: GLM, GLMNET, SVM and ANN through dummy (one-hot) columns; GAM through a factor term; MaxEnt through the categorical feature class; tree-based methods through integer splits; BIOCLIM, Domain, ENFA, Mahalanobis and ESM use only continuous variables (`tabs/tab_vif.py:536-545`). If records were dropped in ① Data solely because of the NoData of one raster, the tooltip reports this with the counts and notes that excluding the raster does not redo the sampling. The example run has no categorical raster, so the box is not shown.

4.7 Confirmation (Confirm) and the records-per-predictor check

Name in the UI	What it does	Default	Code
`✓ Confirm Variable Selection → Proceed to ③ Models`	Saves the selection, writes the files, unlocks ③ Models	—	`tabs/tab_vif.py:1160-1225`

4.7.1 Confirmation

- **What it does.** Writes the ticked continuous variables in natural order to `state["var_names"]`, the categorical selection to `state["cat_selected"]` and the excluded ones to `state["cat_excluded"]` (`tabs/tab_vif.py:1187-1191`). The order in which variables are ticked does not matter; the selection is always sorted by natural name order (`sdm_core.canonical_order`). It then writes the files in §4.8, unlocks ③ Models and switches to its `Algorithm Settings` sub-tab.
- **When and how to set it.** Press it once when the selection is finished. If the selection is changed later, it must be pressed again; otherwise ③ Models uses the old selection.
- **Effect on results.** All subsequent steps (models, future projections, novelty maps) use only the confirmed variables.
- **Caution.** At least two continuous variables are required: if nothing is selected, the message "Please select at least one variable." appears; with a single variable, "At least 2 variables are needed for modelling. Please select more variables." appears (`tabs/tab_vif.py:1162-1167`). If models have already been run, the "Re-confirm Variable Selection" dialog states that confirming will delete all models, maps, projections and evaluation results; if `Yes` is chosen, these are deleted (`tabs/tab_vif.py:1170-1185`, `main_dialog.py:528-542`).

4.7.2 Records-per-predictor check

After confirmation the program divides the number of presences used by the number of selected predictors (continuous + categorical) (`sdm_core.py:2889-2909`). If the ratio is below 10, it adds a warning of the following form at the top of the information dialog and to the log ("ADVISORY: ..."): "... presence records for the k predictors selected (... per predictor). Below about 10 per predictor the model is prone to overfitting and the evaluation statistics become unstable; n records support about n/10 predictors." This repeats, for the selected set, the same check made in ① Data; ① Data can only see the number of loaded layers. In the example run, with 143 presences and 19 layers, ① Data had issued the warning (7.5 records per predictor; "143 records allow about 14"); with the 8 selected variables the ratio is 17.9, so no warning appeared on confirmation. A new confirmation replaces the old warning.

4.8 Export

On confirmation the `VIF + Correlation` sub-tab writes the following to the `figures/` subfolder of the output folder (`tabs/tab_vif.py:1193-1197`):

File	Content
`high_correlation_pairs.csv`	`Variable_A`, `Variable_B`, `Correlation_r`, `AUC_A`, `AUC_B`, `Direction`, `Recommend_drop` (pairs exceeding the threshold)
`variable_priority_ranking.csv`	The table in §4.6.1
`vif_barplot.png`	Full-set VIF bars; threshold line (dashed) and line at 10 (dotted); bars are truncated at 50, with `∞` and "(>50, capped)" labels
`correlation_heatmap_selected.png`	Correlation matrix of the selected variables only

In addition, `correlation_heatmap.png` is written at every heatmap redraw. If no output folder is defined, no file is written.

> **Caution:** The `Recommend_drop` column in `high_correlation_pairs.csv` gives the member of the pair with the lower univariate AUC (`tabs/tab_vif.py:1066-1070`). This differs from the rule of `Recommended` (remove the one with the higher VIF) and is an AUC comparison, not a recommendation.

The `Save All Results...` button of `Advanced Analysis` additionally writes to the `advanced_analysis/` folder (§4.9.6). The variables section of the report (⑧) and the ODMAP export read the correlation threshold, method and VIF threshold of this tab (`tabs/tab_report.py:528-530`, `odmap.py:487-492`).

4.9 Advanced analysis (Advanced Analysis)

This sub-tab offers four complementary diagnostics. None of them changes the ticks in the variable list; their suggestions (`RECOMMENDED` in PCA, `SELECTED` in LASSO) are for information only.

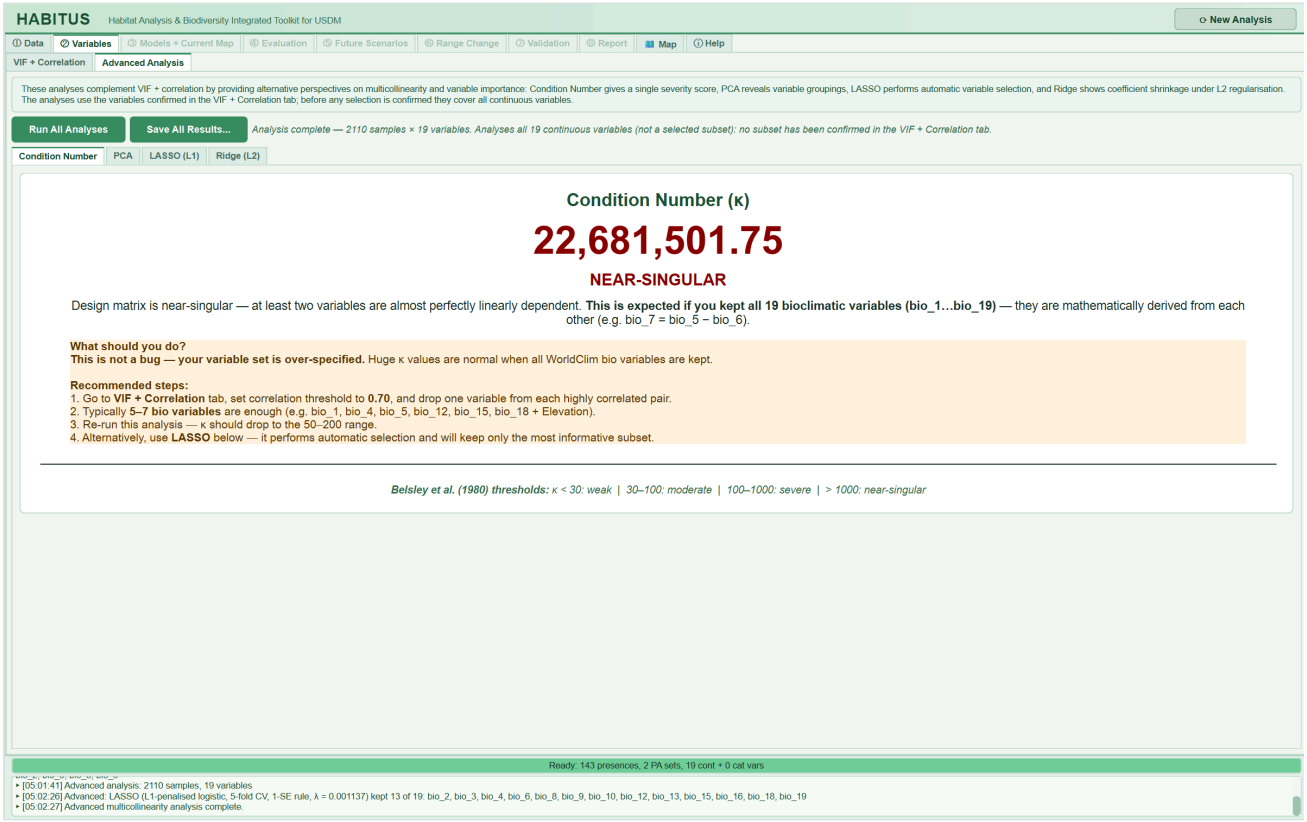


Figure 4.2. Advanced Analysis > Condition Number: condition number computed with all 19 continuous variables (22,681,501.75, NEAR-SINGULAR), the to-do box and the Belsley thresholds.

Name in the UI	What it does	Default	Range / options	Code
`Run All Analyses`	Runs the four analyses in turn	—	button	`tabs/tab_vif_advanced.py:384-407`
`Save All Results...`	Writes the plots and tables to `advanced_analysis/`	disabled until the first run	button	`tabs/tab_vif_advanced.py:760-891`
`Penalty λ:` (LASSO)	How the penalty is chosen	`1-SE rule (CV)`	`1-SE rule (CV)`, `Minimum CV deviance`, `Typed value`	`tabs/tab_vif_advanced.py:259-269`
λ value box (LASSO)	λ used when `Typed value` is selected	0.0100 (disabled)	0.0001–10, 4 decimals	`tabs/tab_vif_advanced.py:270-281`
`CV folds:`	Number of folds of stratified cross-validation for LASSO and Ridge	5	3–10	`tabs/tab_vif_advanced.py:282-284`
`Recompute`	Recomputes LASSO only	—	button	`tabs/tab_vif_advanced.py:285-287`, `409-415`

4.9.1 Scope and sample

The analysis uses the variables confirmed in `VIF + Correlation`. If there is no confirmation yet, or if the confirmed set covers all continuous variables, all continuous variables are used and the status line says so: "Analyses all 19 continuous variables (not a selected subset): no subset has been confirmed in the VIF + Correlation tab." (`tabs/tab_vif_advanced.py:337-345`, `375-380`). The sample is the deduplicated PA pool of §4.3; constant columns (variance below 10^{-10}) are dropped and written to the log (`tabs/tab_vif_advanced.py:365-373`). The recommended order of work is: first diagnose with all variables, then, after confirmation, run `Run All Analyses` again with the selected set.

4.9.2 Condition number (`Condition Number`)

- **What it does.** The variables are standardized (mean 0, standard deviation 1), the singular values of the matrix are computed and $\kappa = s_{\max}/s_{\min}$ is reported (`tabs/tab_vif_advanced.py:419-431`). Singular values below 10^{-10} are discarded; if any are discarded, a note on near-linear dependence is added to the text. Classification: $\kappa < 30$ `LOW`, $30 \le \kappa < 100$ `MODERATE`, $100 \le \kappa < 1000$ `SEVERE`, $\kappa \ge 1000$ `NEAR-SINGULAR`; for each class a "What should you do?" box lists the actions to take.
- **When and how to set it.** It has no settings. It is useful to look at it once for the full set and once after the selection: κ is expected to drop after the selection.
- **Effect on results.** Diagnostic only. In the example run, with 19 variables, κ = 22,681,501.75 (Figure 4.2); this is an expected value when all the mutually derived bioclimatic variables are retained.
- **Caution.** The thresholds on the screen are attributed to Belsley et al. (1980). Belsley's criterion consists of the condition indices and variance-decomposition proportions for each singular value; a single κ and this four-class scale are a practical summary. The suggestions in the to-do box of the NEAR-SINGULAR class ("Typically 5–7 bio variables are enough …", "κ should drop to the 50–200 range") are general guidance, not a computation specific to these data.

4.9.3 Principal components (PCA)

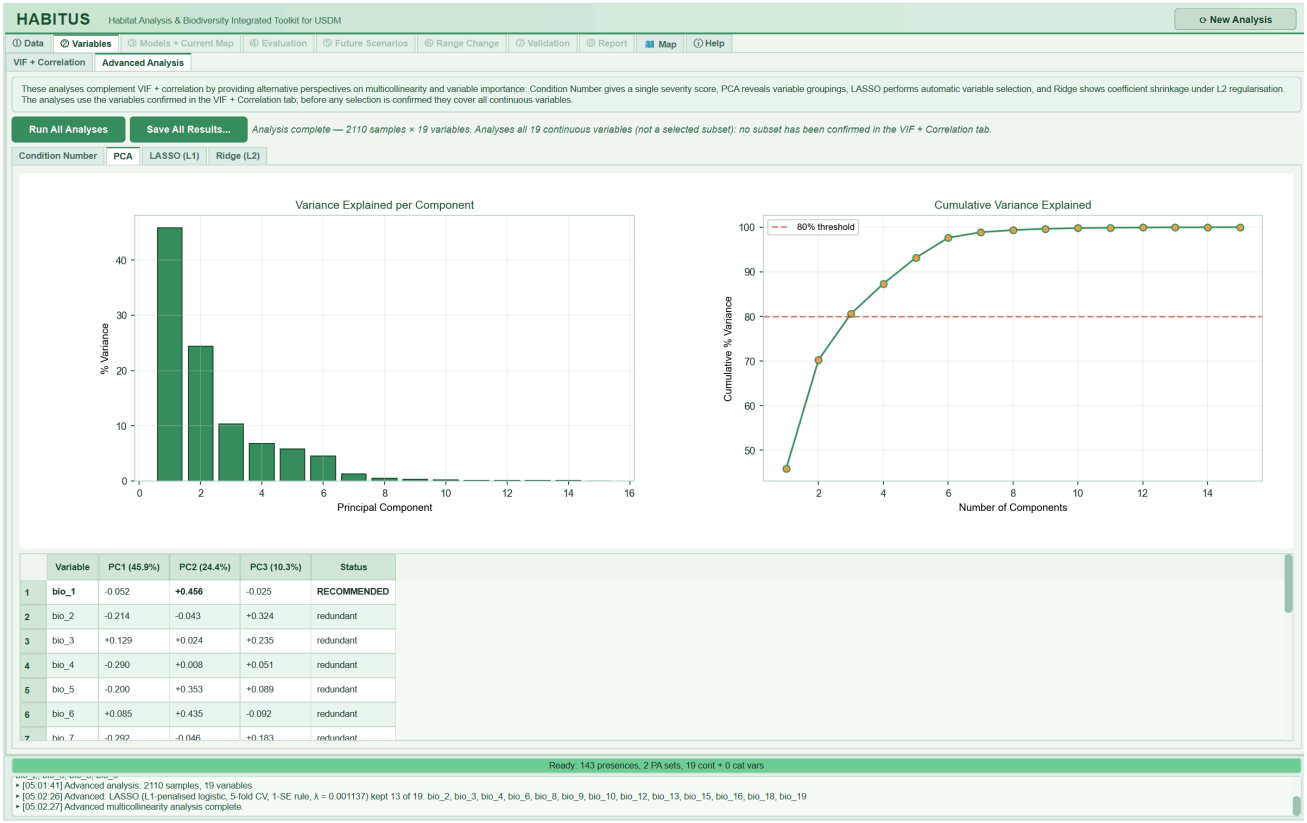


Figure 4.3. Advanced Analysis > PCA: variance explained per component on the left, cumulative variance and the 80 percent line on the right; loadings of the first three components and the Status column below.

- **What it does.** `sklearn.decomposition.PCA` is applied to the standardized variables (`tabs/tab_vif_advanced.py:499-596`). The percentage of variance explained by the first 15 components is plotted on the left, and the cumulative variance with an 80% line on the right. The table shows the loadings of the k components whose cumulative variance reaches 80% (at least 2, at most 8); the column headers give the variance share of the component ("PC1 (45.9%)"). In each component the variable with the largest absolute loading is highlighted in bold green and receives `RECOMMENDED` in the `Status` column; the other variables read `redundant`. Other cells with $|\text{loading}| > 0.4$ are shaded light yellow.
- **When and how to set it.** It has no settings. It is used to see in which groups the variables cluster (for example, temperature and precipitation axes).
- **Effect on results.** Information only. In the example run the first three components explain 45.9%, 24.4% and 10.3% of the variance; since the cumulative 80% is exceeded at the third component, the table has three components and at most three variables receive `RECOMMENDED` (in the rows visible in Figure 4.3, bio_1, with a loading of +0.456 on PC2).
- **Caution.** The rule of "the variable with the largest loading" per component is a heuristic: if the same variable has the largest loading in two components, the number of recommended variables is less than k. The 80% limit is also conventional, a fixed value in the code (`tabs/tab_vif_advanced.py:539-540`).

4.9.4 LASSO (L1) — penalized logistic regression

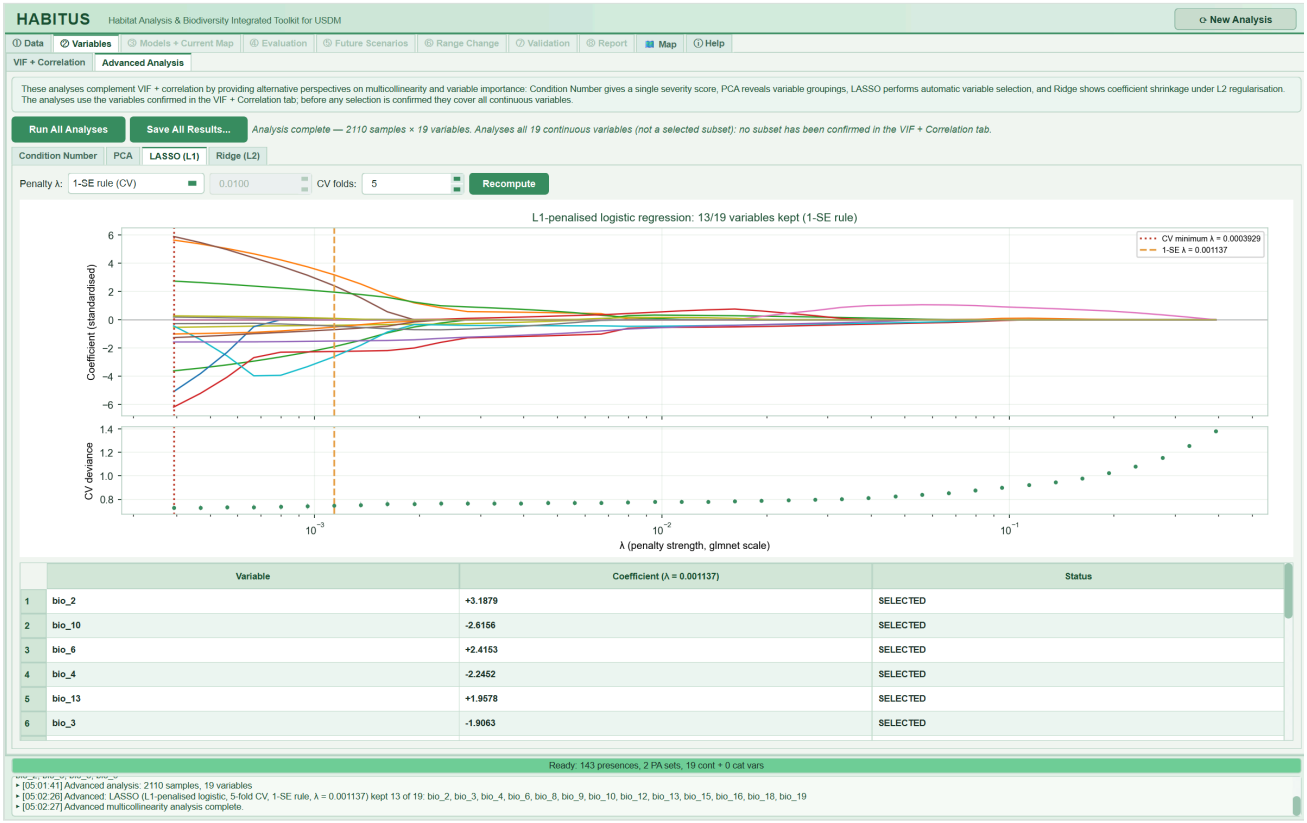


Figure 4.4. Advanced Analysis > LASSO (L1): paths of the standardized coefficients against λ at the top, cross-validation deviance below; the red dotted line marks the λ of minimum deviance, the orange dashed line the λ of the 1-SE rule; coefficient table and Status column at the bottom.

- **What it does.** An L1-penalized logistic regression (Tibshirani 1996) is fitted to the presence (1) / pseudo-absence (0) response with standardized predictors (`tabs/tab_vif_advanced.py:43-128`, `638-711`). Details:
  - Presences and absences receive **equal total weight** (`compute_sample_weight("balanced")`; Barbet-Massin et al. 2012), so that the large number of pseudo-absences does not dominate the result.
  - λ is on the glmnet scale: mean weighted binomial deviance + λ × penalty; scikit-learn's `C` is taken as $1/(n\lambda)$. The 40-value logarithmic grid starts at the smallest λ at which no variable remains ($\lambda_{\max}$) and descends to $\lambda_{\max}/1000$.
  - For each λ the weighted binomial deviance is computed by stratified k-fold cross-validation; in each fold standardization uses the training part only. The number of folds is set by `CV folds`, but no larger than the number of observations in the minority class and at least 2.
  - `1-SE rule (CV)`: the largest λ whose deviance lies within one standard error of the minimum deviance (Hastie et al. 2009, §7.10). `Minimum CV deviance`: the λ with the smallest deviance. `Typed value`: a single fit to all data with the λ typed in the box.
  - Variables with an absolute coefficient larger than 10^{-8} read `SELECTED`, the others `dropped`. The table is sorted by absolute coefficient.
- **When and how to set it.** The default `1-SE rule (CV)` is the common choice for variable selection: it retains fewer and more stable variables. If predictive power is the focus, `Minimum CV deviance` can be chosen. With very few presences, reducing the number of folds (3) may improve the stability of the folds.

- **Effect on results.** Information only; it is not passed to the model settings. In the example run (19 variables, 2110 rows), with the 1-SE λ = 0.001137, 13 of the 19 variables were retained (bio_2, bio_3, bio_4, bio_6, bio_8, bio_9, bio_10, bio_12, bio_13, bio_15, bio_16, bio_18, bio_19); the λ of minimum deviance is 0.0003929 (Figure 4.4).
- **Caution.** LASSO picks one of a group of strongly related variables almost arbitrarily; selection among correlated variables is unstable and can change with small changes in the sample. For this reason the set retained by LASSO differs from the set of `Recommended` (in the example bio_4, bio_6, bio_10, bio_12, bio_13, bio_16 were retained by LASSO but removed by stepwise VIF), and this is expected: LASSO looks at discriminatory power, stepwise VIF only at collinearity. The coefficients refer to the pooled PA sets; they are not the coefficients of the final model. In the example run the minimum deviance coincided with the smallest λ of the grid; this indicates that the grid may need to be extended to smaller λ values (see Q8).

4.9.5 Ridge (L2) — coefficient shrinkage

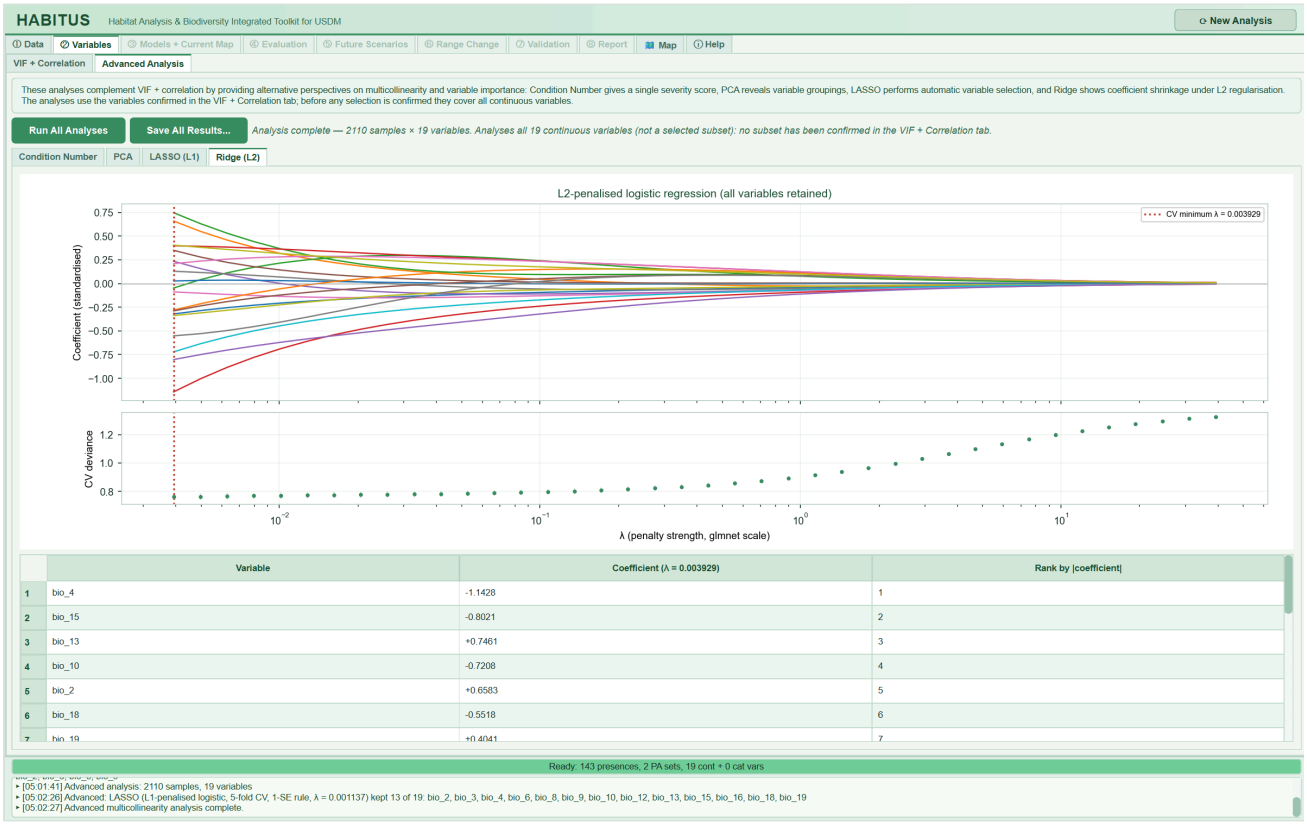


Figure 4.5. Advanced Analysis > Ridge (L2): paths of the standardized coefficients against λ and cross-validation deviance; the red dotted line marks the λ of minimum deviance; ranking by absolute coefficient at the bottom.

- **What it does.** The same set-up as LASSO, with an L2 penalty (`tabs/tab_vif_advanced.py:715-748`). The grid starts at 100 × λ_{max} and descends four orders of magnitude. λ is always chosen by the minimum cross-validation deviance; it has no setting, only `CV folds` is shared with LASSO. Ridge sets no variable to zero; the table ranks the variables by absolute standardized coefficient and gives the rank in the `Rank by |coefficient|` column. No variable is "recommended".
- **When and how to set it.** To see how coefficients are shared when correlated variables are kept together. Ridge coefficients are more stable than those of LASSO.
- **Effect on results.** Information only. In the example run λ = 0.003929; the largest absolute coefficients are bio_4 (−1.1428), bio_15 (−0.8021), bio_13 (+0.7461), bio_10 (−0.7208), bio_2 (+0.6583) (Figure 4.5).

- **Caution.** The `Recompute` button recomputes LASSO only; after changing `CV folds`, `Run All Analyses` must be pressed for Ridge to be refreshed as well. In the example run the minimum deviance of Ridge also coincided with the smallest λ of the grid.

4.9.6 Export of the advanced analysis

`Save All Results...` creates the `advanced_analysis/` subfolder in the output folder and writes the following (`tabs/tab_vif_advanced.py:760-891`): `condition_number.txt` (κ, numbers of variables and rows, first 10 singular values), `pca_variance.png`, `pca_loadings.csv` (first k components), `pca_variance.csv` (variance and cumulative variance percentages of all components), `lasso_coefficient_paths.png`, `lasso_coefficients.csv`, `lasso_path.csv` (λ grid, CV deviance and its standard error, all coefficients), `lasso_info.txt` (number of records, number of folds, the two λ values, the λ used, selected variables), `ridge_coefficient_paths.png`, `ridge_coefficients.csv`, `ridge_path.csv`. If no output folder is defined, the warning "Set the output directory in the Data tab first." appears.

4.10 Outputs and what passes to the next step

- `state["var_names"]`: the confirmed continuous variables, in natural name order. Used by ③ Models, ⑤ Future Scenarios (projection and novelty maps) and ⑧ Report.
- `state["cat_selected"]` / `state["cat_excluded"]`: the categorical selection; `None` and an empty list if there is no categorical raster.
- The files in `figures/` and (if requested) `advanced_analysis/` (§4.8, §4.9.6).
- The thresholds and method of the tab are written to the report and to the "Multicollinearity" element of ODMAP.
- The records-per-predictor warning (if any) is added to the `sample_warnings` list and passed to the report.

4.11 Expert questions

Q1. Does the stepwise VIF selection give the same result as usdm::vifstep? Yes, it gives the same result on the same input. Both methods recompute the VIF of the remaining set at each step, remove the largest if it exceeds the threshold, and stop when the largest VIF is equal to or below the threshold (`sdm_core.py:427-434`; `th = 10` in usdm). In this book they were compared on a 5000-cell WorldClim matrix: the removal order is identical for 11 variables, and the VIFs of the remaining eight variables agree to the fourteenth decimal (§4.6.2). As in usdm, VIF is computed from the R^2 of a linear regression with an intercept (`sdm_core.py:275-295`).

Q2. Can capping VIF at 999 change the removal order? It can. Every variable with $R^2 > 0.9999$ gets a VIF of 999 (`sdm_core.py:292-293`); if several variables are in this state at the same step, `idxmax()` removes the first in natural order (`sdm_core.py:430`). Since usdm uses the true VIF, it removes the one that is truly the largest at that step. In the example run, bio_4 and bio_5 were removed with 999 in the first two steps. In the validation matrix this did not change the result; but in data with capped ties the final set may differ from that of usdm. The difference concerns only which of the extremely high-VIF variables goes first; if exactly the same result is required, the most obviously derived variables (such as bio_7) can be removed manually beforehand.

Q3. Why are VIF and correlation computed on the presence + pseudo-absence pool rather than on raster cells? Because the model is trained on these points; collinearity matters in the data the model sees (`tabs/tab_vif.py:349-353`). As a consequence, the results depend on the extent of the accessible area and on the number of PAs (§4.3). In the `VIF + Correlation` pool each presence is repeated as many times as there are PA sets, whereas `Advanced Analysis` drops duplicates (`tabs/tab_vif_advanced.py:356-363`); the samples

of the two sub-tabs are therefore not exactly the same. A user who wants a correlation over the whole area can apply usdm::vifstep in R directly to the rasters.

Q4. Why does `Recommended` not take the univariate AUC into account? The explanation in the code: univariate discriminatory power promotes a variable that looks good in this sample thanks to its correlation with the true driver; such a variable may fail when transferred to another region or climate (`sdm_core.py:323-339`). For this reason both `Priority_Score` and `Recommended` are based only on collinearity, and the AUC badge is deliberately neutral in colour (`tabs/tab_vif.py:471-499`). Which member of a pair to drop should be decided by the ecology of the species (Dormann et al. 2013).

Q5. Why does the correlation step remove the member of a pair with the higher VIF; how does it differ from usdm::vifcor? HABITUS first applies the VIF step, then, in the remaining set, removes from the strongest pair with $|r| \geq$ threshold the member with the larger VIF in the current set (`sdm_core.py:435-445`). usdm::vifcor instead first removes variables pair by pair using the correlation threshold (from each pair, the one more strongly related to the other variables), and then looks at VIF. The final sets of the two methods may differ. With the default thresholds (VIF 10, $|r|$ 0.80), a highly correlated pair rarely remains after the VIF step; in the example run none remained.

Q6. I changed the thresholds but `Recommended` gives the same suggestion. Why? The suggestion is computed when `variable_priority_ranking` last ran. When the tab first opens, the pre-computation of ① Data is shown (correlation 0.70, Pearson, VIF 10; `sdm_core.py:3225-3227`). Changing the correlation threshold refreshes only the pairs table and the heatmap (`tabs/tab_vif.py:880-884`); the method and the VIF threshold refresh nothing on their own. For a new suggestion, press `Run / Refresh Analysis` and then `Recommended`.

Q7. Do LASSO and Ridge give the same results as glmnet in R? They use the same definition (Friedman et al. 2010): binomial family, λ on the glmnet scale, equal class weights, standardized predictors, stratified k-fold cross-validation, `lambda.min` and `lambda.1se` (`tabs/tab_vif_advanced.py:63-128`). The program's test suite compares the coefficients at two grid points on simulated data with R glmnet: within an absolute difference of 2×10^{-4} for Ridge and 3×10^{-3} for LASSO, and the variables set to zero by LASSO are the same (`tests/test_202_small_samples_lasso.py:81-98`). The solver is liblinear and the intercept is estimated with a very small penalty (`intercept_scaling=100`); this is the source of small differences from glmnet. The λ grid is not the same as that of glmnet (100 values, `lambda.min.ratio`): HABITUS uses 40 values and descends three orders of magnitude for LASSO and four for Ridge.

Q8. What happens if the minimum cross-validation deviance falls at the end of the grid? The program does not report this separately; `lambda_min` becomes the smallest value of the grid (`tabs/tab_vif_advanced.py:120`). In the example run this happened for LASSO ($\lambda_{\min} = 0.0003929 = \lambda_{\max}/1000$) and for Ridge (0.003929) (in Figures 4.4 and 4.5 the red line is at the left end). In this case the true minimum may lie at a smaller λ; the 1-SE rule, by contrast, is more reliable because the deviance curve is flat. Since these analyses are for information only, they do not affect model results; if a detailed choice of penalty is required, `advanced_analysis/lasso_path.csv` can be inspected or glmnet can be run with a wider grid.

Q9. Why is the condition number so large, and how does it relate to VIF? κ is the ratio of the largest to the smallest singular value of the standardized matrix (`tabs/tab_vif_advanced.py:421-431`). As the smallest singular value approaches zero, κ grows; this means that at least one variable is approaching a linear combination of the others. VIF shows the same situation per variable. Because the 19 bioclimatic variables contain exact derivations such as bio_7 = bio_5 − bio_6, κ is in the millions (22,681,501.75 in the example run). A drop in κ when recomputed with the selected set shows that the selection has worked.

Q10. Can a model be built with a single variable? No. With a single variable the ranking table is built (VIF = 1; `sdm_core.py:356-368`), but `Confirm` requires at least two continuous variables (`tabs/tab_vif.py:1164-`

`1167`); `Run / Refresh Analysis` also asks for layers to be added in ① Data if only one variable is loaded (`tabs/tab_vif.py:367-372`).

Q11. Why are categorical variables not included in the collinearity check? VIF and Pearson/Spearman correlation assume ordered numeric variables; the numeric values of categorical codes are meaningless. Categorical rasters are therefore listed separately and sparse classes (fewer than 3 presences) are flagged (`tabs/tab_vif.py:583-664`). The program does not measure the relationship between categorical and continuous variables (for example, a land-cover class coinciding with elevation); this is a point the user must check. `Advanced Analysis` does not use categorical rasters; it works only with continuous variables (`tabs/tab_vif_advanced.py:343-354`).

Q12. Is the selection in this tab reproducible? VIF, correlation, PCA and the condition number involve no randomness; the same PA sets give the same result. The PA sets depend on the seed in ① Data. The cross-validation folds of LASSO and Ridge are shuffled with the run seed (`formatter.random_seed`) (`tabs/tab_vif_advanced.py:600-604`, `108`). The selected variables, thresholds and method are written to the report, to ODMAP and to `figures/variable_priority_ranking.csv`; the removals made with `Recommended` are in the session log in order.

4.12 Common mistakes and pitfalls

- **Changing a threshold without pressing `Run / Refresh Analysis`.** The suggestion stays with the old thresholds (§4.4.1). First set the thresholds, then `Run / Refresh Analysis`, then `Recommended`.
- **Treating `Recommended` as the final decision.** The tooltip explicitly says "A starting point". Compare the recommended set with the ecology of the species; for example, in the example run the recommended set does not contain annual mean temperature (bio_1); bio_11 (mean temperature of the coldest quarter) has been retained instead.
- **Removing a recommended variable because of its full-set VIF badge.** The badge shows the VIF in the presence of all variables; the VIF within the recommended set is in the `VIF_in_selection` column of `variable_priority_ranking.csv` (in the example bio_11: ∞ in the full set, 2.85 in the recommended set).
- **Reading a red AUC in the pairs table as "drop this".** The AUC colouring only compares two AUCs; a low univariate AUC does not make a variable redundant (§4.5.1).
- **Changing the selection without confirming.** ③ Models uses the last confirmed set. After models have been run, re-confirmation deletes everything ("Re-confirm Variable Selection").
- **Assuming the suggestions of `Advanced Analysis` have been applied to the list.** PCA and LASSO provide information only; the ticks in the variable list do not change.
- **Ignoring the records-per-predictor warning.** If the "Below about 10 per predictor …" warning appears, select fewer variables or use simpler algorithms in ③ Models (GLM, MaxEnt with linear/quadratic features).

4.13 References

Barbet-Massin, M., Jiguet, F., Albert, C.H., Thuiller, W., 2012. Selecting pseudo-absences for species distribution models: how, where and how many? Methods in Ecology and Evolution 3, 327–338. https://doi.org/10.1111/j.2041-210X.2011.00172.x

Belsley, D.A., Kuh, E., Welsch, R.E., 1980. Regression Diagnostics: Identifying Influential Data and Sources of Collinearity. Wiley, New York. https://doi.org/10.1002/0471725153

Dormann, C.F., Elith, J., Bacher, S., Buchmann, C., Carl, G., Carré, G., et al., 2013. Collinearity: a review of methods to deal with it and a simulation study evaluating their performance. Ecography 36, 27–46. https://doi.org/10.1111/j.1600-0587.2012.07348.x

Friedman, J., Hastie, T., Tibshirani, R., 2010. Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software 33(1), 1–22. https://doi.org/10.18637/jss.v033.i01

Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning, 2nd ed. Springer, New York. https://doi.org/10.1007/978-0-387-84858-7

Naimi, B., Hamm, N.A.S., Groen, T.A., Skidmore, A.K., Toxopeus, A.G., 2014. Where is positional uncertainty a problem for species distribution modelling? Ecography 37, 191–203. https://doi.org/10.1111/j.1600-0587.2013.00205.x

O'Brien, R.M., 2007. A caution regarding rules of thumb for variance inflation factors. Quality & Quantity 41, 673–690. https://doi.org/10.1007/s11135-006-9018-6

Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society Series B 58, 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

5. Models I: algorithms and their settings (③ Models → Algorithm Settings)

Tab: ③ Models + Current Map → Algorithm Settings · Code: `tabs/tab_models.py` (UI), `sdm_core.py` `build_model`, `SDMModeler` (computation), `sdm_niche.py` (BIOCLIM, Domain, ESM), `sdm_tuning.py` (tunable parameters)

5.1 What this tab does

The `Algorithm Settings` sub-tab answers the "with which method" question of modelling. Here you tick which of the 18 algorithms will be run and enter the settings (hyperparameters) of each. A hyperparameter is a setting that the model does not learn from the data but that the user fixes in advance: the number of trees, a penalty coefficient, the number of layers and so on.

Its place in the workflow is as follows:

- **Input:** the pseudo-absence datasets produced by the ① Data tab (presence + pseudo-absence rows for each replicate), the MaxEnt background, the optional presence-only background, and the predictors (continuous and categorical) selected in the ② Variables tab. Until these two steps are complete, the run button gives the message "Complete ① Data and ② Variables steps first." (`tabs/tab_models.py:1394-1395`).
- **Processing:** when the `▶ Run Models + Generate Current Distribution Map` button is pressed, every ticked algorithm is trained on every pseudo-absence set, in the cross-validation folds and with the full data (`sdm_core.py:5352` `SDMModeler.run`). The settings on this page are collected by `_collect_algo_opts` and passed to `SDMModeler` as `algo_options` (`tabs/tab_models.py:1142-1209`, `1569-1573`).
- **Output:** the trained models, evaluation scores, variable importances and the current distribution map. These are shown in the `Evaluation Results` and `Current Distribution Map` sub-tabs and in the ④ Evaluation tab. All settings used are written to the `settings.models.algorithm_options` field of the `habitus_run_config.json` file.

The cross-validation design, threshold and ensemble settings are in the lower half of the same page; they are described in Chapter 6. This chapter covers only the choice of algorithms and the algorithm settings.

5.2 Screen layout

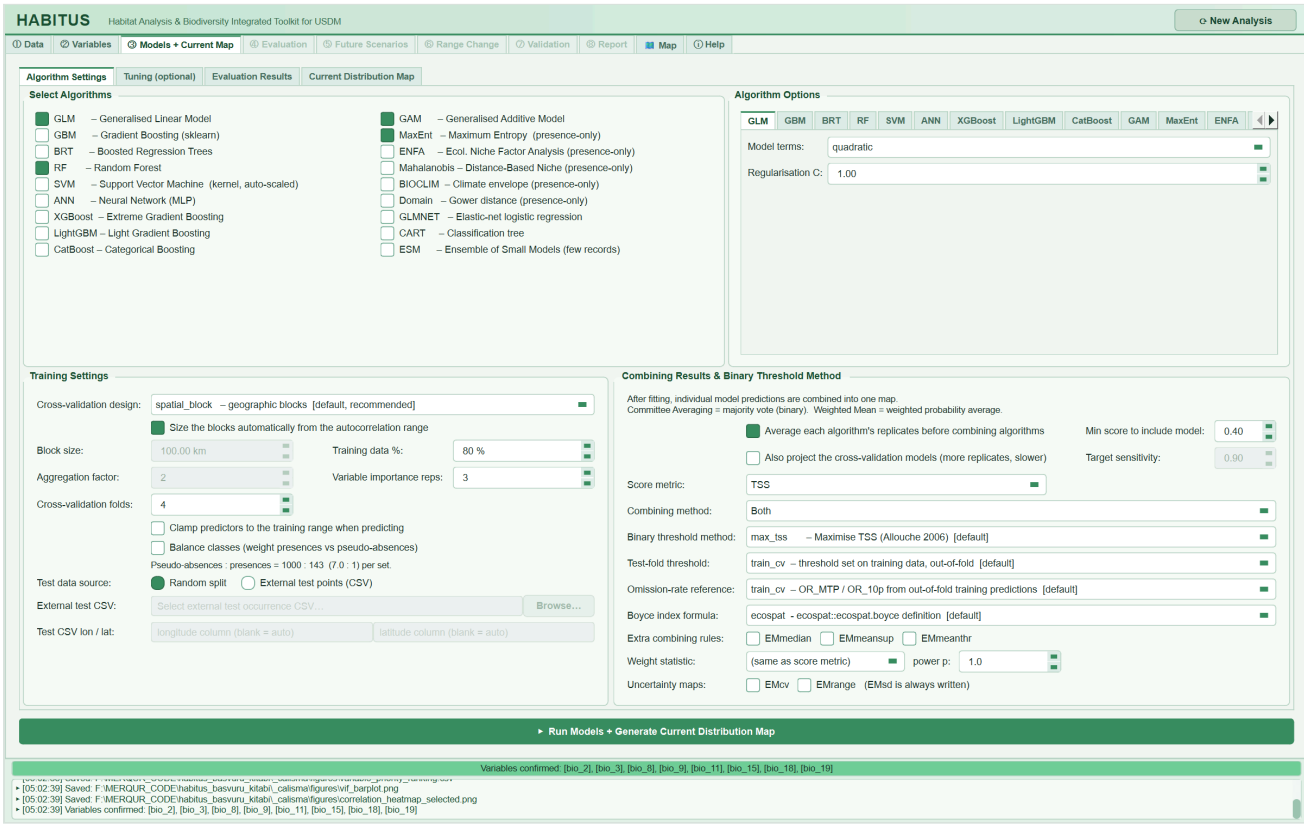


Figure 5.1. Algorithm Settings sub-tab: the Select Algorithms box on the left (GLM, RF, GAM and MaxEnt ticked in the example run), the Algorithm Options panel on the right (GLM page open); below, the Training Settings and Combining Results & Binary Threshold Method groups, and at the bottom the run button.

The page consists of four areas:

1. **`Select Algorithms`** (top left): 18 check boxes, nine in each of two columns (`tabs/tab_models.py:125-150`). See §5.3.
2. **`Algorithm Options`** (top right): one tab per algorithm. Because the tab bar cannot fit all 18 tabs, it is scrolled with the arrow buttons on the right (`tabs/tab_models.py:173-194`). Pages with six or more rows (BRT, XGBoost, LightGBM, CatBoost, MaxEnt, ESM) are laid out in two columns (`tabs/tab_models.py:197-201`). Each tab is described in its own subsection in §5.6–§5.23.
3. **`Training Settings`** and **`Combining Results & Binary Threshold Method`** (lower half): Chapter 6.
4. **Run button:** ▶ `Run Models + Generate Current Distribution Map`.

The data of the example run are *Abies nordmanniana* (626 records; 143 presences after thinning; WorldClim 2.1 bioclimatic variables, 2.5′; 8 variables selected in ② Variables: bio_2, bio_3, bio_8, bio_9, bio_11, bio_15, bio_18, bio_19; 2 pseudo-absence sets × 1000 points; MaxEnt background 10,000 points; seed 42). The numbers in this chapter are taken from the `habitus_run_config.json`, `evaluation_scores.csv` and session log of this run.

5.3 Algorithm selection (Select Algorithms)

Name in the UI	What it does	Default	Range / options	Code
`GLM – Generalised Linear Model`	Logistic regression	on	on / off	`tabs/tab_models.py:129`
`RF – Random Forest`	Random forest	on	on / off	`tabs/tab_models.py:132`
The other 16 boxes (`GBM`, `BRT`, `SVM`, `ANN`, `XGBoost`, `LightGBM`, `CatBoost`, `GAM`, `MaxEnt`, `ENFA`, `Mahalanobis`, `BIOCLIM`, `Domain`, `GLMNET`, `CART`, `ESM`)	Runs the corresponding algorithm	off	on / off	`tabs/tab_models.py:130-146`

When the program starts, only GLM and RF are ticked. If the package required by an algorithm is not installed, its box is unticked and disabled, `[not installed]` is appended to its name and the installation command is shown in the tooltip (e.g. "Not installed — pip install pygam"; `tabs/tab_models.py:152-168`). Algorithms that need a package: XGBoost (`xgboost`), LightGBM (`lightgbm`), CatBoost (`catboost`), GAM (`pygam`), MaxEnt (`elapid`); the other 13 algorithms run with scikit-learn or numpy (`sdm_core.py:127-146`). If no box is ticked, the run stops with the message "Select at least one algorithm." (`tabs/tab_models.py:1397-1400`).

> **Note:** The order in which algorithms are ticked or predictors are added does not change the result. In every run the columns are ordered first continuous, then categorical, and each group in natural name order (`sdm_core.py:3892-3899`).

5.4 Rules common to all algorithms

5.4.1 Which rows are used for training: presence-absence and presence-only

In HABITUS the algorithms fall into three groups according to the rows they use in training:

- **Presence–pseudo-absence (classifier) models:** GLM, GLMNET, GAM, GBM, BRT, RF, CART, SVM, ANN, XGBoost, LightGBM, CatBoost and the GLM/RF-based ESM. They use all rows of the pseudo-absence set (presence = 1, pseudo-absence = 0) (`sdm_core.py:4480-4498`).
- **Presence + background:** MaxEnt. It uses the presences in the training fold and the MaxEnt background sampled separately in ① Data; pseudo-absences do not enter MaxEnt training (`sdm_core.py:4595-4603`, `tabs/tab_models.py:1580`). ENFA also uses the presence-only background, if available, to estimate the global environmental distribution (`sdm_core.py:4079-4092`).
- **Presence only:** Mahalanobis, BIOCLIM and Domain are fitted from presence rows only (`sdm_core.py:4227-4228`, `4281-4282`).

Evaluation, however, is the same whatever the group: AUC, TSS, Boyce and the others are computed from the presences and pseudo-absences in the test fold. For presence-only methods, if no presence is left in the test fold, about one fifth of the presences in the training fold are moved to the test set (`sdm_core.py:5455-5468`; for the full model `5686-5696`).

5.4.2 Scaling of the input

For scale-sensitive methods, standardisation (mean 0, standard deviation 1) is done inside the program, learned from the training rows only; the user does not need to scale beforehand:

- GLM and GLMNET: the terms (linear, quadratic, interaction) are generated first, then these terms are standardised (`sdm_core.py:3488-3505`).
- SVM and ANN: the predictors are standardised (`sdm_core.py:3562-3581`).
- Tree methods (GBM, BRT, RF, CART, XGBoost, LightGBM, CatBoost) are unaffected by scale; no scaling is done.
- MaxEnt features are generated by elapid's transformer. GAM, ENFA, Mahalanobis, BIOCLIM and Domain work with raw values; ENFA and Mahalanobis remove scale themselves through the covariance matrix, BIOCLIM through percentiles, and Domain through differences divided by the presence range.

5.4.3 Categorical variables

Categorical rasters (land cover, aspect class and the like) reach the model as integer class codes; each algorithm uses them as follows:

Algorithm	Handling of the categorical variable	Code
GLM, GLMNET, SVM, ANN	One 0/1 column (one-hot) for each class seen in training; the codes are not squared, scaled or multiplied. A class not seen in training gets 0 in all columns of that variable	`sdm_core.py:3364-3443`
GAM	Factor term `f()`, one coefficient per class; a single-class variable is skipped; the effect of an unseen class is taken as 0	`sdm_core.py:3320-3361`, `4525-4549`
MaxEnt	elapid's categorical feature class (one-hot, penalised with `beta_categorical`); unseen class 0	`sdm_core.py:4606-4612`, `4712-4718`, `4760-4770`
GBM, BRT, RF, CART, XGBoost, LightGBM, CatBoost	The integer code is split like a numeric column	`sdm_core.py:3476-3479`
ENFA, Mahalanobis, BIOCLIM, Domain	Categorical variables are left out; "… is an envelope/distance model for continuous predictors: the categorical variable(s) … are excluded from it." is written to the log	`sdm_core.py:4010-4036`
ESM	Categorical variables do not enter the pairs	`sdm_core.py:4389-4393`

> **Caution:** In tree methods the class code is split like an ordered number: codes 3 and 4 are treated as "close" and 3 and 12 as "far". If the order of the codes carries no ecological meaning, the tree can separate the classes only with several splits. CatBoost's own categorical handling is not used in HABITUS.

5.4.4 Reproducibility and the random seed

Every learner that involves randomness (RF, GBM, BRT, CART, SVM, ANN, GLMNET, XGBoost, LightGBM, CatBoost, MaxEnt, ESM folds) is seeded with the `Random seed` in ① Data (default 42) (`sdm_core.py:115-124`, `3474-3475`, `4643`). Two runs with the same data, the same settings and the same seed produce the same models. Permutation importance is also computed with a generator specific to the (PA set, fold, algorithm) triple but derived from the seed (`sdm_core.py:4847-4853`). The seed used is in the `random_seed` field of `habitus_run_config.json`.

5.4.5 Class balance and clamping (two settings from Training Settings)

`Balance classes (weight presences vs pseudo-absences)` and `Clamp predictors to the training range when predicting` are in the Training Settings group and are described in Chapter 6; but because they behave differently depending on the algorithm, they are summarised here. When balancing is on (default off), each algorithm takes its class weight as follows (`sdm_core.py:3670-3716`):

Algorithm	Form of balancing
GLM, RF, SVM, GLMNET, CART, LightGBM	`class_weight='balanced'`
CatBoost	`auto_class_weights='Balanced'`
GBM, BRT	balanced sample weights (`sample_weight`)
XGBoost	`scale_pos_weight` = number of pseudo-absences / number of presences
ANN	balanced sample weights (if the installed scikit-learn supports this)
ESM	only in the GLM- or RF-based small models
GAM, MaxEnt, ENFA, Mahalanobis, BIOCLIM, Domain	not supported; the setting has no effect for these algorithms

When clamping is on, before every prediction the predictors are clamped to the min–max range of all pseudo-absence sets; in addition, elapid's `clamp=True` setting is passed explicitly to MaxEnt (`sdm_core.py:4650-4655`, `4929-4962`).

5.4.6 Relation to the tuning sub-tab

The `Tuning (optional)` sub-tab tries the parameters of some algorithms with a cross-validated grid search and writes the selected value to this page with `Apply selected setting`. The parameters that can be written to this page (`tabs/tab_models.py:1212-1229`, `1231-1259`) and the default grids of the tuning tab (`sdm_tuning.py`) are:

Algorithm	Parameters written to this page	Default grid (source)
MaxEnt	`Feature types`, `Regularisation multiplier (β / RM)`	rm 0.5–4.0 step 0.5; fc L, LQ, H, LQH, LQHP (`sdm_tuning.py:502-552`)
GBM	`n_estimators`, `learning_rate`, `subsample`, `min_samples_leaf`, `max_depth`	flexsdm `tune_gbm`: trees 20–500, learning rate 0.01/0.1/0.3, leaf 1–10, 20, 30 (`sdm_tuning.py:626-630`)
RF	`n_estimators`, `min_samples_leaf`, `max_features`	flexsdm `tune_raf`: mtry 2…number of predictors, ntree 200–1200 (`sdm_tuning.py:633-634`, `787-793`)
SVM	`C`, `gamma`	flexsdm `tune_svm`: C 1–16, sigma 0.001–0.2 → gamma the same value (`sdm_tuning.py:636-637`, `654-675`)
ANN	`alpha (L2)`, `Hidden layers`	flexsdm `tune_net`: single layer 2–8 neurons, decay 0.01–10 → alpha = 2 × decay (`sdm_tuning.py:639-640`, `678-689`)
XGBoost	`n_estimators`, `max_depth`, `learning_rate`	HABITUS's grid: 100/300/500; 3/6/9; 0.01/0.05/0.1 (`sdm_tuning.py:779-781`)
LightGBM	`n_estimators`, `num_leaves`, `learning_rate`	HABITUS's grid: 100/300/500; 15/31/63; 0.01/0.05/0.1 (`sdm_tuning.py:782-784`)
GLMNET	`C (inverse penalty)`, `l1_ratio (alpha)`	C 0.01/0.1/1/10; l1_ratio 0–1 step 0.25 (`sdm_tuning.py:837-838`)
CART	`max_depth`, `min_samples_leaf`, `ccp_alpha`	2/3/5/8; 1/5/10/20; 0/0.002/0.01 (`sdm_tuning.py:839-841`)
GAM	`n_splines`, `lam (smoothing)`	5/10/15/20; 0.1/0.6/2/10 (`sdm_tuning.py:842-843`)

GLM, BRT, CatBoost, ENFA, Mahalanobis, BIOCLIM, Domain and ESM are not in the tuning tab; their values are entered only on this page. The tuning search itself is described in Chapter 6.

5.5 Guide to choosing algorithms

The table below is a starting guide derived from the structure seen in the code and from the example run; it does not say that one method will be better than another on every dataset. The numbers in the "Speed" column are the time taken by the 4 cross-validation folds of one pseudo-absence set in the example run (including inner folds; from the log).

Algorithm	Data type	Speed	Interpretability	Tendency to overfit (with defaults)	When
GLM	presence–pseudo-absence	very fast (0.7 s)	high (coefficients)	low–medium (L2-penalised, quadratic)	baseline model in every study
GLMNET	presence–pseudo-absence	fast	high	low (elastic net)	many predictors, few records

Algorithm	Data type	Speed	Interpretability	Tendency to overfit (with defaults)	When
GAM	presence–pseudo-absence	medium (10 s)	high (response curves)	medium	non-linear, smooth responses
MaxEnt	presence + background	slow (165 s)	medium–high	depends on β and feature classes	presence-only data, comparability
RF	presence–pseudo-absence	medium (21 s)	low	high training fit (training AUC 1.000)	many records, complex interactions
GBM	presence–pseudo-absence	medium	low–medium	low (stumps)	flexible additive model
BRT	presence–pseudo-absence	medium–slow	low–medium	medium	boosting along the lines of Elith et al. (2008)
XGBoost, LightGBM, CatBoost	presence–pseudo-absence	fast–medium	low	high (deep trees)	many records (hundreds), comparison
CART	presence–pseudo-absence	very fast	very high (single tree)	unstable, stepped surface	explanatory purposes
SVM	presence–pseudo-absence	slows as n grows	low	depends on C and gamma	comparison
ANN	presence–pseudo-absence	medium	very low	high (100+50 neurons)	many records
ENFA	presence + background	very fast	medium (axes)	low	presence-only, coarse niche
Mahalanobis	presence only	very fast	medium	low	elliptical niche assumption
BIOCLIM	presence only	very fast	very high	low; sensitive to extreme records	climate envelope, comparison
Domain	presence only	fast (grows with number of presences)	high	low; sensitive to single records	similarity-based comparison
ESM	presence–pseudo-absence (pairs)	grows with number of pairs	medium	low (bivariate models)	very few records (about < 30)

In the example run (143 presences, 8 variables) the cross-validation means of the four algorithms were: GLM AUC 0.889, TSS 0.687, Boyce 0.607; MaxEnt 0.895, 0.672, 0.605; RF 0.892, 0.566, 0.677; GAM 0.872, 0.611, 0.653. The difference between training and test AUC (dAUC) is 0.031 for GLM, 0.055 for MaxEnt, 0.087 for GAM and 0.108 for RF; RF fits the training data most tightly.

> **Tip:** Instead of looking for a single "best" algorithm, it is common practice to run several algorithms from different families (a regression, a tree ensemble, MaxEnt) and combine them in an ensemble map; combining is

in Chapter 6.

5.6 GLM: generalised linear model

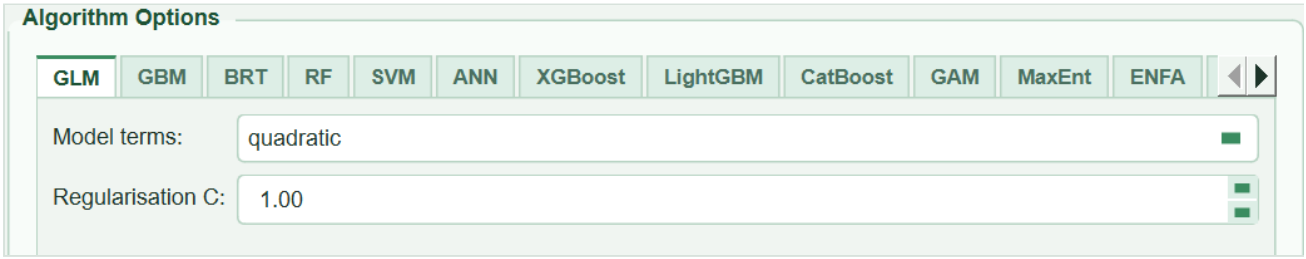


Figure 5.2. GLM tab: Model terms (quadratic) and Regularisation C (1.00).

Family and implementation. Binary (logistic) regression: the logit of the probability of presence is a linear combination of the predictors. It is fitted with scikit-learn `LogisticRegression` (solver `lbfgs`, at most 2000 iterations) inside a pipeline that first generates terms and then standardises them (`sdm_core.py:3481-3492`). Data type presence–pseudo-absence. The penalty is an L2 (ridge) penalty applied to the coefficients of the standardised terms.

Name in the UI	What it does	Default	Range / options	Code
`Model terms:`	Which terms enter the model	`quadratic`	`linear`, `quadratic`, `interactions`	`tabs/tab_models.py:721-735`; `sdm_core.py:3300-3317`
`Regularisation C:`	Inverse penalty strength (L2)	1.00	0.001 – 1 000 000	`tabs/tab_models.py:736-745`; `sdm_core.py:3482`

5.6.1 `Model terms`

- **What it does.** `linear` generates only the x term for each predictor, `quadratic` the x and x^2 terms, and `interactions` in addition all pairwise products ($x_i x_j$) (`sdm_core.py:3300-3317`). Compared with MaxEnt feature classes they are L, LQ and LQP respectively. With p predictors the number of terms is p, $2p$ and $p + p(p+1)/2$; with 8 predictors 8, 16 and 44.
- **When and how to set it.** `quadratic` allows a unimodal (optimum) response along a variable; it is the classical assumption of the ecological niche and suits most studies. `linear` gives only monotonically increasing or decreasing responses; it is chosen with very few records or to keep interpretation simple. `interactions` needs at least a few hundred presence records; by the ten-events-per-variable rule in the tooltip, about 440 records with 8 predictors.
- **Effect on results.** As terms increase, training fit increases, response curves bend and the risk of going beyond the calibration range (extrapolation) in projections grows; a squared term gives a curve that falls or rises steeply outside the range.
- **Caution.** Terms are not applied to the codes of categorical variables; they are applied only to continuous variables (`sdm_core.py:3364-3378`).

5.6.2 `Regularisation C`

- **What it does.** The objective function of scikit-learn is $C \sum_i \ell_i + \frac{1}{2}\|\beta\|^2$: as C decreases the penalty grows and the coefficients are pulled towards zero. Because the penalty is applied to standardised terms, it affects all terms equally.
- **When and how to set it.** If the number of terms is large relative to the number of records (`interactions`, many predictors), C is reduced (0.1 or 0.01). R's `glm()` is unpenalised (as are the GLMs of biomod2, sdm and flexsdm); to reproduce their results, enter C = 1e6 (tooltip, `tabs/tab_models.py:741-743`).
- **Effect on results.** Because the penalty is defined relative to a sum of losses, its relative effect decreases as the number of rows increases: in the example run a training fold has about 860 rows (three quarters of 1143 rows) and C = 1 is a light penalty. The tooltip reports that in a *Pinus brutia* example the default penalty raised the cross-validation AUC from 0.804 (unpenalised) to 0.824.
- **Caution.** GLM is not in the tuning tab; if a search over C is wanted, GLMNET (l1_ratio = 0), which carries the same penalty family, can be used.

Equivalence with R. With `Model terms` = `linear` and C = 1e6 the model solves the same objective function as `glm(y ~ x1 + x2 + …, family = binomial)` in R; for `quadratic` the corresponding form is `y ~ x + I(x^2)`. biomod2's GLM eliminates terms by AIC-based stepwise selection; HABITUS has no stepwise selection, it has a penalty instead.

5.7 GBM: gradient boosting (scikit-learn)

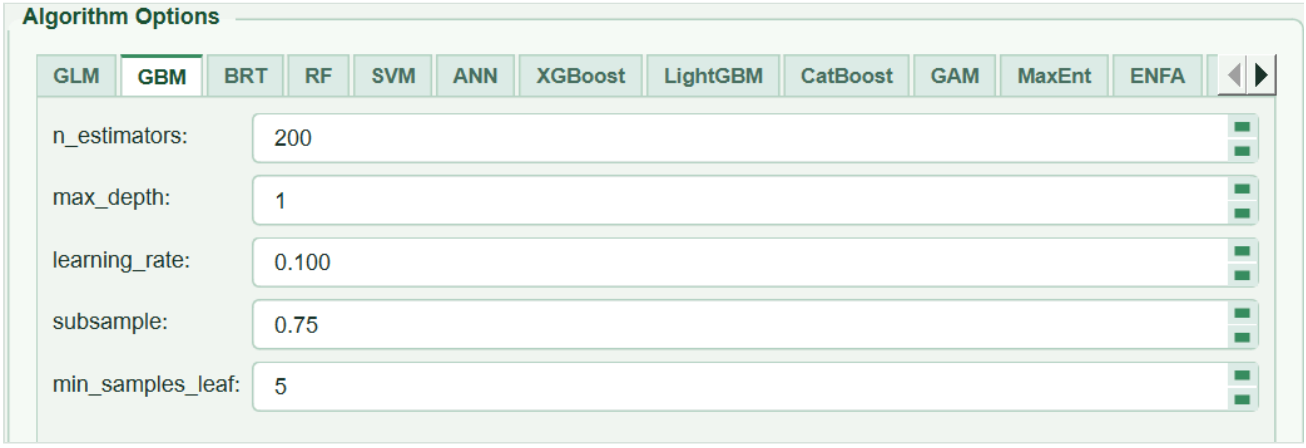


Figure 5.3. GBM tab: n_estimators 200, max_depth 1, learning_rate 0.100, subsample 0.75, min_samples_leaf 5.

Family and implementation. Gradient-boosted decision trees (boosting): each new tree corrects the error (the gradient of the log-loss) of the sum of the previous trees. scikit-learn `GradientBoostingClassifier` (`sdm_core.py:3516-3529`). Presence–pseudo-absence. No scaling needed. GBM and BRT use the same implementation; only their defaults and ranges differ.

Name in the UI	What it does	Default	Range / options	Code
`n_estimators:`	Number of trees	200	10 – 5000 (step 100)	`tabs/tab_models.py:750`; `sdm_core.py:3524`
`max_depth:`	Tree depth (gbm: `interaction.depth`)	1	1 – 10	`tabs/tab_models.py:751-752`; `sdm_core.py:3525`

Name in the UI	What it does	Default	Range / options	Code
`learning_rate:`	Coefficient that shrinks the contribution of each tree (gbm: `shrinkage`)	0.100	0.001 – 1.0	`tabs/tab_models.py:753`; `sdm_core.py:3526`
`subsample:`	Fraction of rows drawn for each tree (gbm: `bag.fraction`)	0.75	0.1 – 1.0 (step 0.05)	`tabs/tab_models.py:756-758`; `sdm_core.py:3527`
`min_samples_leaf:`	Minimum rows in a leaf (gbm: `n.minobsinnode`)	5	1 – 200	`tabs/tab_models.py:754-755`; `sdm_core.py:3528`

5.7.1 `n_estimators` and `learning_rate`

- **What it does.** The model has the form $F = \sum_{m=1}^{M} \nu\, h_m(x)$; M is the number of trees and ν the learning rate. Together they determine the model's total "amount of learning".
- **When and how to set it.** If the learning rate is reduced, the number of trees must be increased proportionally for the same fit (0.1 → 0.01 means about 10 times as many trees). Elith et al. (2008) recommend a learning rate that reaches at least 1000 trees. The best pair can be searched with the flexsdm grid in the `Tuning (optional)` tab.
- **Effect on results.** Many trees + a high rate lead to overfitting, few trees + a low rate to underfitting. Run time increases linearly with the number of trees.
- **Caution.** The number of trees is not selected automatically by cross-validation (there is no early stopping as in dismo's `gbm.step`); the number entered is used as is.

5.7.2 `max_depth`

- **What it does.** The depth of each tree. 1 means a single-split tree (a "stump") and the model becomes an additive function of the predictors with no interactions. 2 and above allow variable interactions.
- **When and how to set it.** The default 1 is close to the `interaction.depth` = 1 default of flexsdm `fit_gbm` (`sdm_core.py:3517-3522`). If interactions are thought to be important and the number of records is high, 2–3 are tried.
- **Effect on results.** According to the tooltip, stumps prevent the surface from collapsing to zero in most cells and give a higher Boyce at the same AUC (`tabs/tab_models.py:752`).
- **Caution.** gbm's `interaction.depth` and scikit-learn's `max_depth` define the same tree at 1; at larger values the two packages grow trees differently, so the numbers do not map one to one.

5.7.3 `subsample`

- **What it does.** Each tree is built with a random subset of the training rows of this fraction (stochastic gradient boosting).
- **When and how to set it.** The range 0.5–0.75 is common; 1.0 uses all rows and removes the randomness.
- **Effect on results.** Values below 1 reduce variance and overfitting; the result becomes dependent on the seed (it remains reproducible because the seed is fixed).
- **Caution.** Very small values can leave some trees almost without presences in data with few presences.

5.7.4 `min_samples_leaf`

- **What it does.** The minimum number of rows a leaf may hold; splits that would create smaller leaves are not made.
- **When and how to set it.** The default 5 corresponds to gbm's `n.minobsinnode` default. With noisy data or few records it can be raised to 10–20.
- **Effect on results.** As it grows, the trees become coarser, the surface smoother and overfitting smaller.
- **Caution.** With `max_depth` = 1 each tree has only one split, so its effect is limited.

5.8 BRT: boosted regression trees

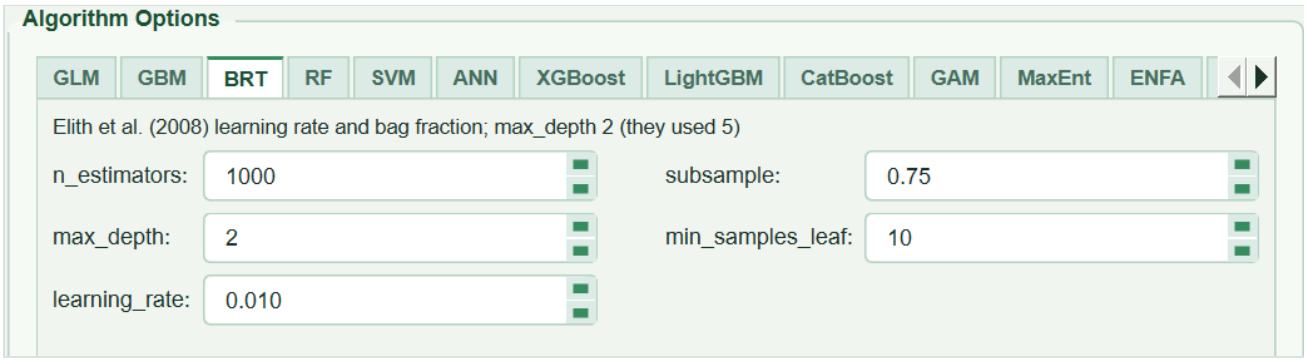


Figure 5.4. BRT tab: note on Elith et al. (2008); n_estimators 1000, max_depth 2, learning_rate 0.010, subsample 0.75, min_samples_leaf 10.

Family and implementation. The same method and the same scikit-learn class as GBM (`GradientBoostingClassifier`), but with defaults chosen according to the guide of Elith et al. (2008): a low learning rate, many trees, 75% bagging (`sdm_core.py:3530-3548`). Presence–pseudo-absence.

Name in the UI	What it does	Default	Range / options	Code
`n_estimators:`	Number of trees	1000	100 – 10 000 (step 100)	`tabs/tab_models.py:765`; `sdm_core.py:3543`
`max_depth:`	Tree depth (Elith et al.: `tree.complexity`)	2	1 – 10	`tabs/tab_models.py:766-767`; `sdm_core.py:3544`
`learning_rate:`	Learning rate	0.010	0.001 – 0.5	`tabs/tab_models.py:768`; `sdm_core.py:3545`
`subsample:`	Bagging fraction (`bag.fraction`)	0.75	0.1 – 1.0	`tabs/tab_models.py:769`; `sdm_core.py:3546`
`min_samples_leaf:`	Minimum rows in a leaf	10	1 – 100	`tabs/tab_models.py:770`; `sdm_core.py:3547`

5.8.1 `max_depth`

- **What it does.** Tree complexity; it determines the level of interaction.
- **When and how to set it.** Elith et al. (2008) used `tree.complexity` = 5 in their examples. HABITUS takes 2 as the default; according to the code comment, on presence–pseudo-absence data deep trees push the suitability of most cells to zero and halve the Boyce index (`sdm_core.py:3532-3537`). For the original setting, enter 5.
- **Effect on results.** As depth increases, interaction and training fit increase and the suitability surface becomes sharper.
- **Caution.** With greater depth, fewer trees are needed for the same fit; it should be considered together with `n_estimators`.

5.8.2 `n_estimators`, `learning_rate`, `subsample`, `min_samples_leaf`

- **What it does.** The same as in GBM (§5.7).
- **When and how to set it.** Elith et al. (2008) recommend choosing the learning rate so that at least 1000 trees are formed; the default 0.01 and 1000 trees follow this rule. `subsample` 0.75 is the bagging fraction recommended by the authors.
- **Effect on results.** If the learning rate is lowered from 0.01 to 0.005 and the number of trees raised to 2000, the model is usually similar but smoother; run time doubles.
- **Caution.** dismo `gbm.step` selects the number of trees itself by 10-fold cross-validation; in HABITUS the number of trees is fixed and BRT is not in the tuning tab. A user who wants to choose the number of trees from the data can search GBM in the tuning tab and use the same values here.

5.9 RF: random forest

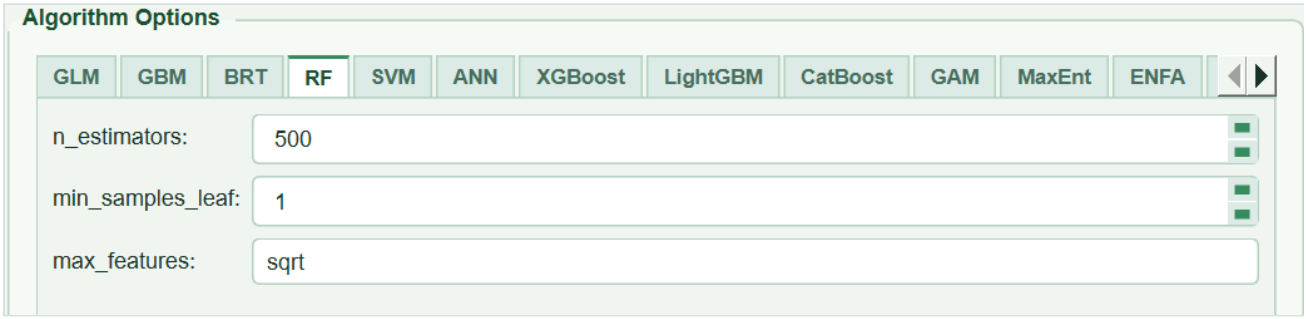


Figure 5.5. RF tab: n_estimators 500, min_samples_leaf 1, max_features sqrt.

Family and implementation. An ensemble of bagged (bootstrap) classification trees; at each split a random subset of the predictors is tried (Breiman 2001). scikit-learn `RandomForestClassifier`; tree depth unlimited, all processor cores used (`sdm_core.py:3549-3561`). Suitability is the mean of the leaf proportions of the trees. Presence–pseudo-absence.

Name in the UI	What it does	Default	Range / options	Code
`n_estimators:`	Number of trees (randomForest: `ntree`)	500	50 – 2000 (step 100)	`tabs/tab_models.py:778`; `sdm_core.py:3555`
`min_samples_leaf:`	Minimum rows in a leaf (randomForest: `nodesize`)	1	1 – 50	`tabs/tab_models.py:779`; `sdm_core.py:3557`

Name in the UI	What it does	Default	Range / options	Code
`max_features:`	Number of predictors tried at each split (randomForest: `mtry`)	`sqrt`	`sqrt`, `log2`, integer, fraction between 0 and 1	`tabs/tab_models.py:780-783`, `1160-1172`; `sdm_core.py:3558`

5.9.1 `n_estimators`

- **What it does.** The number of trees in the forest.
- **When and how to set it.** The code comment notes the recommendation of Oshiro et al. (2012) of at least 128 trees and that 500 is stable on most SDM data (`sdm_core.py:3550-3553`). 500 is sufficient in most cases; if the map looks noisy, it is raised to 1000.
- **Effect on results.** The number of trees does not create overfitting; it only reduces random fluctuation of the prediction. Time and memory increase linearly (in the example run 4 folds 21 s).
- **Caution.** Increasing the number of trees does not change RF's tight fit to the training data (training AUC 1.000 in the example run); leaf size is used for that.

5.9.2 `min_samples_leaf`

- **What it does.** The minimum number of rows in a leaf; 1 means the trees are fully grown (the randomForest classification default is `nodesize` = 1).
- **When and how to set it.** If the training–test AUC difference is large or the map looks patchy, it is raised to 5–10.
- **Effect on results.** As it grows, the probability surface becomes smoother, training AUC falls and overfitting decreases.
- **Caution.** A very large value forces presences into the same leaf as pseudo-absences in data with few presences and lowers discriminatory power.

5.9.3 `max_features`

- **What it does.** The number of candidate predictors chosen at random at each split. `sqrt` = $\lfloor\sqrt{p}\rfloor$ (2 with 8 predictors), `log2` = $\lfloor\log_2 p\rfloor$, integer = that many predictors, a number between 0 and 1 = that fraction of the predictors.
- **When and how to set it.** `sqrt` is the value Breiman recommended for classification and the default `mtry` of R randomForest. The tuning tab, like flexsdm, tries from 2 up to the number of predictors.
- **Effect on results.** A small value makes the trees differ from each other (reduces their correlation); a large value lets strong predictors dominate in every tree.
- **Caution.** An unreadable entry stops the run with the message "RF max_features: '…' is not sqrt, log2, a whole number of variables or a fraction between 0 and 1."; zero or negative values are also rejected (`tabs/tab_models.py:1160-1172`). Entering 1 means "1 predictor", not "100%".

Equivalence with R. The defaults are the same as the R `randomForest(…, ntree = 500, mtry = floor(sqrt(p)), nodesize = 1)` classification. Difference: randomForest computes probability from the proportion of tree votes, scikit-learn from the mean of leaf proportions; with leaf size 1 the two are very close. The down-sampled RF recommended by Valavi et al. (2021) is not available in HABITUS; `Balance classes` applies weighting, not down-sampling.

5.10 SVM: support vector machine

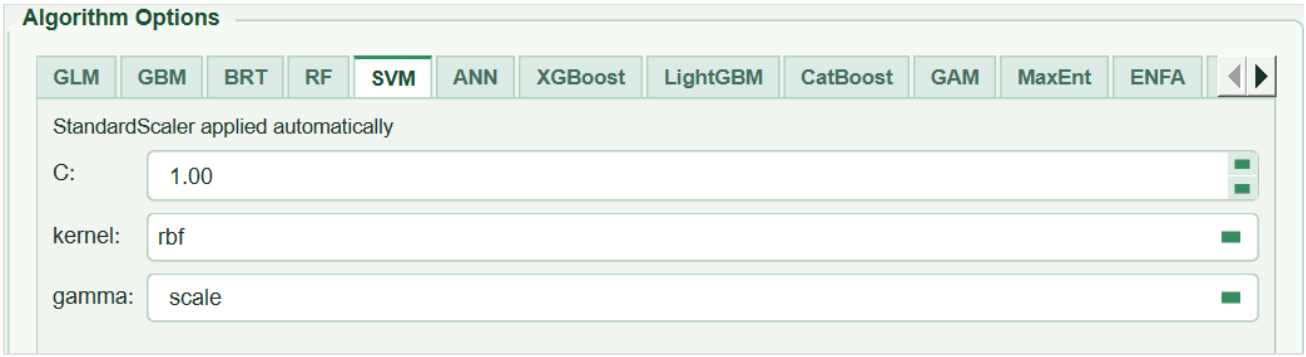


Figure 5.6. SVM tab: StandardScaler note; C 1.00, kernel rbf, gamma scale.

Family and implementation. Kernel-based margin classifier. scikit-learn `SVC` with `probability=True`: probabilities are obtained by Platt scaling in scikit-learn's own internal cross-validation (`sdm_core.py:3562-3568`). Predictors are standardised, categoricals are one-hot encoded. Presence–pseudo-absence.

Name in the UI	What it does	Default	Range / options	Code
`C:`	Error penalty; smaller gives a smoother boundary	1.00	0.001 – 1000	`tabs/tab_models.py:789`; `sdm_core.py:3565`
`kernel:`	Kernel function	`rbf`	`rbf`, `linear`, `poly`, `sigmoid`	`tabs/tab_models.py:790`
`gamma:`	Kernel width (RBF, poly, sigmoid)	`scale`	`scale`, `auto` or a number	`tabs/tab_models.py:791-793`, `1156-1159`

5.10.1 `C`

- **What it does.** The penalty for misclassified training points. A large C gives a complex boundary that fits the training data tightly, a small C a smoother boundary.
- **When and how to set it.** flexsdm `tune_svm` tries the range 1–16; the tuning tab uses the same grid.
- **Effect on results.** As C grows, training AUC rises and overfitting increases.
- **Caution.** SVM run time grows roughly on a quadratic-to-cubic scale with the number of rows; it is slow with tens of thousands of pseudo-absences.

5.10.2 `kernel` and `gamma`

- **What it does.** The RBF kernel is $k(x, x') = \exp(-\gamma \|x - x'\|^2)$; γ determines the radius of influence of a training point. `scale` = $1/(p \cdot \mathrm{Var}(X))$, `auto` = $1/p$. Because the predictors are standardised, the two give the same value with continuous variables; they differ if one-hot columns are present.
- **When and how to set it.** RBF is sufficient for most SDMs. A number can also be typed in the box; the tuning tab writes a number here. flexsdm's kernlab `sigma` has the same definition as scikit-learn's `gamma`, no conversion is needed (`sdm_tuning.py:654-675`).
- **Effect on results.** A large γ gives a narrow area of influence and a patchy, overfitted surface; a small γ gives a smooth, nearly linear surface. With the `linear` kernel `gamma` has no effect.
- **Caution.** Because Platt scaling uses its own internal cross-validation, the probability ranking may not be exactly the same as the `decision_function` ranking; this is documented scikit-learn behaviour.

5.11 ANN: artificial neural network (multilayer perceptron)

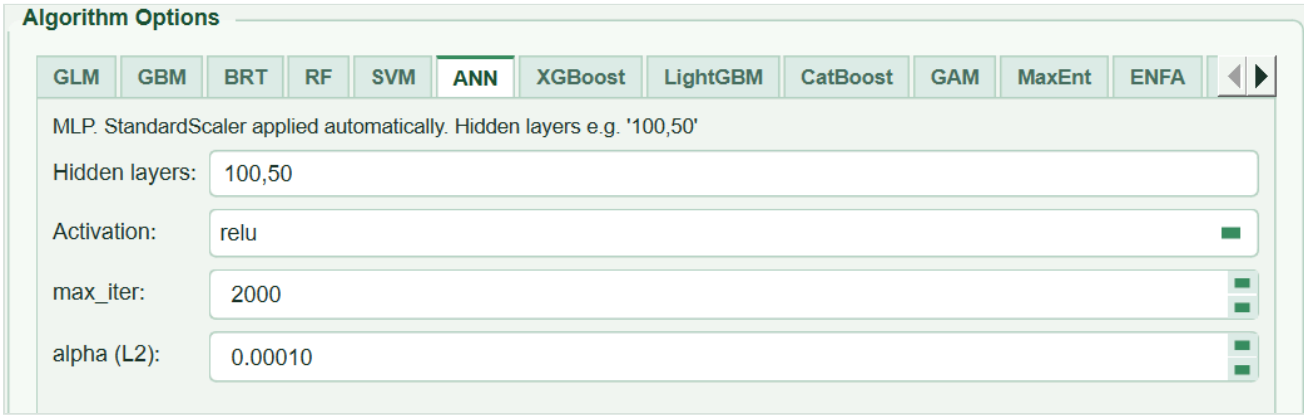


Figure 5.7. ANN tab: Hidden layers 100,50; Activation relu; max_iter 2000; alpha (L2) 0.00010.

Family and implementation. Feed-forward neural network. scikit-learn `MLPClassifier` (Adam optimiser, the scikit-learn default); predictors are standardised, categoricals are one-hot encoded (`sdm_core.py:3569-3581`). Presence–pseudo-absence.

Name in the UI	What it does	Default	Range / options	Code
`Hidden layers:`	Numbers of neurons in the hidden layers, comma-separated	`100,50`	list of positive integers	`tabs/tab_models.py:799`, `1144-1154`
`Activation:`	Activation function	`relu`	`relu`, `tanh`, `logistic`	`tabs/tab_models.py:800`
`max_iter:`	Maximum training iterations (epochs)	2000	100 – 5000	`tabs/tab_models.py:801`
`alpha (L2):`	L2 penalty on the weights	0.00010	0.0 – 100.0	`tabs/tab_models.py:802-804`; `sdm_core.py:3579`

5.11.1 `Hidden layers`

- **What it does.** `100,50` means two hidden layers (100 and 50 neurons); `5` means 5 neurons in a single layer.
- **When and how to set it.** The default network carries thousands of weights; it is large for typical SDM data with one or two hundred presences. R nnet (biomod2, flexsdm) uses a single hidden layer and a few neurons; flexsdm `tune_net` tries 2–8 neurons. With few records a single layer and 4–8 neurons are recommended.
- **Effect on results.** As neurons increase, flexibility and overfitting increase and run time lengthens.
- **Caution.** An unreadable entry stops the run with the message "ANN hidden layers: '...' is not a list of positive whole numbers separated by commas (e.g. 100, 50)." (`tabs/tab_models.py:1144-1154`).

5.11.2 `alpha (L2)`

- **What it does.** A penalty on the squares of the weights (weight decay).
- **When and how to set it.** nnet's `decay` parameter corresponds to $\alpha = 2 \times$ `decay` (tooltip and `sdm_tuning.py:678-689`; this relation is derived from the two objective functions and has not been tested numerically against R nnet). The decay range 0.01–10 of the flexsdm grid corresponds to alpha 0.02–20; the default 0.0001 is much smaller than these.
- **Effect on results.** As it grows, the surface becomes smoother and overfitting decreases.
- **Caution.** A large network + a small alpha is the setting that overfits most easily.

5.11.3 `Activation` and `max_iter`

- **What it does.** `relu` produces a piecewise linear response, `tanh` and `logistic` smooth S-shaped responses. `max_iter` is the maximum number of epochs of the optimiser.
- **When and how to set it.** nnet uses `logistic` activation; for comparison with R, `logistic` can be chosen. If a convergence warning appears, `max_iter` is increased.
- **Effect on results.** `relu` gives angular transitions in response curves, `logistic` / `tanh` smoother curves.
- **Caution.** Early stopping is not used; the network is trained up to `max_iter` or scikit-learn's convergence criterion.

5.12 XGBoost: extreme gradient boosting

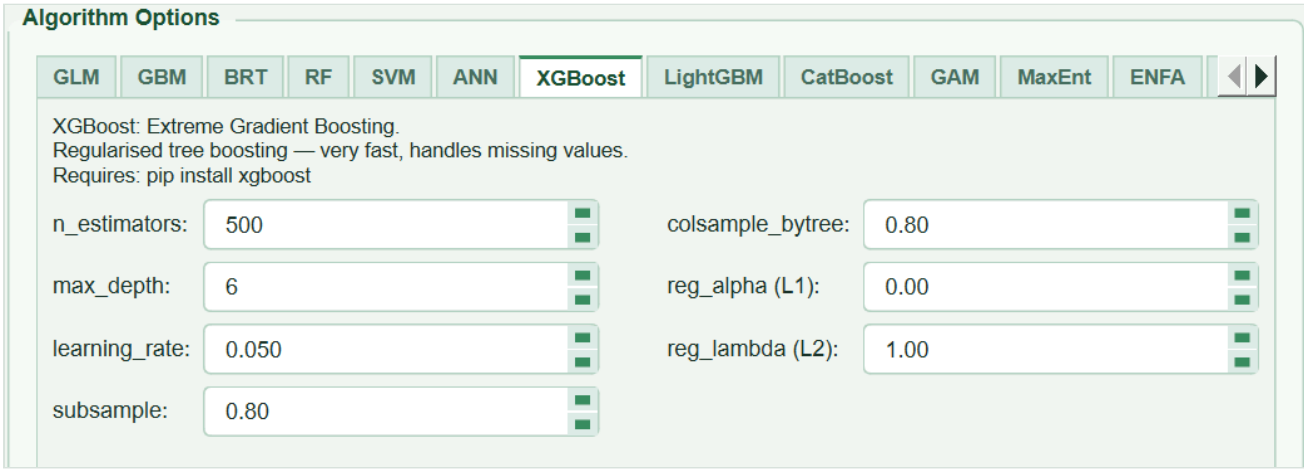


Figure 5.8. XGBoost tab: description note; n_estimators 500, max_depth 6, learning_rate 0.050, subsample 0.80, colsample_bytree 0.80, reg_alpha (L1) 0.00, reg_lambda (L2) 1.00.

Family and implementation. Regularised gradient-boosted trees (Chen and Guestrin 2016), `xgboost.XGBClassifier`, loss function log-loss (`sdm_core.py:3582-3608`). Presence–pseudo-absence. Two settings not in the UI are fixed: `min_child_weight` = 1, `gamma` = 0.

Name in the UI	What it does	Default	Range / options	Code
`n_estimators:`	Number of boosting rounds (trees)	500	50 – 5000	`tabs/tab_models.py:815`
`max_depth:`	Tree depth	6	1 – 12	`tabs/tab_models.py:816`
`learning_rate:`	Learning rate (eta)	0.050	0.001 – 0.5	`tabs/tab_models.py:817`
`subsample:`	Row fraction per tree	0.80	0.1 – 1.0	`tabs/tab_models.py:818`
`colsample_bytree:`	Predictor fraction per tree	0.80	0.1 – 1.0	`tabs/tab_models.py:819`

Name in the UI	What it does	Default	Range / options	Code
`reg_alpha (L1):`	L1 penalty on leaf weights	0.00	0 – 10	`tabs/tab_models.py:820`
`reg_lambda (L2):`	L2 penalty on leaf weights	1.00	0 – 10	`tabs/tab_models.py:821`

5.12.1 `n_estimators`, `learning_rate`, `max_depth`

- **What it does.** Number of trees, learning rate and depth as in GBM.
- **When and how to set it.** The default depth of 6 is very deep compared with GBM (1) and BRT (2) and allows a high level of interaction; with a few hundred presences 2–3 is safer. The grid of the tuning tab is depth 3/6/9, rate 0.01/0.05/0.1, rounds 100/300/500.
- **Effect on results.** As depth and rounds increase, training fit and overfitting increase.
- **Caution.** There is no early stopping; the number of rounds is fixed.

5.12.2 `subsample` and `colsample_bytree`

- **What it does.** Each tree is built with a random subset of the rows and predictors of these fractions.
- **When and how to set it.** The range 0.5–0.9 is common; with few predictors (e.g. 4), if `colsample_bytree` is reduced some trees never see an important variable.
- **Effect on results.** Small fractions reduce variance and overfitting.
- **Caution.** The result depends on the seed; it is reproducible because the seed is fixed.

5.12.3 `reg_alpha` and `reg_lambda`

- **What it does.** L1 and L2 penalties on leaf values; they shrink leaf predictions.
- **When and how to set it.** The defaults are XGBoost's own defaults (0 and 1). If overfitting is seen, `reg_lambda` can be raised to 5–10.
- **Effect on results.** As the penalty increases, the probability surface is compressed towards 0.5 and becomes smoother.
- **Caution.** The UI note says that XGBoost can handle missing values; in HABITUS, rows with missing values are already removed in data preparation, so this feature is not used.

5.13 LightGBM: light gradient boosting

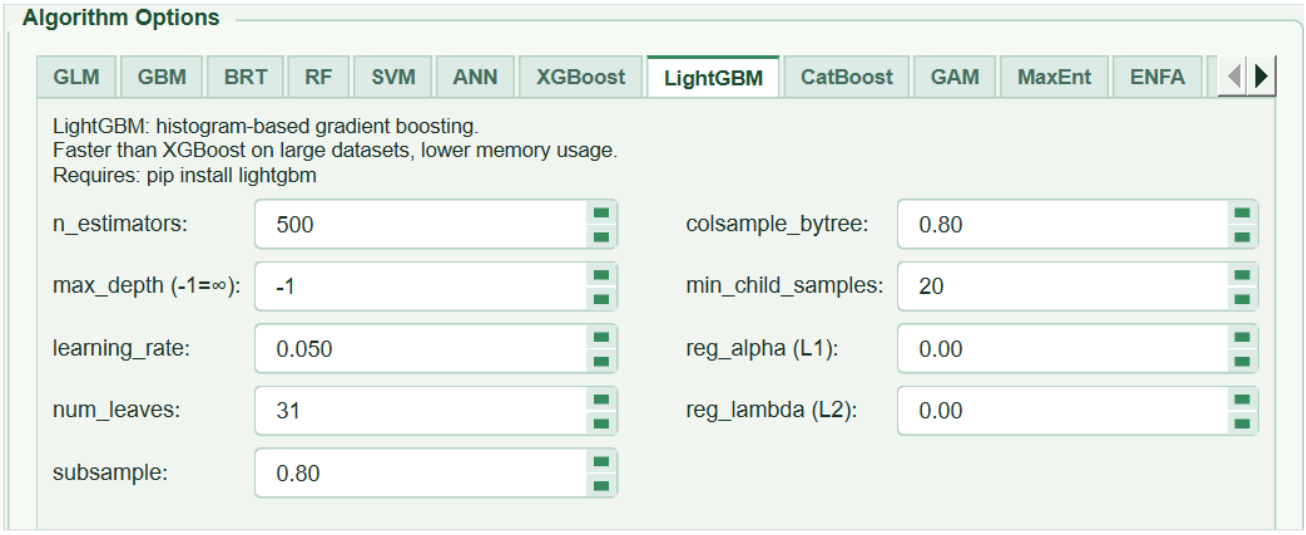


Figure 5.9. LightGBM tab: description note; n_estimators 500, max_depth -1, learning_rate 0.050, num_leaves 31, subsample 0.80, colsample_bytree 0.80, min_child_samples 20, reg_alpha 0.00, reg_lambda 0.00.

Family and implementation. Histogram-based gradient boosting with leaf-wise growth; `lightgbm.LGBMClassifier` (`sdm_core.py:3610-3628`). Tree size is controlled by the number of leaves, not by depth. Presence–pseudo-absence.

Name in the UI	What it does	Default	Range / options	Code
`n_estimators:`	Number of rounds	500	50 – 5000	`tabs/tab_models.py:837`
`max_depth (-1=∞):`	Depth limit	-1 (unlimited)	-1 – 100	`tabs/tab_models.py:838`
`learning_rate:`	Learning rate	0.050	0.001 – 0.5	`tabs/tab_models.py:839`
`num_leaves:`	Maximum leaves per tree	31	2 – 255	`tabs/tab_models.py:840`
`subsample:`	Row subsampling fraction	0.80	0.1 – 1.0	`tabs/tab_models.py:841`
`colsample_bytree:`	Predictor fraction per tree	0.80	0.1 – 1.0	`tabs/tab_models.py:842`
`min_child_samples:`	Minimum rows in a leaf	20	1 – 200	`tabs/tab_models.py:843`
`reg_alpha (L1):`	L1 penalty	0.00	0 – 10	`tabs/tab_models.py:844`
`reg_lambda (L2):`	L2 penalty	0.00	0 – 10	`tabs/tab_models.py:845`

5.13.1 `num_leaves` and `max_depth`

- **What it does.** `num_leaves` directly limits the complexity of the tree; `max_depth` = -1 leaves depth unlimited.
- **When and how to set it.** 31 leaves is more complex than a balanced tree of depth 5; with few records 7–15 leaves and `max_depth` 3–4 are more suitable. The tuning tab tries 15/31/63.
- **Effect on results.** As leaves increase, interaction and overfitting increase.
- **Caution.** Leaf-wise growth can produce very deep, narrow branches with little data; limiting with `max_depth` prevents this.

5.13.2 `min_child_samples`

- **What it does.** The minimum number of rows that must be in a leaf.
- **When and how to set it.** Default 20; with fewer than 50 presences it can make it hard to form leaves containing presences and can be lowered to 5–10.
- **Effect on results.** As it grows, the model becomes coarser.
- **Caution.** With very few presences the model may find no split at all and give an almost constant prediction.

5.13.3 `subsample` and `colsample_bytree`

- **What it does.** Row and predictor subsampling fractions.
- **When and how to set it.** `colsample_bytree` is applied in each tree.
- **Effect on results.** As `colsample_bytree` decreases, the trees become more diverse.
- **Caution.** In LightGBM, row subsampling (`subsample`) is active only when `subsample_freq` is greater than zero; because HABITUS does not pass this setting (`sdm_core.py:3613-3628`), the `subsample` value is not reflected in the result.

5.13.4 `n_estimators`, `learning_rate`, `reg_alpha`, `reg_lambda`

- **What it does.** The same meanings as in XGBoost (§5.12).
- **When and how to set it.** LightGBM's own default penalties are zero; in case of overfitting `reg_lambda` is increased.
- **Effect on results.** See §5.12.
- **Caution.** There is no early stopping.

5.14 CatBoost: categorical boosting

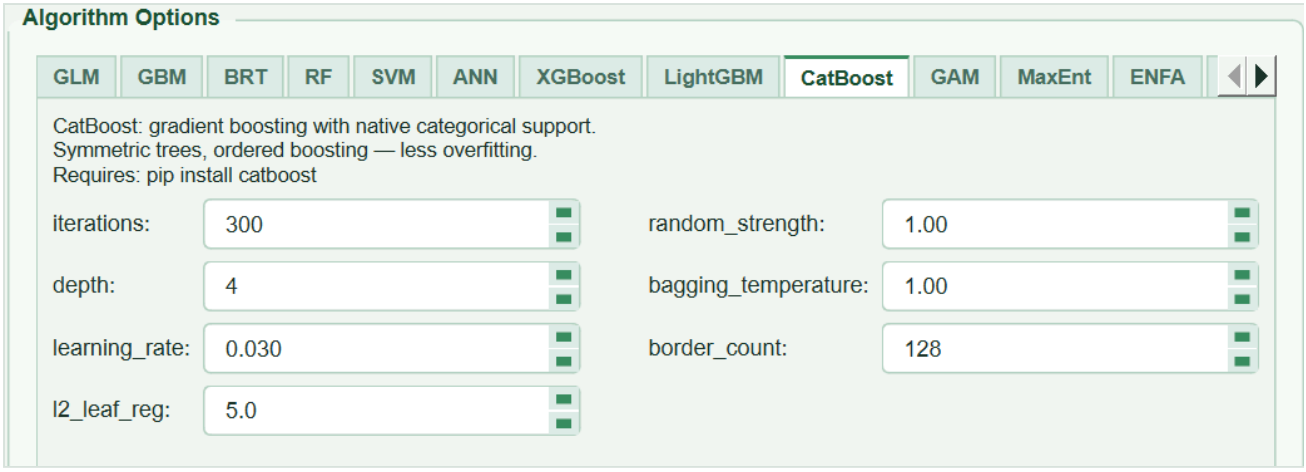


Figure 5.10. CatBoost tab: description note; iterations 300, depth 4, learning_rate 0.030, l2_leaf_reg 5.0, random_strength 1.00, bagging_temperature 1.00, border_count 128.

Family and implementation. Ordered boosting with symmetric trees (using the same split at each level); `catboost.CatBoostClassifier` (`sdm_core.py:3630-3647`). Presence–pseudo-absence.

Name in the UI	What it does	Default	Range / options	Code
`iterations:`	Number of rounds	300	50 – 5000	`tabs/tab_models.py:863`
`depth:`	Symmetric tree depth	4	1 – 16	`tabs/tab_models.py:864`
`learning_rate:`	Learning rate	0.030	0.001 – 0.5	`tabs/tab_models.py:865`
`l2_leaf_reg:`	L2 penalty on leaf values	5.0	0 – 20	`tabs/tab_models.py:866`
`random_strength:`	Noise added to split scores	1.00	0.01 – 10	`tabs/tab_models.py:867`
`bagging_temperature:`	Strength of the Bayesian bootstrap weights	1.00	0 – 10	`tabs/tab_models.py:868`
`border_count:`	Number of split borders for continuous predictors	128	1 – 255	`tabs/tab_models.py:869`

5.14.1 `iterations`, `learning_rate`, `depth`

- **What it does.** Number of rounds, learning rate and tree depth. In a symmetric tree, depth d means exactly 2^d leaves (4 → 16 leaves).
- **When and how to set it.** The defaults (300 rounds, 0.03, depth 4) are a smaller model than CatBoost's library defaults (1000 rounds, depth 6). CatBoost is not in the tuning tab; values are tried by hand.
- **Effect on results.** As depth and rounds increase, training fit increases.
- **Caution.** Rate and number of rounds should be considered together (§5.7.1).

5.14.2 `l2_leaf_reg`, `random_strength`, `bagging_temperature`

- **What it does.** `l2_leaf_reg` shrinks leaf values; `random_strength` adds randomness to split selection; `bagging_temperature` determines the randomness of the row weights (0 = all rows equally weighted).
- **When and how to set it.** In case of overfitting `l2_leaf_reg` is increased (CatBoost's own default is 3, HABITUS uses 5).
- **Effect on results.** As all three grow, the model becomes smoother and its sensitivity to the seed increases.
- **Caution.** `bagging_temperature` = 0 does not remove randomness; `random_strength` is still in effect.

5.14.3 `border_count`

- **What it does.** Into how many possible split points (quantile borders) each continuous predictor is divided.
- **When and how to set it.** 128 is sufficient resolution for most bioclimatic variables.
- **Effect on results.** A small value makes response curves stepped; a large value lengthens run time.
- **Caution.** In HABITUS, categorical variables are not declared to CatBoost as categorical; the integer code is split with these borders like a continuous column (§5.4.3). The "native categorical support" in the UI note is therefore not used in HABITUS.

5.15 GAM: generalised additive model

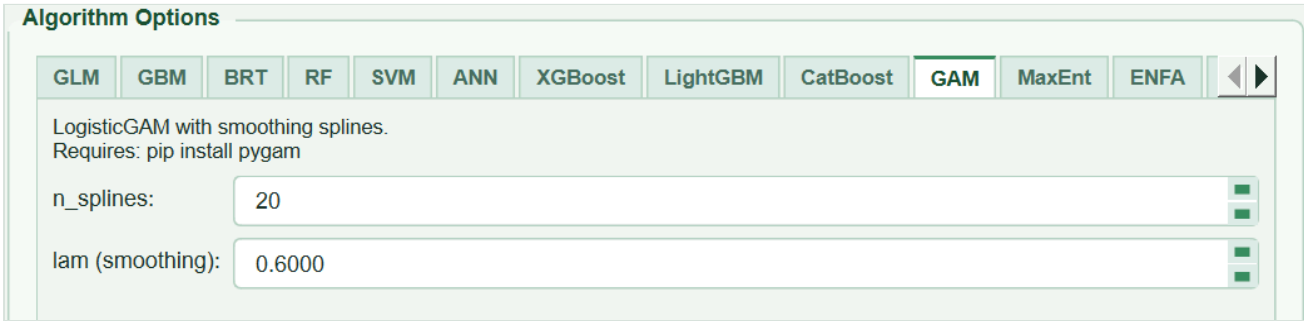


Figure 5.11. GAM tab: LogisticGAM note; n_splines 20, lam (smoothing) 0.6000.

Family and implementation. Logistic additive model: the logit is the sum of a smooth (spline) function of each predictor (Hastie and Tibshirani 1986). pygam `LogisticGAM`; each continuous predictor is given a penalised B-spline term `s()` and each categorical predictor a factor term `f()`; there are no interaction terms (`sdm_core.py:4500-4549`). Presence–pseudo-absence. Works with raw values.

Name in the UI	What it does	Default	Range / options	Code
`n_splines:`	Number of spline basis functions per predictor	20	4 – 100	`tabs/tab_models.py:881-886`; `sdm_core.py:4508-4512`
`lam (smoothing):`	Smoothing penalty	0.6000	0.0001 – 1000	`tabs/tab_models.py:887-892`; `sdm_core.py:4513-4514`

5.15.1 `n_splines`

- **What it does.** The number of basis functions from which each response curve is built; it determines the maximum flexibility of the curve. Because the spline order is 3, it must be greater than 3; otherwise the message "GAM n_splines must exceed the spline order (3)." is given.
- **When and how to set it.** 20 is pygam's default. With few records 5–10 is sufficient and means fewer parameters; the tuning tab tries 5/10/15/20.
- **Effect on results.** As it increases, the curves can become more wiggly (to the extent the penalty allows); run time increases.
- **Caution.** The same value is applied to all predictors.

5.15.2 `lam` (smoothing penalty)

- **What it does.** The penalty coefficient applied to the second derivative (curvature) of the curve. A large `lam` brings the curve closer to a straight line.
- **When and how to set it.** 0.6 is pygam's default. If the response curves are wavy and not ecologically plausible, it is raised to 2–10; the tuning tab tries 0.1/0.6/2/10.
- **Effect on results.** In the example run (with defaults) the training–test AUC difference of GAM is 0.087; almost three times that of GLM (0.031). This indicates that smoothing remains light on these data.
- **Caution.** The penalty is not selected automatically from the data; the `lam` entered is used as a fixed value.

Equivalence with R. The GAMs of biomod2 and flexsdm use mgcv; mgcv estimates the smoothing parameter from the data (GCV/REML) and its default basis is the thin plate regression spline. The pygam model in

HABITUS works with a fixed `lam`; the response curves of the two programs can therefore differ in smoothness on the same data.

> **Caution:** GAM does not support class balancing; `Balance classes` does not affect GAM (`sdm_core.py:3681`).

5.16 MaxEnt: maximum entropy

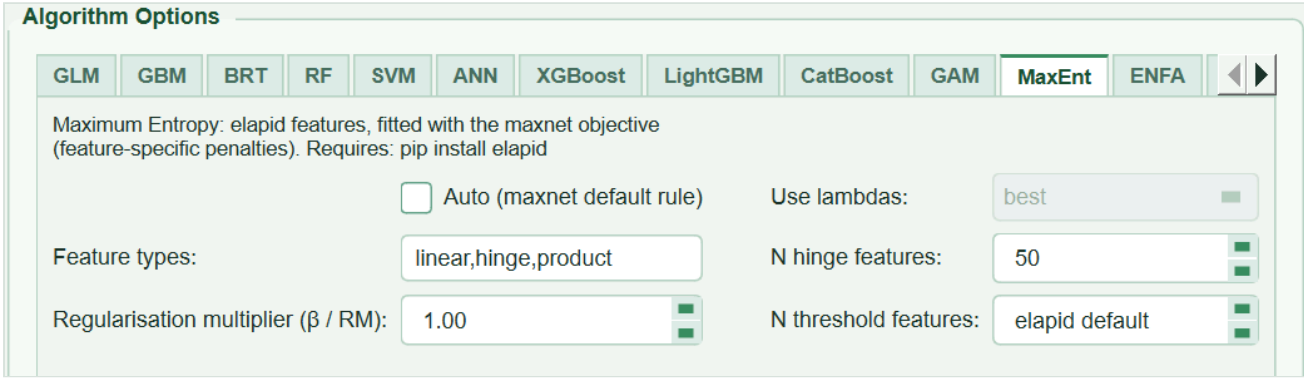


Figure 5.12. MaxEnt tab: note on elapid and the maxnet objective function; Auto (maxnet default rule) off, Feature types linear,hinge,product, Regularisation multiplier (β / RM) 1.00, Use lambdas best (disabled), N hinge features 50, N threshold features elapid default.

Family and implementation. Presence + background method (Phillips et al. 2006, 2017). "Features" (linear, quadratic, product, hinge, threshold) are derived from the predictors and an L1-penalised model determines which features remain. HABITUS generates the features with the elapid library (Anderson 2023) and estimates the coefficients by solving the objective function of R maxnet (`sdm_core.py:3776-3836`, `4551-4757`). The output is elapid's cloglog transformation and is not rescaled afterwards (`sdm_core.py:4772-4845`).

Name in the UI	What it does	Default	Range / options	Code
`Auto (maxnet default rule)`	Selects feature classes from the number of presences	off	on / off	`tabs/tab_models.py:1056-1063`; `sdm_core.py:3758-3773`
`Feature types:`	Feature classes, comma-separated	`linear,hinge,product`	`linear`, `quadratic`, `hinge`, `product`, `threshold`	`tabs/tab_models.py:1048-1055`; `sdm_core.py:4616-4624`
`Regularisation multiplier (β / RM):`	Regularisation multiplier	1.00	0.1 – 20.0 (step 0.5)	`tabs/tab_models.py:1064-1077`; `sdm_core.py:4625`
`Use lambdas:`	Not used (disabled)	`best`	`best`, `last`	`tabs/tab_models.py:1078-1085`
`N hinge features:`	Number of hinge knots per predictor	50	10 – 200	`tabs/tab_models.py:1086`; `sdm_core.py:4626`
`N threshold features:`	Number of threshold features per predictor	`elapid default` (0 = elapid's 10)	0 – 200	`tabs/tab_models.py:1087-1091`; `sdm_core.py:4645-4649`

5.16.1 The maxnet objective function is applied exactly

Because there is no Python counterpart of R glmnet in HABITUS builds, elapid on its own falls back to a scikit-learn logistic regression with a single penalty coefficient. On this path there are no feature-specific penalties and the regularisation multiplier barely changes the model (`sdm_core.py:3780-3787`). HABITUS therefore uses the feature transformer built by elapid but re-estimates the coefficients with the `maxnet_refit` function (`sdm_core.py:4719-4746`). maxnet solves glmnet along a λ path with `standardize = FALSE`, per-feature penalty factors and weights $w_i = p_i + 100(1 - p_i)$ (presence 1, background 100) and keeps the last λ. The problem solved at this last λ is, up to a constant factor (`sdm_core.py:3789-3797`):

$$\min_{\beta_0,\beta} \sum_i w_i \, \ell_i(\beta_0, \beta) \; + \; n_{\text{pres}} \sum_j r_j \, |\beta_j|$$

Here ℓ_i is the logistic loss and r_j the regularisation value of feature j; n_{pres} is the number of presences. r_j is computed with elapid's `compute_regularization` function according to the feature class, the number of presences, the distribution of the feature among the presences and the β multiplier (`sdm_core.py:3810-3816`). The intercept is unpenalised. The problem is solved with liblinear, with the features divided by $n_{\text{pres}} \cdot r_j$; then the MaxEnt normalisation constants (alpha, entropy) are recomputed from the background rows (`sdm_core.py:3817-3835`). In the run record this solver is written as "maxnet objective (feature-specific L1 penalties, last lambda)" (`sdm_core.py:4743-4744`).

Presences are added to the background. As with maxnet's `addsamplestobackground = TRUE` and Maxent's default behaviour, every presence whose predictor values are not found among the background rows is also added to the background (`sdm_core.py:4698-4707`). The number added is written to the `maxent_settings_used.presences_added_to_background` field in `habitus_run_config.json`; in the full model of the example run it is 91.

Automated test against R maxnet. `tests/test_maxent_maxnet.py` compares HABITUS's cloglog prediction with R maxnet 0.1.4 output on a subset of the data of Rutrough et al. (2019) (302 presences, 2000 background rows; ppt, silt, slope). Three settings are tested (LQH β=3, LQH β=1, LQPH β=3); in each, the Pearson and Spearman correlations are expected to exceed 0.995 and the mean absolute difference to be below 0.02 (`tests/test_maxent_maxnet.py:39-50`). Two additional tests check that the number of non-zero coefficients decreases as β grows (`53-59`) and that presences are added to the background (`62-64`). The R output is a file stored with the test; the test itself does not run R. The code comment reports agreement with R maxnet of Schoener D = 0.998 on the same data (`sdm_core.py:3799-3800`).

5.16.2 `Feature types` and `Auto (maxnet default rule)`

- **What it does.** Determines which feature classes are generated: `linear`, `quadratic`, `product` (pairwise product), `hinge` (a piecewise function that increases or decreases linearly after a knot), `threshold` (a step). If `Auto` is ticked, the classes are selected for each fit from the number of training presences by the maxnet rule: fewer than 10 presences L; 10–14 LQ; 15–79 LQH; 80 and above LQHP. The threshold class is never used in the automatic rule; while the box is ticked, the text box is ignored and disabled.
- **When and how to set it.** With few records `linear` or `linear,quadratic` is a defensible choice (tooltip). Merow et al. (2013) recommend choosing complexity according to the question and the data. For a search as in ENMeval/Wallace, the tuning tab tries the sets L, LQ, H, LQH, LQHP.
- **Effect on results.** Few classes give smooth and transferable responses; many classes (especially hinge and threshold) give stepped responses that fit the calibration data tightly. With `Auto` on, the cross-validation folds and the full model may use different class sets depending on their numbers of presences (`sdm_core.py:4619-4622`); the rule used is written to the record.
- **Caution.** The default `linear,hinge,product` is not the same as the maxnet rule (there is no quadratic class). If a comparison with maxnet is to be made, `Auto` should be ticked.

5.16.3 `Regularisation multiplier (β / RM)`

- **What it does.** The coefficient that multiplies all r_j values. As it grows, the L1 penalty increases and fewer features remain non-zero.
- **When and how to set it.** The maxnet and Maxent default is 1. There is no single correct value; ENMeval (Muscarella et al. 2014) and kuenm therefore search for the multiplier. The tuning tab crosses rm 0.5–4.0 (step 0.5) with the feature sets and writes the selected value here. The best value depends on the calibration area; if the area changes it should be searched again (tooltip).
- **Effect on results.** That the number of non-zero coefficients decreases when β is raised from 1 to 6 is checked by a test (`tests/test_maxent_maxnet.py:53-59`). On the map, a large β gives broader and softer suitability, a small β narrower and patchier suitability.
- **Caution.** The effect of the value is meaningful only with the maxnet objective function; HABITUS uses this path in every fit and writes to the record that it was used.

5.16.4 `N hinge features` and `N threshold features`

- **What it does.** The number of hinge knots and threshold features to be generated per predictor.
- **When and how to set it.** 50 hinges is maxnet's default. The number of thresholds is used only if the `threshold` class is selected; 0 leaves elapid's own value (10).
- **Effect on results.** As the number of knots increases, the response curve can capture finer detail; the penalty limits this.
- **Caution.** In the tuning tab these two values are held fixed for all grid points (`sdm_tuning.py:550-551`).

5.16.5 `Use lambdas`

The box is disabled and does not affect the result: the model is fitted at the last value of the λ path, as in maxnet (`tabs/tab_models.py:1079-1085`). The value is nevertheless written to the record (`use_lambdas`).

> **Note:** In MaxEnt cross-validation the background is not split into folds; every fold is trained with the same background. Test scores are computed with the presences and pseudo-absences in the test fold. The size and sampling scheme of the background are set in ① Data (Chapter 3). If the installed elapid does not recognise a setting, the setting is not silently dropped; the warning "MaxEnt: the installed elapid (…) has no setting for …" is given (`sdm_core.py:4684-4691`).

5.17 ENFA: ecological niche factor analysis



Figure 5.13. ENFA tab: note on Hirzel et al. (2002); Specialisation axes (K) 3.

Family and implementation. A multivariate niche method that compares the environment used by the species with the general (global) environment (Hirzel et al. 2002). In HABITUS it is written with numpy (`sdm_core.py:4046-4204`):

1. The global mean and covariance are estimated from the presence-only background in ① Data if available, otherwise from all rows of the training fold (presence + pseudo-absence); the covariance is computed with Ledoit–Wolf shrinkage and 1e-6 is added to its diagonal.
2. The data are whitened with the global covariance; the marginality axis is the direction of the difference between the species mean and the global mean.
3. In the subspace orthogonal to marginality, an eigendecomposition of the covariance of the presences is performed and the K axes with the largest eigenvalues are kept.
4. The raw score is $\exp\{-\frac{1}{2}[d_m^2 + \sum_k (z_k - \mu_k)^2/\lambda_k]\}$; d_m is the distance to the species mean along the marginality axis.
5. During fitting, the score is scaled to 0–1 with the score range of the training rows; this range is stored with the model, so the same cell gets the same value in every prediction (`sdm_core.py:3719-3743`, `4158-4162`).

Categorical variables are left out. At least 2 presences are needed.

Name in the UI	What it does	Default	Range / options	Code
`Specialisation axes (K):`	Number of specialisation axes to keep	3	1 – 10	`tabs/tab_models.py:1102-1104`; `sdm_core.py:4074`, `4133`

5.17.1 `Specialisation axes (K)`

- **What it does.** Determines how many additional axes enter the score besides the marginality axis. It is limited to one less than the number of predictors (`sdm_core.py:4133`).
- **When and how to set it.** 2–4 axes carry most of the variance in most cases; with few predictors (e.g. 4) K can be at most 3.
- **Effect on results.** As K increases, the score measures distance in more dimensions and the suitable area narrows.
- **Caution.** ENFA takes no other settings apart from class weighting and clamping; scale and regularisation constants are fixed in the code.

Comparison with R. adehabitatHS `enfa` computes the marginality and specialisation axes; for the suitability map, the original work of Hirzel et al. uses a separate computation. HABITUS's ENFA score is the Gaussian-shaped distance above; there is no numerical comparison test against R implementations.

5.18 Mahalanobis distance

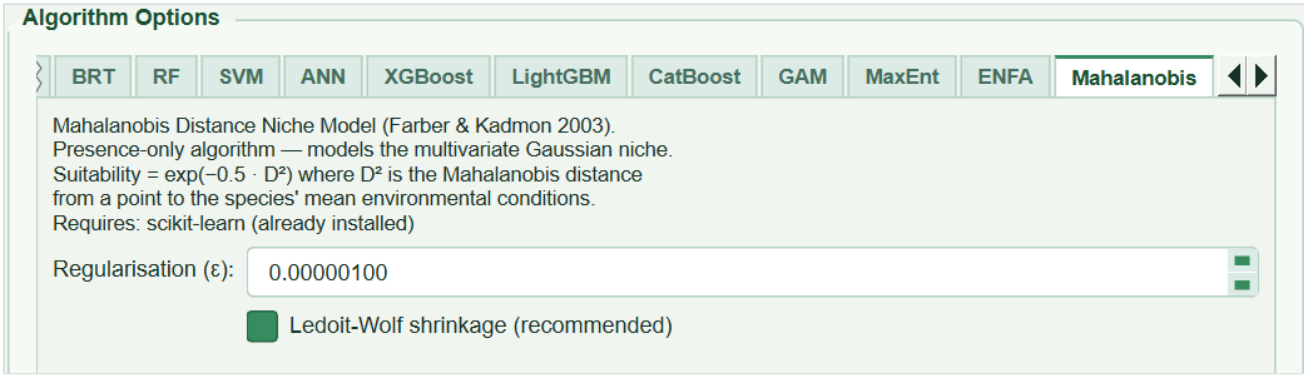


Figure 5.14. Mahalanobis tab: note on Farber and Kadmon (2003); Regularisation (ε) 0.00000100; Ledoit-Wolf shrinkage (recommended) ticked.

Family and implementation. Presence-only distance method: the Mahalanobis distance $D^2(x) = (x - \mu)^\top \Sigma^{-1}(x - \mu)$ to the mean environment of the presences is computed and the raw score is $\exp(-D^2/2)$ (Farber and Kadmon 2003). The covariance is estimated from the presences only (`sdm_core.py:4207-4273`). As in ENFA, the score is scaled to 0–1 with the score range of the training rows (presence + pseudo-absence) and this range is stored. Categorical variables are left out; at least 2 presences are needed.

Name in the UI	What it does	Default	Range / options	Code
`Regularisation (ε):`	Constant added to the covariance diagonal	0.00000100	0.0 – 1.0	`tabs/tab_models.py:1116-1124`; `sdm_core.py:4237, 4245`
`Ledoit-Wolf shrinkage (recommended)`	Estimates the covariance with Ledoit–Wolf shrinkage	on	on / off	`tabs/tab_models.py:1125-1129`; `sdm_core.py:4238-4244`

5.18.1 `Ledoit-Wolf shrinkage`

- **What it does.** Shrinks the sample covariance towards a diagonal target by the optimal amount (Ledoit and Wolf 2004); if off, the ordinary sample covariance is used.
- **When and how to set it.** If the number of presences is less than a few times the number of predictors, it must be kept on.
- **Effect on results.** Shrinkage stabilises the distances; with correlated predictors the ellipse becomes rounder and the suitable area widens somewhat.
- **Caution.** If it is turned off and the predictors are strongly correlated, the matrix becomes nearly singular; inversion becomes unstable.

5.18.2 `Regularisation (ε)`

- **What it does.** By using $\Sigma + \varepsilon I$, the matrix is kept invertible.
- **When and how to set it.** It is increased if a `LinAlgError` is obtained (tooltip); the range 1e-4–1e-2 can be tried.
- **Effect on results.** A large ε reduces distances in directions of small variance and widens the suitable area.
- **Caution.** The box shows 8 decimal places; smaller values are rounded to zero.

Comparison with R. dismo `mahal` gives the raw distance with a different transformation; HABITUS's $\exp(-D^2/2)$ transformation and scaling with the training range differ from it. When the same mean and covariance are used, the ranking (and hence AUC) is largely the same; only HABITUS's clipping to the training range can create tied values in cells outside the range.

5.19 BIOCLIM: climate envelope

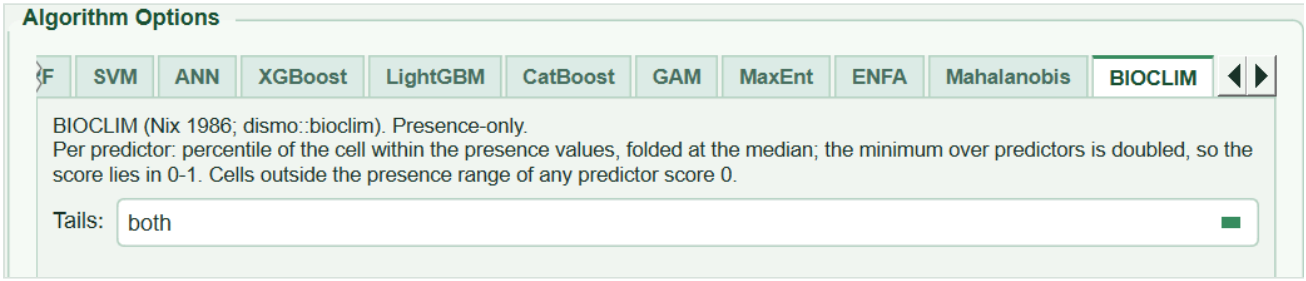


Figure 5.15. BIOCLIM tab: method note (Nix 1986; dismo::bioclim); Tails both.

Family and implementation. Presence-only envelope method (Nix 1986). Written with numpy (`sdm_niche.py:44-90`). For each predictor, the percentile p of the cell value among the presence values is computed with the mid-rank rule; if $p > 0.5$, $1 - p$ is taken (folded at the median); the minimum over predictors is taken and multiplied by two. The result lies between 0 and 1: 1 means at the median of all predictors, 0 means outside the presence range of at least one predictor. The score is not rescaled. Categorical variables are left out; at least 2 presences are needed.

Name in the UI	What it does	Default	Range / options	Code
`Tails:`	Which tail lowers the score	`both`	`both`, `low`, `high`	`tabs/tab_models.py:900-907`; `sdm_niche.py:79-88`

5.19.1 `Tails`

- **What it does.** `both` is classical BIOCLIM: both tails lower the score. `low` counts only the lower tail (above the median 1, and above the presence range also 1); `high` counts only the upper tail. It has the same meaning as dismo `predict(…, tails=)`.
- **When and how to set it.** If all variables are thought to be limiting in one direction only (e.g. minimum temperatures only), `low` or `high` is chosen. The choice is applied to all predictors.
- **Effect on results.** `low` or `high` gives a wider suitable area than `both`.
- **Caution.** The UI has no separate tail choice per predictor.

Comparison with R. Same definition as dismo `bioclim`. The percentile is computed with the mid-rank; because the way dismo handles ties could not be verified, values may differ at the $1/n$ level (`sdm_niche.py:16-18`).

5.20 Domain: Gower distance

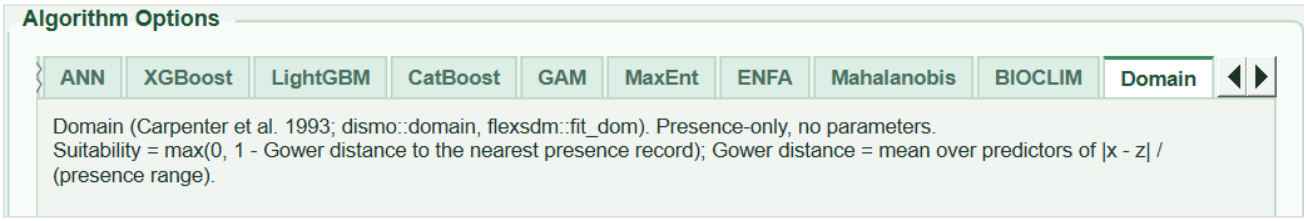


Figure 5.16. Domain tab: method note (Carpenter et al. 1993; dismo::domain, flexsdm::fit_dom); no settings.

Family and implementation. Presence-only similarity method (Carpenter et al. 1993). The Gower distance between a cell and each presence is the mean over predictors of $|x_j - z_j|/\text{range}_j$, where range_j is the range of the presence values. Suitability is $\max(0,\ 1 - \min_z G)$ (`sdm_niche.py:95-127`). A predictor with zero range contributes 0 if equal and 1 otherwise. Categorical variables are left out; at least 2 presences are needed; duplicate presence rows are counted once. It has no settings; the page contains only a description note (`tabs/tab_models.py:917-925`).

> **Caution:** Domain looks at the single nearest presence. A single erroneous or extreme record creates high suitability around it; record cleaning (Chapter 2) is especially important for this method. Computation time increases linearly with the number of presences; memory is limited by processing cells in blocks.

Comparison with R. Same definition as dismo `domain` and flexsdm `fit_dom`; there is no comparison test in HABITUS.

5.21 GLMNET: elastic-net logistic regression

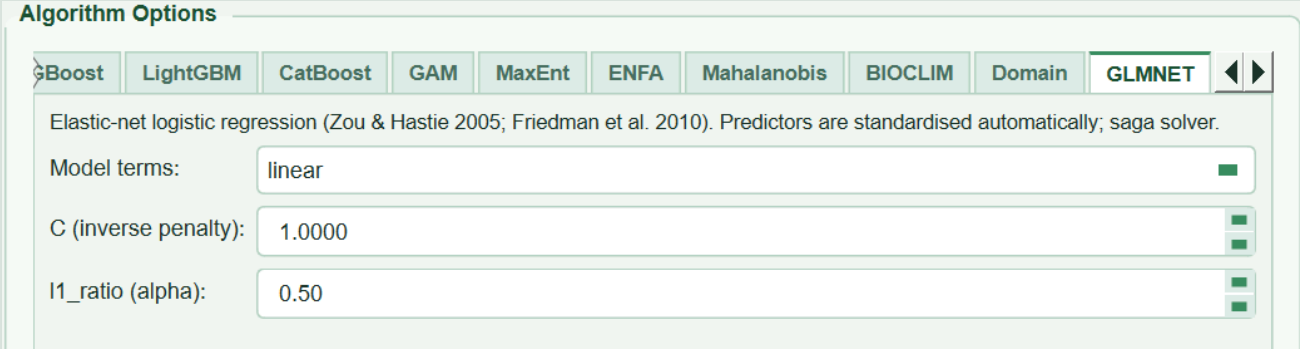


Figure 5.17. GLMNET tab: method note; Model terms linear, C (inverse penalty) 1.0000, l1_ratio (alpha) 0.50.

Family and implementation. Elastic-net logistic regression combining L1 and L2 penalties (Zou and Hastie 2005; Friedman et al. 2010). scikit-learn `LogisticRegression`, `saga` solver, at most 5000 iterations; terms are generated and standardised as in GLM (`sdm_core.py:3446-3461`, `3493-3505`). Presence–pseudo-absence.

Name in the UI	What it does	Default	Range / options	Code
`Model terms:`	Term structure	`linear`	`linear`, `quadratic`, `interactions`	`tabs/tab_models.py:929-934`
`C (inverse penalty):`	Inverse penalty strength	1.0000	0.0001 – 10 000	`tabs/tab_models.py:935-939`
`l1_ratio (alpha):`	L1 / L2 mixture	0.50	0.0 – 1.0 (step 0.1)	`tabs/tab_models.py:940-945`

5.21.1 `C (inverse penalty)`

- **What it does.** If the scikit-learn objective function is divided by n to bring it into glmnet form, $\lambda \approx 1/(C\,n)$; n is the number of training rows.
- **When and how to set it.** In the example run a training fold has about 860 rows; C = 1 means $\lambda \approx 0.0012$ and is a light penalty. For a real choice, the tuning tab tries C = 0.01/0.1/1/10.
- **Effect on results.** As C decreases, the coefficients shrink; if l1_ratio > 0, some predictors drop out entirely.
- **Caution.** R glmnet usually selects λ with `cv.glmnet`; HABITUS uses a single fixed C.

5.21.2 `l1_ratio (alpha)`

- **What it does.** It is glmnet's `alpha`: 0 is ridge (L2 only), 1 is lasso (L1 only; drops variables), in between is elastic net.
- **When and how to set it.** If there are strongly correlated predictors, a value around 0.5 softens the lasso's tendency to pick one of the correlated variables at random.
- **Effect on results.** As it approaches 1, the model becomes sparser and easier to interpret.
- **Caution.** `saga` may converge slowly; with very small C and many terms a convergence warning may appear.

5.21.3 `Model terms`

- **What it does.** The same as GLM (§5.6.1); here the default is `linear` (like the glmnet of the sdm package).
- **When and how to set it.** Because the penalty slows overfitting, `quadratic` or `interactions` can be tried with more confidence than in GLM (tooltip).
- **Effect on results.** As terms increase, flexibility increases; the penalty can set superfluous terms to zero.
- **Caution.** The tuning tab does not change the term structure; the value here stays fixed.

5.22 CART: classification tree

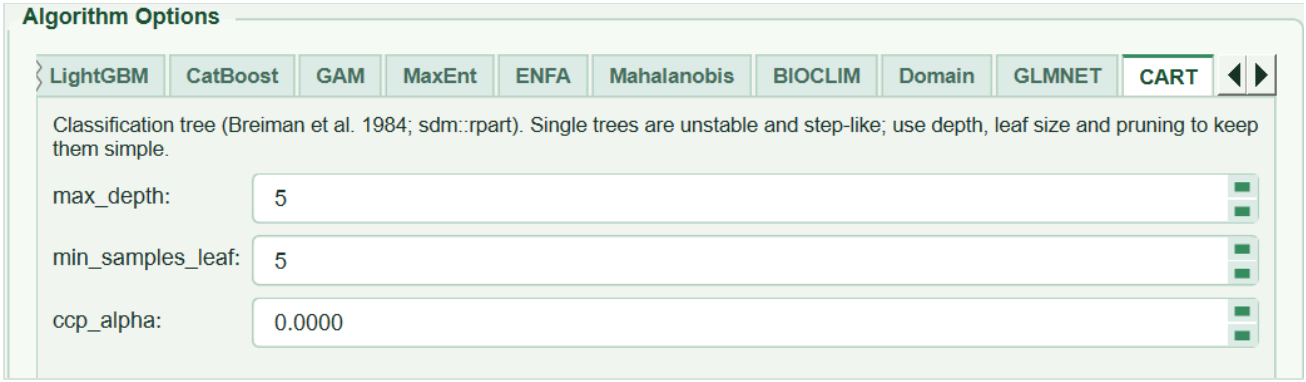


Figure 5.18. CART tab: method note (Breiman et al. 1984; sdm::rpart); max_depth 5, min_samples_leaf 5, ccp_alpha 0.0000.

Family and implementation. A single classification tree (Breiman et al. 1984). scikit-learn `DecisionTreeClassifier`; suitability is the proportion of presences in the leaf into which the cell falls (`sdm_core.py:3506-3515`). Presence–pseudo-absence.

Name in the UI	What it does	Default	Range / options	Code
`max_depth:`	Maximum depth	5	0 (`no limit`) – 30	`tabs/tab_models.py:957-960`; `sdm_core.py:3509-3511`
`min_samples_leaf:`	Minimum rows in a leaf	5	1 – 500	`tabs/tab_models.py:961-963`
`ccp_alpha:`	Cost-complexity pruning	0.0000	0.0 – 0.5	`tabs/tab_models.py:964-967`

5.22.1 `max_depth` and `min_samples_leaf`

- **What it does.** The maximum depth of the tree (0 = unlimited) and the leaf size.
- **When and how to set it.** For explanatory use, a depth of 3–5 gives a readable tree. Because leaf proportions are the probabilities, very small leaves produce a 0/1 surface (tooltip).
- **Effect on results.** As depth increases, the map is divided into more stepped regions and training fit increases.
- **Caution.** A single tree is unstable: when the data change slightly, the structure of the tree can change considerably.

5.22.2 `ccp_alpha`

- **What it does.** Prunes the tree according to the complexity cost of each added leaf; it is the counterpart of rpart's `cp` parameter, but its scale is not the same.
- **When and how to set it.** The tuning tab tries 0/0.002/0.01.
- **Effect on results.** As it grows, the tree becomes smaller.
- **Caution.** rpart prunes with `cp` = 0.01 by default; HABITUS's default is 0 (no pruning), and the limiting is done by depth and leaf size.

5.23 ESM: ensemble of small models

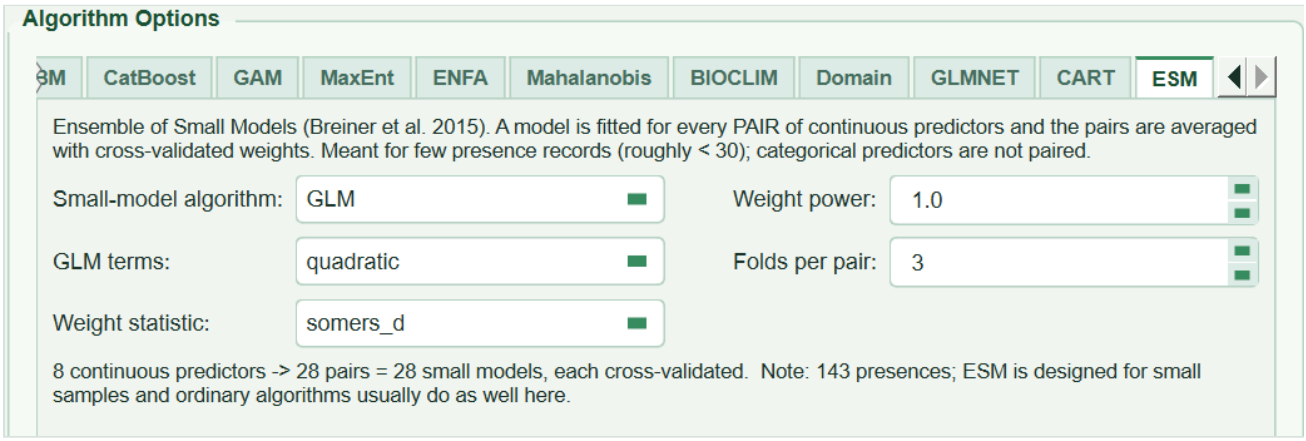


Figure 5.19. ESM tab: method note; Small-model algorithm GLM, GLM terms quadratic, Weight statistic somers_d, Weight power 1.0, Folds per pair 3; at the bottom, the information line for the example run (8 continuous predictors, 28 pairs; note on 143 presences).

Family and implementation. The method recommended for species with very few records (Lomba et al. 2010; Breiner et al. 2015). For each pair of continuous predictors a small bivariate model is fitted; the pairs are weighted by their own cross-validation performance and averaged (`sdm_core.py:4370-4455`, `sdm_niche.py:130-265`). With p predictors there are $p(p-1)/2$ pairs; 28 with 8 predictors. Categorical variables do not enter the pairs. If there are more than 100 pairs, a warning is given (`sdm_niche.py:39`).

Name in the UI	What it does	Default	Range / options	Code
`Small-model algorithm:`	Algorithm of the small models	`GLM`	`GLM`, `RF`, `MAXENT`	`tabs/tab_models.py:979-983`; `sdm_core.py:4380-4385`, `4407-4414`
`GLM terms:`	Terms of the GLM small models	`quadratic`	`quadratic`, `linear`	`tabs/tab_models.py:1004-1005`; `sdm_core.py:4409`
`Weight statistic:`	Criterion for the pair weight	`somers_d`	`somers_d`, `auc`, `boyce`	`tabs/tab_models.py:984-991`; `sdm_niche.py:142-157`
`Weight power:`	Exponent of the weight	1.0	0.0 – 10.0	`tabs/tab_models.py:992-996`
`Folds per pair:`	Number of folds that score each pair	3	2 – 10	`tabs/tab_models.py:997-1003`; `sdm_core.py:4296-4368`

The information line at the bottom of the page computes the number of pairs from the selected predictors; with 50 or more presences it reminds the user that ESM is designed for small samples (`tabs/tab_models.py:1022-1044`).

5.23.1 `Small-model algorithm` and `GLM terms`

- **What it does.** The model of each pair. GLM is fitted with the same settings as the main GLM (C = 1) and the terms selected here; RF with 200 trees and leaf size 2 (cannot be changed in the UI); MaxEnt with the settings in the MaxEnt tab and the relevant two columns of the MaxEnt background (`sdm_core.py:4407-4445`).
- **When and how to set it.** A linear + quadratic GLM is the usual choice (Breiner et al. 2015). RF is slower, MaxEnt much slower. The default of flexsdm's `esm_glm` is the linear term only; for comparison with it, `linear` is chosen.
- **Effect on results.** Quadratic terms allow an optimum response in each pair.
- **Caution.** `Balance classes` is applied to the small models only with the GLM and RF bases; with the MaxEnt base it is ignored and "not supported (ignored)" is written to the record (`sdm_core.py:4402-4406`).

5.23.2 `Weight statistic` and `Weight power`

- **What it does.** The weight of each pair is the criterion computed from the pair's out-of-fold predictions raised to the power `power`. `somers_d` = $\max(2(\mathrm{AUC} - 0.5),\ 0)$; `auc` = raw AUC; `boyce` = $\max(\mathrm{Boyce},\ 0)$. A pair no better than chance gets weight 0 with `somers_d` and `boyce`.
- **When and how to set it.** Breiner et al. (2015) use Somers' D. `auc` weights are almost equal; to make them discriminating, the exponent should be chosen greater than 1. An exponent of 0 means equal weights.
- **Effect on results.** As the exponent grows, the ensemble concentrates on the best few pairs.
- **Caution.** If no pair is better than chance, equal weights are used and the warning "No small model beat chance in cross-validation; the ensemble uses equal weights." is given (`sdm_niche.py:260-263`).

5.23.3 `Folds per pair`

- **What it does.** The number of inner folds in which the pairs are scored. The folds apply the main cross-validation design (spatial block, environmental, band, checkerboard) to the rows on which ESM is trained; if the design cannot be applied or the design is `random` / `jackknife`, random stratified folds are used (`sdm_core.py:4296-4368`). The number of folds cannot exceed the number of presences.
- **When and how to set it.** With very few presences (10–20) 2–3 folds; with more presences 4–5.
- **Effect on results.** As folds increase, the weights become more stable and run time longer.
- **Caution.** When the design cannot be applied, the warning "ESM: the … design could not be applied to the folds that weight the small models (…); random stratified folds were used for them." is given once.

Comparison with R. ecospat `ecospat.ESM.Modeling` implements the same idea with biomod2 models. HABITUS's implementation is its own; there is no numerical comparison test against ecospat.

5.24 Outputs and what passes to the next step

- **Saved settings:** all values on this page (including those of unticked algorithms) go to the `habitus_run_config.json` → `settings.models.algorithm_options` field; the settings actually passed to MaxEnt (`maxent_settings_used`: elapid arguments, seed, number of presences added to the background, solver) are written separately. If ESM was run, the number of pairs, the weights and the ten best pairs are written to the `esm` field (`sdm_core.py:4457-4478`). How class balancing was applied to each algorithm is in the `class_weight_applied` field.
- **Models:** each algorithm × PA set × fold and the full model; scores in the `Evaluation Results` sub-tab of ③, response curves, importance plots and ROC curves in ④ Evaluation.
- **Maps:** current suitability per algorithm and the ensemble map (`Current Distribution Map`), projection with the same models in ⑤ Future Scenarios.

5.25 Expert questions

Q1. Which of the eighteen algorithms should I choose; why are only GLM and RF the default? Answer: At start-up the program ticks only GLM and RF (`tabs/tab_models.py:129-132`): one an interpretable parametric regression, the other a flexible tree ensemble; neither requires an additional package and both are fast. The code contains no definitive rule for the choice. A typical study runs several methods from different families (GLM or GAM, RF or BRT, MaxEnt) and combines them in an ensemble map. With very few records (below about 30), ESM, GLMNET or MaxEnt with linear/quadratic features; where there are only presence records and working with pseudo-absences is not wanted, MaxEnt, ENFA, Mahalanobis, BIOCLIM and Domain are considered. Valavi et al. (2022) examine comparative performance on presence-only data across many species; their results cannot be transferred directly to HABITUS's implementations, but they can still be consulted for the general behaviour of method families.

Q2. Where do the default values come from? Answer: From three sources. (1) Library defaults: RF (500 trees, `sqrt`, leaf 1; the same as randomForest), SVM, ANN's alpha, GAM (20 splines, lam 0.6; pygam), XGBoost's penalties. (2) Publications: BRT learning rate and bagging fraction from Elith et al. (2008) (`sdm_core.py:3530-3548`), MaxEnt number of hinges and β = 1 from maxnet, ESM from Breiner et al. (2015). (3) HABITUS's own paired cross-validation comparisons: GBM's stumps and BRT's depth of 2 (code comments `sdm_core.py:3517-3522`, `3532-3537`), GLM's C = 1 penalty (tooltip, `tabs/tab_models.py:741-745`). The details of these comparisons are in the internal notes referred to in the code comments; they are not the subject of this book. In a publication, settings should be reported not as "HABITUS defaults" but with the numbers in `habitus_run_config.json`.

Q3. Does HABITUS's MaxEnt give the same result as R maxnet? Answer: The objective function is the same: per-feature L1 penalty, weights of presence 1 / background 100, last λ, addition of presences to the background (`sdm_core.py:3776-3836`, `4698-4707`). `tests/test_maxent_maxnet.py` checks, on the data of Rutrough et al. (2019) at three settings, that the correlation with R maxnet 0.1.4 predictions exceeds 0.995 and that the mean absolute difference stays below 0.02. The remaining differences come from the solver (liblinear instead of glmnet's coordinate descent) and from the very small but non-zero penalty on the intercept. Feature generation belongs to elapid; the placement of hinge knots by elapid and maxnet agrees on this test dataset but has not been tested separately for every dataset. For an exact comparison with maxnet, `Auto (maxnet default rule)` should be ticked; HABITUS's default `linear,hinge,product` set is not the same as the maxnet rule.

Q4. Why is adding presences to the background important in MaxEnt? Answer: This is the default behaviour of Maxent and maxnet; the environment where a presence occurs is also part of the available

environment. If they are not added, the environment of presence cells is under-represented in the background distribution and the model's raw output and normalisation constant diverge from maxnet. HABITUS adds only presences whose predictor values are not already in the background (`sdm_core.py:4702-4706`); in the full model of the example run, 91 of the 143 presences were added.

Q5. Why are presence-only methods (MaxEnt, ENFA, Mahalanobis, BIOCLIM, Domain) evaluated with pseudo-absences? Answer: HABITUS scores all algorithms in the same test fold with the same presence and pseudo-absence rows (`sdm_core.py:5530-5540`); in this way the scores are comparable across algorithms and the ensemble weights are consistent. In this case AUC is the power to "separate presence from pseudo-absence", not performance against true absence; the Boyce index, which relies on presences only, is a more suitable measure for presence-only data. If no presence remains in the test fold, one fifth of the training presences is moved to the test set (`sdm_core.py:5455-5468`); in that case the fold differs from the fold of the other algorithms. A user who wants to evaluate presence-only models separately should focus on the Boyce and OR_10p columns.

Q6. Should class balancing (`Balance classes`) be turned on; what does prevalence change? Answer: The default is off. In the example run the pseudo-absence : presence ratio is 7.0 : 1 (`pa_to_presence_ratio` = 6.993). When balancing is turned on, the total weights of the two classes are equalised in the supported algorithms (§5.4.5). This changes the meaning of the predicted value: the output no longer reflects the pseudo-absence ratio, thresholds are re-derived; ranking measures (AUC) are mostly little affected (tooltip, `tabs/tab_models.py:346-348`). Because the number of pseudo-absences is arbitrary anyway, balancing is not required if absolute "probability" is not interpreted. What matters is that balancing is not applied at all in GAM, MaxEnt and the presence-only methods; balanced and unbalanced models can be mixed in one ensemble. A breakdown of how it was applied is in the `class_weight_applied` field.

Q7. Are the results reproducible; which sources add randomness? Answer: Yes, with a single seed. All random learners are given the seed from ① Data (`sdm_core.py:3474-3475`); `random_state` is also passed to MaxEnt (`sdm_core.py:4643`), and permutation importance is computed with a generator derived from the seed (`sdm_core.py:4847-4853`). The settings that add randomness are RF's bootstrap, the subsampling of GBM/BRT/XGBoost/LightGBM, CatBoost's `random_strength` and `bagging_temperature`, ANN's initial weights, SVM's Platt folds and ESM's inner folds. Multithreaded libraries (RF, XGBoost, LightGBM `n_jobs = -1`) may give differences in the last digits on a different number of processors because of the floating-point summation order; this has not been tested in the code. Library versions are recorded in the `habitus_run_config.json` → `environment` field; the same versions are needed for the same result.

Q8. Why is GLM penalised; how do I compare my result with glm in R? Answer: Quadratic and interaction terms inflate the coefficients with few records; HABITUS applies a light L2 penalty to the standardised terms (`sdm_core.py:3481-3492`). R `glm()` is unpenalised. For comparison, enter `Regularisation C` = 1e6; the term structure is built in R with `y ~ x + I(x^2)` (quadratic) or `y ~ poly(x1, x2, degree = 2, raw = TRUE)` (interactions). biomod2's GLM additionally performs stepwise selection by AIC; HABITUS has no stepwise selection, so the coefficients will not be exactly the same as biomod2's.

Q9. Why can GAM results differ from mgcv? Answer: HABITUS fits pygam `LogisticGAM` with a fixed smoothing penalty (`lam`, default 0.6) and 20 B-spline bases per predictor (`sdm_core.py:4500-4523`). mgcv, in contrast, selects the penalty from the data (GCV/REML) and uses a different basis. On the same data mgcv may choose a smoother or a more wiggly curve. To bring them closer, `lam` can be searched in the tuning tab; but the two models will still not be identical. GAM has no interaction terms and no class balancing.

Q10. Does BRT select the number of trees itself like dismo `gbm.step`? Answer: No. `n_estimators` is fixed (BRT 1000, GBM 200) and early stopping is not used (`sdm_core.py:3523-3548`). dismo `gbm.step` stops the number of trees where the 10-fold cross-validation deviance is smallest. For a similar result in HABITUS, an

`n_estimators` × `learning_rate` grid is searched in the GBM tuning tab and the selected value is written into BRT by hand. Also, whereas Elith et al. used `tree.complexity` = 5, HABITUS BRT's depth is 2.

Q11. Are the defaults of XGBoost, LightGBM and CatBoost suitable for SDM? Answer: The defaults of these three (XGBoost depth 6 and 500 rounds; LightGBM 31 leaves, unlimited depth and 500 rounds; CatBoost depth 4 and 300 rounds) are much more complex models than GBM and BRT (`sdm_core.py:3586-3647`). The comparison table in the help tab notes at least 100 records and a tendency to overfit for them. With data of fewer than a few hundred presences it is recommended to reduce depth to 2–3 and the number of leaves to 7–15 and to check the result with the training–test AUC difference (dAUC). Note that the `subsample` setting has no effect in LightGBM (§5.13.3).

Q12. When I add a categorical predictor (e.g. land cover), does every algorithm use it in the same way? Answer: No (§5.4.3). Regression and kernel methods use one 0/1 column per class, GAM a factor term, MaxEnt a categorical feature; tree methods handle the integer code with numeric splits; envelope and distance methods leave the variable out and write a note to the log (`sdm_core.py:4010-4036`). So some of the models in an ensemble may not see the categorical variable at all. A class not seen in training (present only in the projection area) is treated as "no class effect" in regression methods, GAM and MaxEnt, and in trees it is evaluated in the branch into which its numeric code falls.

Q13. Why are ENFA and Mahalanobis scores rescaled to the range 0–1; is the map comparable? Answer: The raw score $\exp(-D^2/2)$ approaches zero quickly with many predictors. HABITUS stores the score range of the training rows during fitting and uses the same range in every prediction (`sdm_core.py:3719-3743`). Therefore a cell gets the same value in the test fold and in the projection, and the threshold has the same meaning in both settings. Even so, the scale of the score is specific to the algorithm; 0.6 from ENFA and 0.6 from GLM are not the same probability. Ensemble weights and binary thresholds are therefore determined separately per algorithm (Chapter 6).

Q14. How should I report the tuned parameters in a publication? Answer: `algorithm_options` (the values entered), `maxent_settings_used` (what was actually passed to MaxEnt, and the solver), `class_weight`, `clamp`, `random_seed` and `environment` (library versions) in `habitus_run_config.json` should be reported together. If the tuning tab was used, the grid, the selection rule and the selected point should also be stated; that the score may be optimistic when tuning and evaluation are done on the same folds is discussed in Chapter 6. The ODMAP report (Chapter 11) carries most of this information automatically.

5.26 Common mistakes and pitfalls

- **Package not installed:** the box is grey with `[not installed]` appended. Example tooltip: "Not installed — pip install elapid". After installation the program must be restarted.
- **Running without selecting any algorithm:** "Select at least one algorithm."
- **Writing the ANN layers incorrectly** (e.g. `100;50`, `100 50`, `0`): "ANN hidden layers: … is not a list of positive whole numbers separated by commas (e.g. 100, 50)."
- **Writing `auto` or `%50` in RF `max_features`:** "RF max_features: … is not sqrt, log2, a whole number of variables or a fraction between 0 and 1." The fraction should be written as 0.5.
- **GAM `n_splines` ≤ 3:** the UI lower limit is 4; if given by script, "GAM n_splines must exceed the spline order (3)."
- **Misspelling a MaxEnt feature name** (e.g. `hinges`, `quad`): elapid rejects the setting and MaxEnt fails with the message "MaxEnt could not be fitted…" or "MaxEnt could not be configured with the settings you chose…". Valid names: `linear`, `quadratic`, `hinge`, `product`, `threshold`.
- **Changing the text box in MaxEnt while `Auto` is ticked:** the box is disabled and ignored.
- **Changing `N threshold features` without selecting the `threshold` class:** the value is not used.
- **Giving categorical variables to presence-only methods:** the variable is left out, not silently, but with the message "NOTE: … is an envelope/distance model for continuous predictors…" in the log; if there are only categorical variables, the error "… needs at least one continuous predictor (only categorical ones were given)." is given.
- **Running ESM with many predictors:** 15 predictors means 105 pairs and 105 separate cross-validated models; the information line says "WARNING: more than 100 pairs…".
- **Using ESM for a species with many records:** with 50 or more presences the information line reminds the user that ESM is designed for small samples; the usual algorithms are generally as good.
- **`LinAlgError` in Mahalanobis:** increase ε or keep Ledoit–Wolf on; strongly correlated predictors are removed in ② Variables.
- **Assuming that `Balance classes` affects every algorithm:** it has no effect in GAM, MaxEnt, ENFA, Mahalanobis, BIOCLIM and Domain.
- **Adjusting LightGBM `subsample`:** because row subsampling is not active in HABITUS, the value does not change the result.
- **Using the default deep boosting models with few records:** in XGBoost/LightGBM, training AUC approaches 1 while test AUC falls; the dAUC column should be monitored.

5.27 References

Anderson, C.B., 2023. elapid: Species distribution modeling tools for Python. Journal of Open Source Software 8, 4930. https://doi.org/10.21105/joss.04930

Breiman, L., 2001. Random forests. Machine Learning 45, 5–32. https://doi.org/10.1023/A:1010933404324

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J., 1984. Classification and Regression Trees. Wadsworth, Belmont (new edition: Routledge, 2017). https://doi.org/10.1201/9781315139470

Breiner, F.T., Guisan, A., Bergamini, A., Nobis, M.P., 2015. Overcoming limitations of modelling rare species by using ensembles of small models. Methods in Ecology and Evolution 6, 1210–1218. https://doi.org/10.1111/2041-210X.12403

Carpenter, G., Gillison, A.N., Winter, J., 1993. DOMAIN: a flexible modelling procedure for mapping potential distributions of plants and animals. Biodiversity and Conservation 2, 667–680. https://doi.org/10.1007/BF00051966

Chen, T., Guestrin, C., 2016. XGBoost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794. https://doi.org/10.1145/2939672.2939785

Elith, J., Leathwick, J.R., Hastie, T., 2008. A working guide to boosted regression trees. Journal of Animal Ecology 77, 802–813. https://doi.org/10.1111/j.1365-2656.2008.01390.x

Farber, O., Kadmon, R., 2003. Assessment of alternative approaches for bioclimatic modeling with special emphasis on the Mahalanobis distance. Ecological Modelling 160, 115–130. https://doi.org/10.1016/S0304-3800(02)00327-7

Friedman, J., Hastie, T., Tibshirani, R., 2010. Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software 33(1), 1–22. https://doi.org/10.18637/jss.v033.i01

Hastie, T., Tibshirani, R., 1986. Generalized additive models. Statistical Science 1, 297–310. https://doi.org/10.1214/ss/1177013604

Hirzel, A.H., Hausser, J., Chessel, D., Perrin, N., 2002. Ecological-niche factor analysis: how to compute habitat-suitability maps without absence data? Ecology 83, 2027–2036. https://doi.org/10.1890/0012-9658(2002)083[2027:ENFAHT]2.0.CO;2

Ledoit, O., Wolf, M., 2004. A well-conditioned estimator for large-dimensional covariance matrices. Journal of Multivariate Analysis 88, 365–411. https://doi.org/10.1016/S0047-259X(03)00096-4

Lomba, A., Pellissier, L., Randin, C., Vicente, J., Moreira, F., Honrado, J., Guisan, A., 2010. Overcoming the rare species modelling paradox: a novel hierarchical framework applied to an Iberian endemic plant. Biological Conservation 143, 2647–2657. https://doi.org/10.1016/j.biocon.2010.07.007

Merow, C., Smith, M.J., Silander, J.A., 2013. A practical guide to MaxEnt for modeling species' distributions: what it does, and why inputs and settings matter. Ecography 36, 1058–1069. https://doi.org/10.1111/j.1600-0587.2013.07872.x

Muscarella, R., Galante, P.J., Soley-Guardia, M., Boria, R.A., Kass, J.M., Uriarte, M., Anderson, R.P., 2014. ENMeval: An R package for conducting spatially independent evaluations and estimating optimal model complexity for Maxent ecological niche models. Methods in Ecology and Evolution 5, 1198–1205. https://doi.org/10.1111/2041-210X.12261

Nix, H.A., 1986. A biogeographic analysis of Australian elapid snakes. In: Longmore, R. (Ed.), Atlas of Elapid Snakes of Australia. Australian Flora and Fauna Series 7. Australian Government Publishing Service, Canberra, 4–15.

Oshiro, T.M., Perez, P.S., Baranauskas, J.A., 2012. How many trees in a random forest? In: Perner, P. (Ed.), Machine Learning and Data Mining in Pattern Recognition. Lecture Notes in Computer Science 7376. Springer, Berlin, 154–168. https://doi.org/10.1007/978-3-642-31537-4_13

Phillips, S.J., Anderson, R.P., Schapire, R.E., 2006. Maximum entropy modeling of species geographic distributions. Ecological Modelling 190, 231–259. https://doi.org/10.1016/j.ecolmodel.2005.03.026

Phillips, S.J., Anderson, R.P., Dudík, M., Schapire, R.E., Blair, M.E., 2017. Opening the black box: an open-source release of Maxent. Ecography 40, 887–893. https://doi.org/10.1111/ecog.03049

Rutrough, A., Widick, I.V., Bean, W.T., 2019. Reconstruction of the historical range alters niche estimates in an endangered rodent. Ecography 42, 1742–1751. https://doi.org/10.1111/ecog.04238

Valavi, R., Elith, J., Lahoz-Monfort, J.J., Guillera-Arroita, G., 2021. Modelling species presence-only data with random forests. Ecography 44, 1731–1742. https://doi.org/10.1111/ecog.05615

Valavi, R., Guillera-Arroita, G., Lahoz-Monfort, J.J., Elith, J., 2022. Predictive performance of presence-only species distribution models: a benchmark study with reproducible code. Ecological Monographs 92, e01486. https://doi.org/10.1002/ecm.1486

Velazco, S.J.E., Rose, M.B., de Andrade, A.F.A., Minoli, I., Franklin, J., 2022. flexsdm: An R package for supporting a comprehensive and flexible species distribution modelling workflow. Methods in Ecology and Evolution 13, 1661–1669. https://doi.org/10.1111/2041-210X.13874

Zou, H., Hastie, T., 2005. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society Series B 67, 301–320. https://doi.org/10.1111/j.1467-9868.2005.00503.x

6. Models II: cross-validation, threshold, ensemble and results (③ Models)

Tab: ③ Models + Current Map · Code: `tabs/tab_models.py`, `tabs/tab_tuning.py` (UI); `sdm_core.py` `SDMModeler`, `EnsembleModeler`, `Projector`, `evaluate_all`; `sdm_spatial.py` (blocks, variogram, folds, SSB); `sdm_tuning.py` (grid search)

6.1 What this tab does

The ③ Models tab fits the selected algorithms with the predictors chosen in the ② Variables step, tests each of them by cross-validation (CV), projects the models refitted on all data onto the current climate layers, and combines them into an ensemble map. Chapter 5 describes the algorithms' own options (`Algorithm Options`); this chapter describes everything else on the same tab: which algorithms run, how the data are split into folds, how the binary (presence/absence) threshold is chosen, how the ensemble is built, the optional tuning search, and the result sub-tabs.

- **Input:** the pseudo-absence (PA) sets produced in ① Data, the continuous and categorical predictors confirmed in ② Variables, the training rasters and the seed. If these are missing, the button gives the message `"Complete ① Data and ② Variables steps first."` (`tabs/tab_models.py:1394-1395`).
- **Output:** evaluation scores for each algorithm × PA set × CV run; probability and binary maps of the models fitted on all data (`Full`); mean and spread maps per algorithm; `EM*` ensemble maps; `evaluation_scores.csv`, `habitus_run_config.json`, training/test point files (§6.10).
- **Lock order:** when the run finishes, the ④ Evaluation, ⑤ Future Scenarios, ⑥ Range Change, ⑦ Validation and ⑧ Report tabs are unlocked (`tabs/tab_models.py:1848-1852`).

A single button, `▶ Run Models + Generate Current Distribution Map`, does four jobs in sequence (`tabs/tab_models.py:1559-1659`): (1) CV and full-data models with `SDMModeler.run()`, (2) filtering of ensemble members with `EnsembleModeler`, (3) projection onto the current climate and writing of the maps with `Projector.project()`, (4) writing of the run record (`habitus_run_config.json`).

6.2 Screen layout

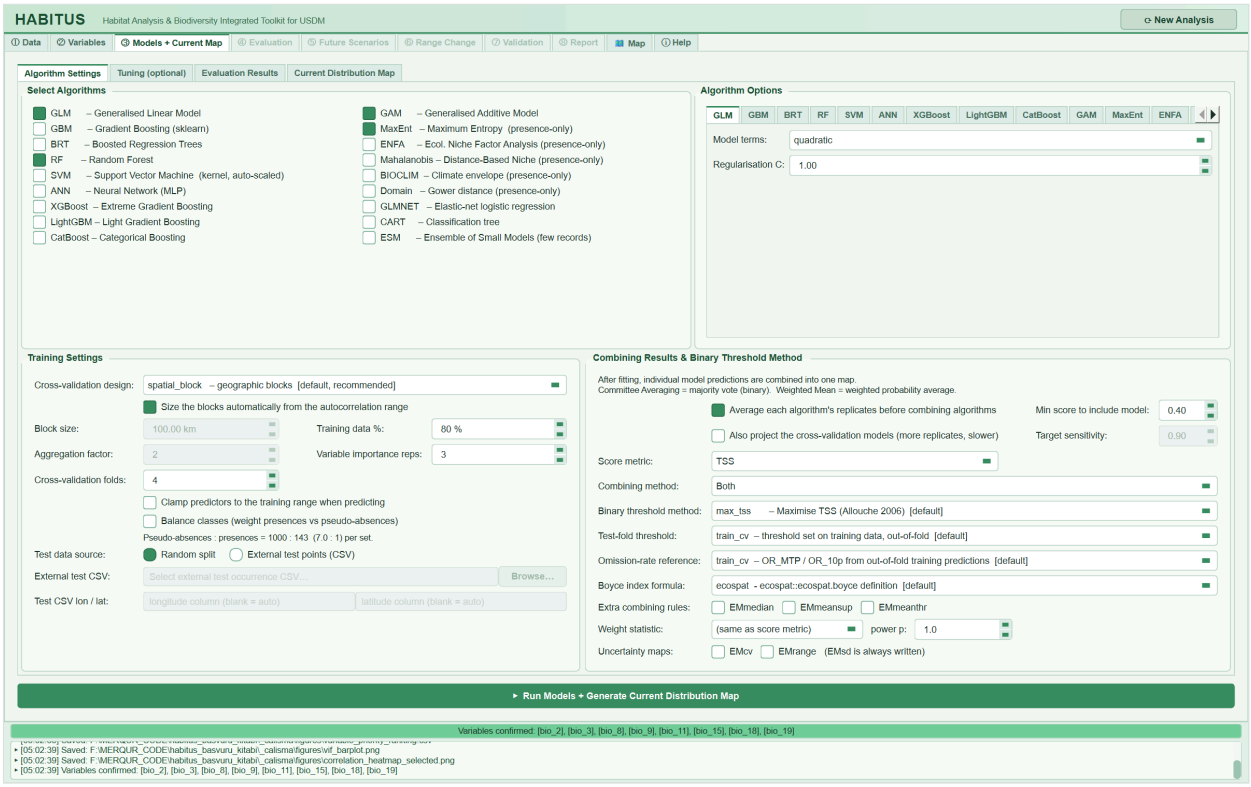


Figure 6.1. The ③ Models + Current Map tab before a run: four sub-tabs at the top, Select Algorithms on the left, Algorithm Options on the right, the Training Settings and Combining Results && Binary Threshold Method boxes below, and the run button at the very bottom.

Figure 6.1. The ③ Models + Current Map tab before a run: four sub-tabs at the top, Select Algorithms on the left, Algorithm Options on the right, the Training Settings and Combining Results && Binary Threshold Method boxes below, and the run button at the very bottom.

The tab has four sub-tabs and a single run button that stays below all of them (`tabs/tab_models.py:91-110`):

Region	Content	In this chapter
`Algorithm Settings`	`Select Algorithms`, `Algorithm Options`, `Training Settings`, `Combining Results && Binary Threshold Method`	§6.3–6.6 (algorithm options: Chapter 5)
`Tuning (optional)`	Grid search for a single algorithm	§6.7
`Evaluation Results`	ROC-AUC, TSS, Boyce table per model	§6.8
`Current Distribution Map`	Status text, table of produced maps, output folder, map viewer	§6.9
Bottom strip	`▶ Run Models + Generate Current Distribution Map`	§6.1

When the run starts, the program switches to the `Evaluation Results` sub-tab, and when it finishes it opens the `Current Distribution Map` sub-tab (`tabs/tab_models.py:1491`, `1835`).

6.3 Algorithm selection (Select Algorithms)

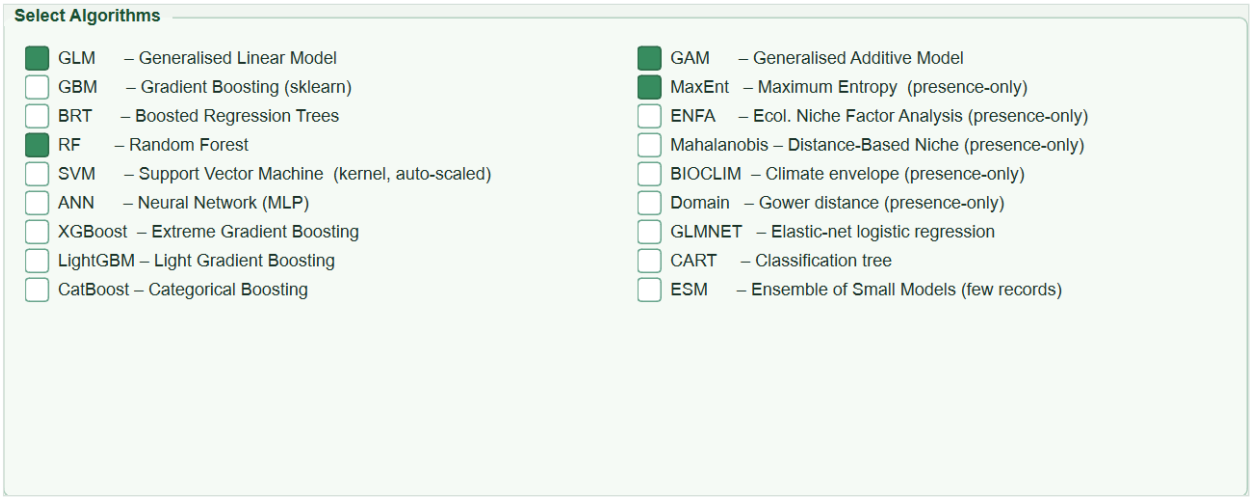


Figure 6.2. The Select Algorithms box: eighteen algorithms in two columns; GLM, RF, GAM and MaxEnt are ticked in the example run.

Figure 6.2. The Select Algorithms box: eighteen algorithms in two columns; GLM, RF, GAM and MaxEnt are ticked in the example run.

The box contains eighteen check boxes; each adds one algorithm to the run list (`tabs/tab_models.py:125-150`). Every box that is ticked **and** enabled runs (`tabs/tab_models.py:1397-1398`). If none is selected, the message `"Select at least one algorithm."` appears.

Name in the UI (short)	What it does	Default	Code
`GLM`, `RF`	Generalized linear model; random forest	on	`tabs/tab_models.py:129-132`
`GBM`, `BRT`, `SVM`, `ANN`	Gradient boosting (sklearn), boosted regression trees, support vector machine, multilayer neural network	off	`tabs/tab_models.py:130-134`
`XGBoost`, `LightGBM`, `CatBoost`	Three boosting libraries; if not installed the box is greyed out and shows `[not installed]`	off	`tabs/tab_models.py:135-137`, `152-168`
`GAM`	Generalized additive model (requires pygam)	off	`tabs/tab_models.py:138`
`MaxEnt`, `ENFA`, `Mahalanobis`, `BIOCLIM`, `Domain`	Presence-only methods	off	`tabs/tab_models.py:139-143`
`GLMNET`, `CART`, `ESM`	Elastic-net logistic regression, classification tree, ensemble of small models	off	`tabs/tab_models.py:144-146`

What it does. Each selected algorithm is fitted on every PA set, in every CV fold, and once more on all data (`sdm_core.py:5409-5643`, `5654-5768`). The ensemble map consists only of the selected algorithms (`sdm_core.py:7844-7855`).

When and how to set it. The ensemble approach envisages combining algorithms with different biases (Araújo and New 2007); adding algorithms that are very similar to one another (e.g. GBM, BRT, XGBoost and LightGBM all together) weights the ensemble towards a single family. Two-level combining (§6.6.1) removes

this effect with respect to the number of replicates, but not with respect to the number of families. For species with few records (roughly fewer than 30 presences), ESM and the presence-only methods are meaningful (Chapter 5).

Effect on results. Run time grows linearly with the number of algorithms: the total number of fits is roughly algorithms × PA sets × (folds + 1); when `Test-fold threshold` = `train_cv`, three inner fits are added to every CV fit (§6.5.2). In the example run the MaxEnt full model alone took 25–28 s (session log).

Caution. The default the program loads itself is GLM and RF only; the four algorithms in the figure were ticked by hand for the example run.

6.4 Training settings (Training Settings)

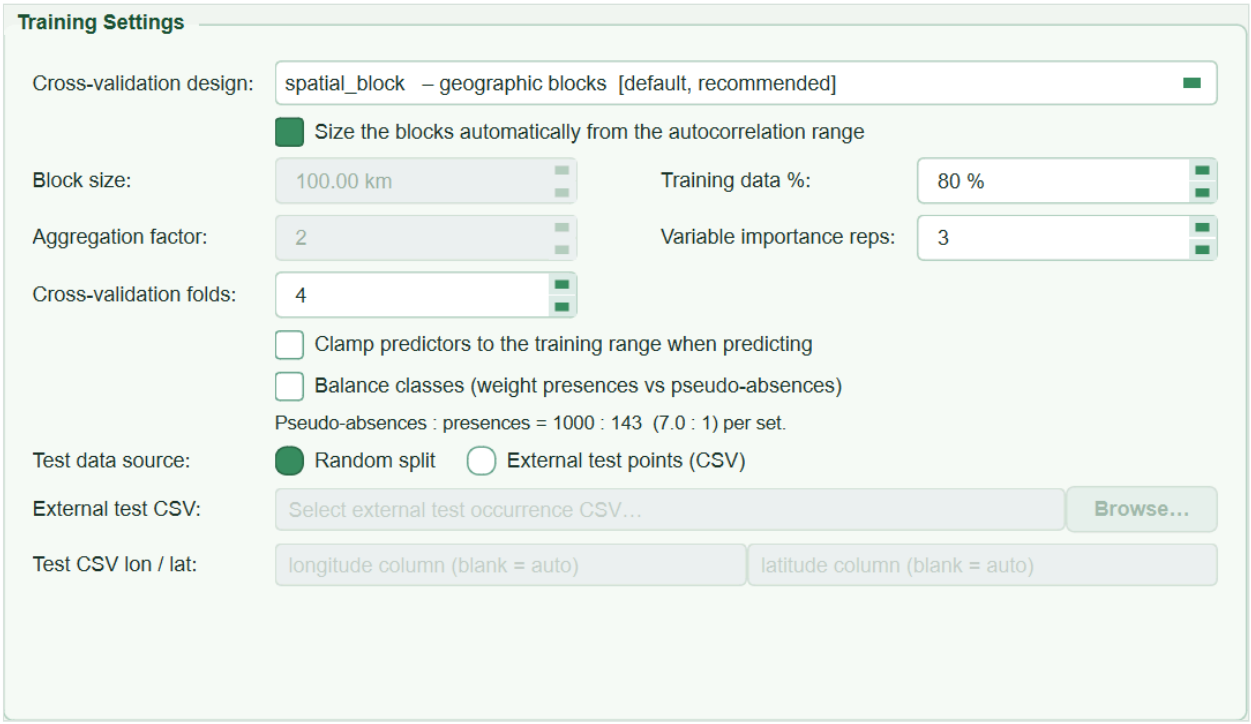


Figure 6.3. The Training Settings box with default values: spatial_block design, automatic block size (Block size and Aggregation factor greyed out), 4 folds, Random split, and the Clamp/Balance boxes off.

Figure 6.3. The Training Settings box with default values: spatial_block design, automatic block size (Block size and Aggregation factor greyed out), 4 folds, Random split, and the Clamp/Balance boxes off.

This box determines how the data are split into training and test parts. HABITUS produces its scores from the held-out part of the cross-validation folds; the choice of design therefore directly affects all reported scores.

Name in the UI	What it does	Default	Range / options	Code
`Cross-validation design:`	Fold design	`spatial_block`	8 designs (§6.4.1)	`tabs/tab_models.py:219-255`
`Size the blocks automatically from the autocorrelation range`	Takes the block size from the variogram range	on	on / off	`tabs/tab_models.py:265-271`

Name in the UI	What it does	Default	Range / options	Code
`Block size:`	Manual block side length	100 km (only when automatic is off)	1–5000 km	`tabs/tab_models.py:273-279`
`Aggregation factor:`	Checkerboard cell multiplier	2	1–20 (only `checkerboard*`)	`tabs/tab_models.py:256-263`
`Cross-validation folds:`	Number of folds	4	1–10 (greyed out for `jackknife`)	`tabs/tab_models.py:211-216`
`Clamp predictors to the training range when predicting`	Clamps predictors to the training range	off	on / off	`tabs/tab_models.py:354-364`
`Balance classes (weight presences vs pseudo-absences)`	Weights the classes inversely to their size	off	on / off	`tabs/tab_models.py:333-352`
`Test data source:`	Source of the test data	`Random split`	`Random split` / `External test points (CSV)`	`tabs/tab_models.py:294-299`
`Training data %:`	Training share in the fallback evaluation split	80 %	50–95 %	`tabs/tab_models.py:301-303`
`External test CSV:`	External test file	empty	CSV file	`tabs/tab_models.py:305-311`
`Test CSV lon / lat:`	Coordinate columns in the external test file	empty (automatic)	column name	`tabs/tab_models.py:313-320`
`Variable importance reps:`	Number of permutation-importance repeats	3	0–10	`tabs/tab_models.py:290-291`

The enabling rules are in the `_toggle_block` function (`tabs/tab_models.py:281-289`): the block boxes are enabled only for `spatial_block`, `checkerboard` and `checkerboard2`; `Aggregation factor` only in the checkerboard designs; `Cross-validation folds` in every design except `jackknife`.

6.4.1 Cross-validation design (Cross-validation design)

What it does. For each PA set it assigns a fold number to every row (`SDMModeler._build_folds`, `sdm_core.py:4997-5162`). Fold k becomes the test set in turn and the remaining folds form the training set (`sdm_core.py:5429-5450`). There are eight designs:

Option	How the folds are formed	Code
`spatial_block`	A square grid is laid over the study area; each block is assigned as a whole to one fold. Blocks are distributed by the best of 50 seeded trials so that the numbers of presences and absences in each fold are balanced	`sdm_spatial.py:385-511`, `305-357`
`environmental`	Predictors are standardized and split into `folds` clusters with k-means; each cluster is one fold	`sdm_spatial.py:514-558`
`random`	Random folds stratified by class	`sdm_spatial.py:561-580`
`checkerboard`	Checkerboard with cell side = block size × `Aggregation factor`; two colours, two folds	`sdm_spatial.py:623-706`
`checkerboard2`	Two checkerboards, fine (block size) and coarse (block size × factor); fold = 2 × coarse colour + fine colour, four folds	`sdm_spatial.py:666-672`
`jackknife`	Each presence is its own fold; absences are distributed equally and randomly among the folds	`sdm_spatial.py:709-755`
`lon_bands` / `lat_bands`	`folds` bands along longitude or latitude with equal numbers of presences	`sdm_spatial.py:758-790`

In addition, if `Test data source:` = `External test points (CSV)` is selected, the folds are not used for testing (§6.4.5).

When and how to set it. With clustered records, a random split leaves a training point right next to a test point carrying almost the same predictor values, and inflates the scores (Roberts et al. 2017). This is why the default is `spatial_block`. If projections to future climate are planned, `environmental` tests transfer to conditions the model has not seen more directly. `checkerboard` and `checkerboard2` correspond to the partitions of the same name in ENMeval (Muscarella et al. 2014): test and training points are interleaved throughout the area. `jackknife` is for species with fewer than about 25 presences (`sdm_spatial.py:594`; Pearson et al. 2007). Bands test transfer along a latitudinal or longitudinal gradient. `random` should be kept only for reproducing an earlier result or for comparison.

Effect on results. The design determines every reported score (AUC, TSS, Boyce), the members entering the ensemble (§6.6.4) and the members' weights, because the score of the `Full` models is the mean of the CV folds (`sdm_core.py:5818-5853`). Blocked designs give lower but more honest scores than a random split.

Caution.

- If the design cannot be applied (too few blocks, folds that do not contain both classes, no coordinates), the program falls back to random folds and writes this to the log and the report with the note `"WARNING (design not applied): …"`; the summary line shows `[BLOCKING FAILED - scores are optimistic]` (`sdm_core.py:5135-5140`, `5933-5934`).
- Folds containing no presences or no absences are merged into another fold; in this case fewer folds than requested are evaluated and the message `"WARNING: … requested folds held no presence or no absence and were merged into others …"` is given (`sdm_core.py:5124-5134`; `sdm_spatial.py:360-382`).

- In the `random` design the number of folds cannot exceed the number of records of the rarer class (`sdm_spatial.py:572-574`).
- In the checkerboard designs the number of folds is fixed (2 or 4); in these designs `Cross-validation folds:` only limits how many folds are run (`sdm_core.py:5400-5401`). With `checkerboard2`, a value below 4 leaves some folds never tested; keep the number of folds at 4 in this design.

6.4.2 Block size: automatic or manual

What it does. When `Size the blocks automatically from the autocorrelation range` is on, the block side is the empirical variogram range computed in the ① Data stage from the predictor values at the presence points (`sdm_core.py:2433-2462`, `5096-5106`). The computation is as follows (`sdm_spatial.py:166-281`): at most 1500 points are subsampled; the semivariance $\gamma(h) = \frac{1}{2}\overline{(z_i - z_j)^2}$ is computed in 15 distance classes up to half the largest pairwise distance; the median of the outer third of the classes is taken as the sill; the distance at which γ first reaches 95% of the sill is the range of that predictor; the block size is the **median of these ranges across predictors**. If there are fewer than 20 points or the variogram cannot be built, 1/10 of the largest pairwise distance of the study area is used (`method = "fallback_extent"`). When the box is off, the `Block size:` value is used.

When and how to set it. The automatic value is a simplified counterpart of the logic in blockCV (Valavi et al. 2019). It is entered manually if the range is to be fixed to a scale stated in a paper (e.g. the width of a mountain range) or for comparison with another study.

Effect on results. As the block grows, leakage between folds decreases, scores fall and the difference between folds grows. In the example run the variogram range is 151.5 km and the study-area width 606 km; four folds were built from 30 blocks, with 43/33/35/32 presences in the folds (`habitus_run_config.json`).

Caution. If the area does not hold as many blocks as the requested number of folds, the block can be halved up to six times; in this case the note reads `"Block reduced from … km so that the study area could hold … folds …"` (`sdm_spatial.py:454-464`, `503-507`). This reduction applies to both the automatic and the manually entered size. The grid is built in degrees, with the longitude spacing corrected by $\cos\varphi$ at the mean latitude (`sdm_spatial.py:439-446`).

6.4.3 Aggregation factor (Aggregation factor)

What it does. For `checkerboard`, cell side = block size × factor; for `checkerboard2`, fine grid = block size, coarse grid = block size × factor (at least 2) (`sdm_spatial.py:651-672`). If the block size is automatic, it is the variogram range.

When and how to set it. It plays the same role as `aggregation.factor` in ENMeval (Muscarella et al. 2014). The default of 2 is sufficient in most cases; increasing it enlarges the checkerboard cells and moves test points away from training points.

Effect on results. A large factor gives a more pessimistic (more independent) test; with a very large factor some colours may contain only one class and the cell is reduced.

Caution. If each fold does not contain both classes, the cell is halved up to six times; if this still fails, the program falls back to random folds (`sdm_spatial.py:685-698`).

6.4.4 Number of folds (Cross-validation folds)

What it does. It is the number of folds to be formed in the `spatial_block`, `environmental`, `random` and band designs; in all designs it is the upper limit on the number of CV runs (`RUN1…RUNk`) to be executed (`sdm_core.py:5400-5407`).

When and how to set it. 4–5 folds are common. As the number of folds increases, each test set shrinks and the variability of the fold scores increases; 3–4 folds are reasonable for small samples, 5 folds for large areas with many records.

Effect on results. Run time grows linearly with the number of folds. The mean and standard deviation of the scores are affected by the number of folds.

Caution. If 1 is selected, the folds are still built as two parts, but only one is tested (`sdm_core.py:5018`, `5400-5401`); in this case the spread of the fold scores cannot be computed and the hold-out score of the ensemble rests on the members' own fold scores (§6.6.9). In `jackknife` the number of runs equals the number of presences and this field is ignored (`sdm_core.py:5402-5406`).

6.4.5 Test data source and external test CSV

What it does. When `Random split` is selected (default), the tests are carried out on the held-out part of the CV folds. When `External test points (CSV)` is selected, every `RUN` is trained on **all** the PA data and tested on the points in the file (`sdm_core.py:5417-5427`). The coordinate columns are taken from the `Test CSV lon / lat:` fields; if these are empty, the names `longitude/long/lon/lng/decimallongitude/x` and `latitude/lat/decimallatitude/y` are searched for, and if they are not found the 2nd and 3rd columns are used (`sdm_core.py:5165-5189`). If the file has a column named `presence`, `occurrence` or `pa`, rows with value 0 are treated as true absences; otherwise the test absences are background points newly drawn from the study area, independent of the training; if this also fails, a seeded subsample of the training pseudo-absences is taken and its non-independence is noted (`sdm_core.py:5191-5340`).

When and how to set it. It is used when records from an independent field survey, another period or another region are available.

Effect on results. The scores rest on independent data; however, because all runs are trained on the same data and tested on the same points, the difference between runs shows only the algorithm's own randomness. The program writes this to the report with the note `"External test set: RUN1..RUNn all re-fit the same model on the SAME training data …"` (`sdm_core.py:5341-5347`).

Caution. Test records falling outside the raster or on NoData are counted and reported; if fewer than 10 usable presences remain, the warning `"… the test statistics will be very unstable."` appears (`sdm_core.py:5289-5301`). In this mode `Training data %:` is greyed out and 80 is used (`tabs/tab_models.py:1296-1301`, `1578`). If the run is started without selecting a file, the error `"External test points mode selected but no CSV provided."` is given.

6.4.6 Training share (Training data %)

What it does. For every model fitted on all data (`Full`), a random evaluation split is also made: a model is fitted with this percentage of the data and scored on the rest (`sdm_core.py:5675-5722`). If at least one valid CV score exists for the same algorithm and PA set, this split score is **replaced by the mean of the CV folds** and `Score_source` becomes `cv_mean` (`sdm_core.py:5818-5853`). The split score is kept only if no CV score was produced at all (`Score_source` = `random_split`).

When and how to set it. In normal use there is no need to change it.

Effect on results. When CV succeeds, it has no effect on the reported scores, thresholds or maps.

Caution. The field's tooltip text (`"% of occurrence data used for training; remainder used as test set"`) gives the impression that this setting determines the CV scores; it does not.

6.4.7 Variable importance repeats (Variable importance reps)

What it does. In each CV run, each predictor is permuted in the test fold and the drop in AUC is taken as its importance. For scikit-learn models `permutation_importance` (`n_repeats=3`, `scoring="roc_auc"`, fixed seed) is called; in MaxEnt, GAM, ENFA, Mahalanobis, BIOCLIM, Domain and ESM, each predictor is permuted with its own seeded generator and the AUC difference is taken (`sdm_core.py:5562-5636`). The mean of the repeats is stored; 0 switches importance off.

When and how to set it. If the importance plot is not needed, 0 shortens run time. The default is 3.

Effect on results. It does not change the maps or scores; it only feeds the importance plot in ④ Evaluation (Chapter 7).

Caution. For scikit-learn models the loop uses the same seed in every repeat, so the values do not change between repeats; for these algorithms the effective number of repeats is `n_repeats` = 3 and the field in practice acts as an on/off switch. Importance is not computed for the full-data (`Full`) models.

6.4.8 Clamping and class balancing

What it does. `Clamp predictors …` clamps the predictors to the training minimum/maximum values of all PA sets before every prediction (`sdm_core.py:4929-4962`, `4979-4980`). `Balance classes …` weights the presence and pseudo-absence classes inversely to their sizes in the algorithms that support it; the line below it shows the current PA:presence ratio (`1000 : 143 (7.0 : 1)` in the example run) (`tabs/tab_models.py:1316-1335`). Details per algorithm are in Chapter 5.

Effect on results. Clamping is almost without effect within CV; it limits extrapolation in future projections. Balancing changes the scale of the probabilities; thresholds are re-derived and the AUC ranking mostly does not change (tooltip text).

6.4.9 Spatial sorting bias (SSB) report

This is not a setting; it is computed automatically in every run. For the fold design used (Hijmans 2012):

$$\mathrm{SSB} = \frac{\overline{d}(\text{test presence} \rightarrow \text{nearest training presence})}{\overline{d}(\text{test absence} \rightarrow \text{nearest training presence})}$$

Distances are great-circle km, pooled over folds and also given per fold (`sdm_spatial.py:808-881`). The interpretation thresholds are in the code: below 0.7 "spatial sorting bias … scores optimistic", 0.7–1.3 "little or no spatial sorting bias", above 1.3 an extrapolating (pessimistic) split (`sdm_spatial.py:866-875`). The result is written to the log with the line `CV PA1: SSB = …`, to the run record under `per_pa_set.*.ssb`, and to the summary line as `[SSB=…]` (`sdm_core.py:5142-5153`, `5942-5944`).

> **Note:** In the example run SSB = 0.36 was found despite spatial blocks (test presences are on average 45.5 km from the nearest training presence, test absences 127.9 km). Because the pseudo-absences are spread over the whole area while the presences are concentrated in a narrow belt, this ratio can come out low regardless of blocking; use SSB to compare designs, not as a pass/fail criterion on its own.

6.5 Evaluation settings: threshold, omission reference, Boyce

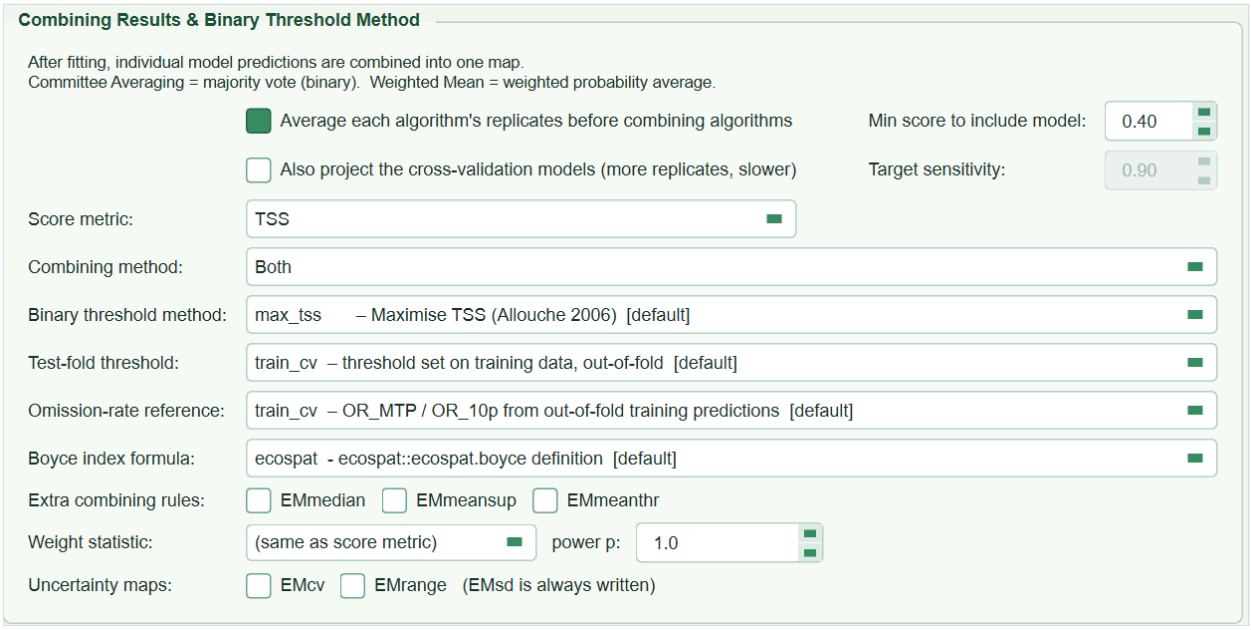


Figure 6.4. The Combining Results && Binary Threshold Method box with default values: two-level combining on, TSS score metric and 0.40 minimum score, max_tss threshold rule, train_cv threshold source and omission reference, ecospat Boyce formula.

Figure 6.4. The Combining Results && Binary Threshold Method box with default values: two-level combining on, TSS score metric and 0.40 minimum score, max_tss threshold rule, train_cv threshold source and omission reference, ecospat Boyce formula.

This box does two jobs: it sets the threshold of the binary map and of the threshold-dependent scores (TSS, Kappa, MCC, sensitivity, specificity, Jaccard, Sørensen, FPB), and the ensemble settings (§6.6). First the threshold and score settings:

Name in the UI	What it does	Default	Range / options	Code
`Binary threshold method:`	Rule that converts probability to presence/absence	`max_tss`	11 rules (§6.5.1)	`tabs/tab_models.py:435-455`
`Target sensitivity:`	Target of the `sensitivity` rule	0.90	0.50–1.00	`tabs/tab_models.py:456-462`
`Test-fold threshold:`	Data on which the threshold is chosen	`train_cv`	`train_cv` / `train` / `test`	`tabs/tab_models.py:465-490`
`Omission-rate reference:`	Source of the OR_MTP and OR_10p thresholds	`train_cv`	`train_cv` / `train`	`tabs/tab_models.py:491-503`
`Boyce index formula:`	Definition of the continuous Boyce index	`ecospat`	`ecospat` / `legacy`	`tabs/tab_models.py:504-514`

6.5.1 Binary threshold rule (Binary threshold method)

What it does. A threshold is chosen for each model; cells with probability equal to or greater than this threshold become 1 (`sdm_core.py:7907-7915`). All rules are in `find_optimal_threshold` (`sdm_core.py:795-948`). Grid-search rules try 200 equally spaced thresholds between 0.01 and 0.99.

Rule	Threshold chosen	Source	Code
`max_tss`	Threshold maximizing TSS = sensitivity + specificity − 1	Allouche et al. 2006	`sdm_core.py:891-897`
`max_kappa`	Threshold maximizing Cohen's kappa	Fielding and Bell 1997	`sdm_core.py:899-915`
`sens_spec_eq`	Minimum |sensitivity − specificity|	Liu et al. 2005	`sdm_core.py:917-931`
`p10`	10th percentile of the presence predictions	Pearson et al. 2007	`sdm_core.py:858-864`
`min_roi`	Point on the ROC curve closest to the (0,1) corner: $\min \sqrt{\mathrm{FPR}^2 + (1 - \mathrm{TPR})^2}$	—	`sdm_core.py:933-945`
`mtp`	Lowest presence prediction (no presence is left out; same as `lpt`)	Pearson et al. 2007	`sdm_core.py:734-735`
`p5`	5th percentile of the presence predictions	—	`sdm_core.py:736-738`
`sensitivity`	Highest threshold retaining at least the target share of presences: the $\lfloor (1 - s)\, n \rfloor$-th value of the sorted presence predictions	Allouche et al. 2006	`sdm_core.py:866-876`
`max_jaccard`	Maximum $\mathrm{TP}/(\mathrm{TP} + \mathrm{FP} + \mathrm{FN})$	—	`sdm_core.py:878-885`
`max_sorensen`	Maximum $2\mathrm{TP}/(2\mathrm{TP} + \mathrm{FP} + \mathrm{FN})$	—	`sdm_core.py:878-885`
`max_fpb`	Maximum FPB = 2 × Jaccard (flexsdm's presence-background F measure)	Li and Guo 2013	`sdm_core.py:701`, `878-885`

When and how to set it. `max_tss` is common with pseudo-absence data and relatively independent of prevalence. In protected-area studies where a false negative is costly, `p10`, `p5`, `mtp` or `sensitivity` (e.g. 0.95) give a larger area. If omission and commission errors are equally important, `sens_spec_eq` is chosen. The Kappa and Jaccard family is affected by prevalence; the threshold shifts as the number of pseudo-absences changes.

Effect on results. The area in the binary maps, the range-change calculations (⑥ Range Change) and all threshold-dependent scores depend on this rule. AUC, Boyce, Brier and the calibration slope are threshold-independent. In the example run the thresholds chosen with `max_tss` lie between 0.06 and 0.21 (GLM_PA1 0.1294, RF_PA2 0.2058; session log): because the probability scale is low, a fixed threshold such as 0.5 would give a very narrow map with these data.

Caution. The threshold used on the map is the mean of the thresholds chosen in that model's CV folds (`sdm_core.py:5852-5853`); it is not re-chosen on the full-data model's own data. If the threshold cannot be computed, 0.5 is used and `FAILED, 0.5 used (…)` is written in the `Threshold_source` column (`sdm_core.py:1221-1227`). The `Target sensitivity:` box is enabled only when `sensitivity` is selected; the

target is carried inside the key (`sensitivity:0.90`) and written to the run record in that form (`tabs/tab_models.py:1504-1507`).

6.5.2 Data on which the threshold is chosen (Test-fold threshold)

What it does. For the threshold-dependent scores in the test fold, it determines on which predictions the threshold is chosen (`evaluate_all`, `sdm_core.py:1122-1257`):

- `train_cv` (default): the training rows are split into three inner folds (`OOF_INNER_FOLDS = 3`, the folds are built with the user's CV design); each row is predicted by a model that has not seen it; the threshold is chosen on these out-of-fold predictions and applied to the test fold (`sdm_core.py:4876-4919`, `4296-4368`).
- `train`: the threshold is chosen on the model's predictions for its own training rows.
- `test`: the threshold is chosen directly on the test fold, so as to maximize that fold's score (oracle threshold).

When and how to set it. `train_cv` is the recommended and honest option. `train` is cheap, but for algorithms that memorize the training data (Domain, RF, GBM, XGBoost) the threshold becomes meaningless. `test` is only for comparison with earlier results.

Effect on results. The `test` option typically gives a TSS 0.1–0.2 higher (tooltip text, `tabs/tab_models.py:480-482`). `train_cv` adds three inner fits to every CV fit; run time roughly doubles. Because the threshold on the map is the mean of the thresholds in the CV folds (§6.5.1), this choice also affects the maps.

Caution. When `test` is selected, an orange warning appears below the box: `"Warning: 'test' picks the threshold on the very data it is scored on …"` (`tabs/tab_models.py:1261-1272`). If the inner fit fails, the source is labelled `train (inner CV unavailable)` (`sdm_core.py:1201-1207`). ESM does not produce inner-fold predictions (`sdm_core.py:4889-4890`).

6.5.3 Omission-rate reference (Omission-rate reference)

What it does. OR_MTP and OR_10p are the shares of test presences falling below, respectively, the lowest and the 10th-percentile **training** presence prediction (`sdm_core.py:1062-1115`). `train_cv` takes these reference values from the inner out-of-fold predictions, `train` from the model's own training predictions.

When and how to set it. The default `train_cv` is recommended; it follows the same logic as ENMeval's definition of omission (Muscarella et al. 2014), with the difference that the reference is not affected by memorization.

Effect on results. It affects only the OR_MTP and OR_10p columns and the ENMeval selection rule in Tuning (§6.7); it does not affect the maps.

Caution. When `train` is selected, the warning is: `"Warning: with in-sample references OR_10p / OR_MTP are unreliable for over-fitting algorithms …"`. If the threshold value is equal to or below the lowest prediction of the evaluation set, omission is 0 by construction; HABITUS flags this with the `OR_10p_degenerate` column and writes a note for the algorithm if this happens in at least half of the folds (`sdm_core.py:1100-1114`, `5787-5816`).

6.5.4 Boyce index formula (Boyce index formula)

What it does. `ecospat` is a line-by-line port of the `ecospat::ecospat.boyce` function (`sdm_core.py:475-543`): the predictions of all evaluated rows give the expected distribution (E), the predictions of the presences give the observed distribution (P); a window whose width is 1/10 of the prediction range is moved across 101 positions, $F = P_i/E_i$ is computed in each window, consecutive repeated F values are removed, and the index is the Spearman correlation between the window lower bounds and F (Hirzel et al. 2006; Boyce et al. 2002). `legacy` is the earlier HABITUS definition, computed with a fixed window of width 0.1 and 101 centres on the [0, 1] axis (`sdm_core.py:612-664`).

When and how to set it. `ecospat` (default) for results comparable with ecospat in R. `legacy` is only for reproducing an earlier result.

Effect on results. It changes the Boyce scores, and the ensemble selection and weights when Boyce is the score metric. A code comment records that on the same BIOCLIM predictions `legacy` gives 0.007 and `ecospat` 0.334 (`sdm_core.py:560-562`).

Caution. If the evaluation set contains no presences or fewer than 5 rows, Boyce is `NaN` (`sdm_core.py:605-609`).

6.6 Ensemble settings (Combining Results)

Name in the UI	What it does	Default	Range / options	Code
`Average each algorithm's replicates before combining algorithms`	Two-level ensemble	on	on / off	`tabs/tab_models.py:516-530`
`Also project the cross-validation models (more replicates, slower)`	Also projects the CV models	off	on / off	`tabs/tab_models.py:532-540`
`Combining method:`	Basic ensemble rules written	`Both`	`Weighted Mean` / `Committee Averaging` / `Both`	`tabs/tab_models.py:403-417`
`Score metric:`	Member filtering metric	`TSS`	`TSS` / `ROC` / `Boyce`	`tabs/tab_models.py:418`
`Min score to include model:`	Minimum score	depends on metric: TSS 0.40, ROC 0.70, Boyce 0.50	0–1	`tabs/tab_models.py:419-433`, `sdm_core.py:6093`
`Extra combining rules:`	Additional rules	all off	`EMmedian`, `EMmeansup`, `EMmeanthr`	`tabs/tab_models.py:543-558`
`Weight statistic:`	Weighting statistic	`(same as score metric)`	+ `AUC`, `TSS`, `Boyce`	`tabs/tab_models.py:559-566`
`power p:`	Weighting power	1.0	0.5–10.0	`tabs/tab_models.py:567-575`
`Uncertainty maps:`	Additional uncertainty maps	off (EMsd always)	`EMcv`, `EMrange`	`tabs/tab_models.py:576-583`

6.6.1 Two-level ensemble

What it does. When on (default), the replicates of the same algorithm (one full-data model per PA set; also the CV models if §6.6.2 is on) are averaged first; the mean per algorithm, the standard deviation between replicates and the vote share of the replicates are computed (`EnsembleModeler.predict_by_algorithm`, `sdm_core.py:6234-6331`). The ensemble is built from these algorithm means (`sdm_core.py:6359-6376`). When off, every model that passes the filter enters a single flat weighted mean (`sdm_core.py:6377-6407`).

When and how to set it. Leave it on: otherwise an algorithm with many replicates gets more votes in the ensemble.

Effect on results. In addition, `<algo>_mean_prob`, `<algo>_mean_bin` and, if there are several replicates, `<algo>_sd` maps are written for each algorithm (`sdm_core.py:7964-8021`). Weights are computed at the algorithm level; in the example run the TSS weights are GLM 0.271, MaxEnt 0.265, GAM 0.241, RF 0.223.

Caution. When off, `EMsd` is not written unless an additional uncertainty map is selected (`sdm_core.py:6416-6419`); the phrase `(EMsd is always written)` in the UI applies only to the two-level mode.

6.6.2 Also projecting the CV models

What it does. By default only the full-data models are projected (as many replicates per algorithm as there are PA sets). When the box is on, the models of the CV folds are also projected and each is written as a separate file (`<algo>_PA1_RUN1_prob.tif` …) (`sdm_core.py:7850-7916`). The choice is also passed to ⑤ Future Scenarios (`tabs/tab_models.py:1538-1546`).

When and how to set it. When more replicates are wanted for within-algorithm instability (SD between replicates). When the number of PA sets is 1, only one replicate per algorithm remains, so the `<algo>_sd` map can only be produced with this box.

Effect on results. Projection time and disk use increase roughly (folds + 1)-fold. The ensemble mean also includes the CV models.

Caution. The ensemble is built only from the projected models (`sdm_core.py:6343-6355, 8028-8035`). When the box is off, if only some CV models pass the minimum score and no full-data model does, the ensemble is not written and the program gives the message `"No full-data model reached the minimum … Lower the minimum score, or tick 'Also project the cross-validation models' …"` (`sdm_core.py:8040-8054`). When the box is on, each CV model is filtered separately by its own fold score; the model of a weak fold may be left out.

6.6.3 Combining method and extra rules

What it does. `Combining method:` determines which basic rules are computed and written: `Weighted Mean` → `EMwmean`, `Committee Averaging` → `EMca`, `Both` → both (`tabs/tab_models.py:1670-1694`; `sdm_core.py:6173-6183, 6409-6412`). A rule that is not selected is not computed, written or scored. Definitions (in two-level mode, $\bar{p}_a$ is the algorithm mean, w_a the weight, v_a the share of the algorithm's replicates that say "present" at their own threshold):

$$\mathrm{EMwmean} = \sum_a w_a \, \bar{p}_a, \qquad \mathrm{EMca} = \frac{1}{A} \sum_a v_a$$

Extra rules (`sdm_core.py:6016-6040`): `EMmedian` is the per-cell median of the member probabilities; `EMmeansup` is the simple mean of the "superior" members whose weight statistic is equal to or above the members' mean

(flexsdm `meansup`); `EMmeanthr` is the mean probability of the members that reach their own threshold in the cell, 0 if none does (flexsdm `meanthr`; Velazco et al. 2022).

When and how to set it. The weighted mean and committee averaging are consensus methods that performed well in the comparison of Marmion et al. (2009). The median is robust to an extreme member; `meanthr` gives a broad map of the "at least one member says present" type.

Effect on results. Each rule produces separate `_prob` and `_bin` maps and receives a separate hold-out score (§6.6.6). `EMca` is a vote share, not a probability.

Caution. Committee averaging is unweighted across algorithms; the weight statistic affects only `EMwmean` and `EMmeansup`.

6.6.4 Score metric and minimum score

What it does. A model enters the ensemble if its `Score metric:` value is equal to or greater than the `Min score to include model:` value (`EnsembleModeler._filter_models`, `sdm_core.py:6185-6196`). The score of the full-data models is the mean of the CV folds (§6.4.6). The default minimum score depends on the metric: ROC 0.70, TSS 0.40, Boyce 0.50 (`sdm_core.py:6086-6098`); it follows the metric until the user types a value, and is not changed after that (`tabs/tab_models.py:1696-1710`).

When and how to set it. With spatially blocked CV, honest TSS values usually lie in the 0.3–0.6 range; this is why 0.40 was chosen for TSS (code comment, `sdm_core.py:6086-6092`). It is raised if weak models are to be excluded; set it by looking at the score of the best model so that the ensemble does not end up empty.

Effect on results. The number of ensemble members, and therefore the map, changes. In the example run all eight of the eight full-data models met the condition TSS ≥ 0.40 (`Models included in combined map: 8/8`).

Caution. If no model passes the threshold, the `EM*` maps are not written but the algorithm maps are; the status text starts with `WARNING - NO COMBINED MAP:` and the message states the score of the best model (`sdm_core.py:6198-6232`; `tabs/tab_models.py:1831-1833`).

6.6.5 Weight statistic, weighting power and uncertainty maps

What it does. The member weight is computed as

$$w_i = \frac{\max(s_i, 0)^p}{\sum_j \max(s_j, 0)^p}$$

where s_i is the statistic selected with `Weight statistic:` (default: the score metric) and p the `power p:` value (`sdm_core.py:5988-6005`). Non-finite scores receive an equal share; if all weights are zero, members are weighted equally. Filtering is still done with the score metric. `EMsd` is the standard deviation between members (ddof = 1); `EMcv` = SD / mean (0 in cells where the mean ≈ 0); `EMrange` = maximum − minimum (`sdm_core.py:6068-6083`). Uncertainty surfaces are written without normalization and are never thresholded (`sdm_core.py:8057-8068`).

When and how to set it. $p = 1$ is weighting proportional to the score (same meaning as `power` in the sdm package); 2–3 can be used to emphasize the best members. It is possible to choose a weight statistic different from the score metric, e.g. filtering with TSS and weighting with Boyce.

Effect on results. As p grows, `EMwmean` approaches the best algorithm. Uncertainty maps do not change the ensemble map.

Caution. Uncertainty surfaces require at least two members; in a single-algorithm run (in two-level mode) `EMsd` is not produced (`sdm_core.py:6073-6074`).

6.6.6 Binary threshold of the ensemble

The program converts the ensemble maps to binary with the following thresholds (`EnsembleModeler.binary_threshold`, `sdm_core.py:6423-6461`): for `EMca` 0.5 (majority vote); for `EMwmean` the mean of the members' thresholds weighted by the map weights; for `EMmedian` the median of the member thresholds; for `EMmeansup` the mean threshold of the superior members; for `EMmeanthr` the smallest member threshold (`sdm_core.py:6043-6065`). In the example run the `EMwmean` threshold is 0.1166 and the `EMca` threshold 0.5. Ensemble maps are not stretched to their own range; they are only clipped to [0, 1] (`sdm_core.py:8074-8081`).

6.6.7 Hold-out score of the ensemble and member selection from the other folds

Because the ensemble map is built from models that have seen all presences, it is not scored on its own map. Instead, for each PA set × fold group, CV models that have not seen that fold (one per algorithm) are combined with the ensemble's rule and scored only on the held-out rows of that fold; the reported value is the mean over fold groups (`EnsembleModeler.evaluate_cv`, `sdm_core.py:6516-6696`). Whether a member enters the ensemble, and its weight, are computed from the mean of the same algorithm's scores in the **other** folds (leave one fold out; the same fold number is also left out in the other PA sets; `sdm_core.py:6463-6496`). The ensemble's threshold is chosen on the training surface built from the members' inner out-of-fold training predictions (`sdm_core.py:6619-6657`). In the example run all eight of the eight fold groups were used; for `EMwmean` ROC 0.902, TSS 0.659, Boyce 0.726, and for `EMca` ROC 0.878, TSS 0.660, Boyce 0.725 were found. These rows are added to `evaluation_scores.csv` with `PA_set` = `Ensemble`, `CV_run` = `CV_holdout`, `Score_source` = `cv_holdout` (`sdm_core.py:6698-6729`).

> **Note:** The hold-out score is computed with a flat (single-level) combination in each fold, because each group contains only one model per algorithm. In a single-fold design there is no "other fold", so the member is weighted by its own fold score, and this is counted in the `cv_note` text (`sdm_core.py:6511-6513`).

6.7 Tuning search (Tuning (optional))



Figure 6.5. The settings panel of the Tuning (optional) sub-tab (the result table and plot on the right are empty because no search has been run): for MaxEnt, rm 0.50–4.00 in steps of 0.50 and the L, LQ, H, LQH, LQHP feature classes; 40 grid points, about 160 model fits.

Figure 6.5. The settings panel of the Tuning (optional) sub-tab (the result table and plot on the right are empty because no search has been run): for MaxEnt, rm 0.50–4.00 in steps of 0.50 and the L, LQ, H, LQH, LQHP feature classes; 40 grid points, about 160 model fits.

This sub-tab searches the settings of a single algorithm over a grid. Each grid point is tested with the CV design, threshold rule and source, omission reference, Boyce formula and clamping setting from `Algorithm Settings`, on **the same folds** at all points (`tabs/tab_models.py:1274-1294`; `sdm_tuning.py:394-494`). For each point only CV is run (`run(cv_only=True)`); the full-data model and variable importance are not computed.

Name in the UI	What it does	Default	Range / options	Code
`Algorithm:`	Algorithm to search	`MaxEnt (rm x feature classes)`	MaxEnt, GBM, RF, SVM, ANN, XGBoost, LightGBM, GLMNET, CART, GAM (those installed)	`tabs/tab_tuning.py:91-96`
`Selection rule:`	Rule for choosing the best point	ENMeval-style	3 rules	`tabs/tab_tuning.py:97-110`, `sdm_tuning.py:84-88`
`Pseudo-absence sets used:`	First n PA sets used in scoring	1	1–20	`tabs/tab_tuning.py:111-114`

Name in the UI	What it does	Default	Range / options	Code
`rm (regularisation multiplier):`	Sequence of MaxEnt regularization multipliers	start 0.50, end 4.00, step 0.50	0.05–50	`tabs/tab_tuning.py:211-226`, `sdm_tuning.py:506`
`Feature classes (fc):`	Sets of MaxEnt feature classes	`L, LQ, H, LQH, LQHP`	letters L, Q, H, P, T	`tabs/tab_tuning.py:227-233`, `sdm_tuning.py:505`
Grid rows of the other algorithms	Comma-separated list of values or `start:end[:step]`	depends on algorithm (below)	—	`tabs/tab_tuning.py:234-242`
`▶ Run grid search`, `Cancel`, `Reset grid to defaults`	Start; stop between points; return to defaults	—	—	`tabs/tab_tuning.py:137-148`
`Apply selected setting to Models tab`, `Save table as CSV…`	Writes the selected point to the algorithm options; saves the table	—	—	`tabs/tab_tuning.py:170-178`, `480-499`

Default grids (`sdm_tuning.py:626-643`, `776-793`, `837-843`): GBM `n_estimators` 20, 50, 100, 200, 500 × `learning_rate` 0.01, 0.1, 0.3 × `min_samples_leaf` 1–10, 20, 30 (flexsdm `tune_gbm`, 180 points); RF `max_features` from 2 to the number of predictors × `n_estimators` 200–1200 (flexsdm `tune_raf`); SVM `C` 1, 2, 4, 8, 16 × `gamma` 0.001, 0.01, 0.1, 0.2 (kernlab `sigma` = sklearn `gamma`); ANN `hidden_layer_sizes` 2, 4, 6, 8 × `alpha` = 2 × nnet `decay`; 3 × 3 × 3 for XGBoost and LightGBM; GLMNET `C` 0.01–10 × `l1_ratio` 0–1; CART `max_depth` × `min_samples_leaf` × `ccp_alpha`; GAM `n_splines` 5–20 × `lam` 0.1–10.

6.7.1 Selection rule (Selection rule)

What it does. `enmeval`: the point with the lowest mean OR_10p, ties broken by the highest mean test AUC (Muscarella et al. 2014; Kass et al. 2021). `enmeval_guard`: first discards points worse than one fold-SD below the best AUC, then applies the same rule. `max_auc`: the highest mean test AUC. Points whose OR_10p is 0 only because of tied low predictions ("degenerate") are not selected under the ENMeval rules (`sdm_tuning.py:238-298`). The rule can be changed after the search has finished without refitting (`tabs/tab_tuning.py:415-418`).

When and how to set it. For MaxEnt, ENMeval-style agrees with common practice; `enmeval_guard` to prevent the selection of points with markedly low AUC.

Effect on results. The selected point is written to the algorithm options with `Apply selected setting to Models tab`; the main run uses it. The search result, the grid, the rule and whether the selected point was actually used in the run are written to `habitus_run_config.json` as a `tuning_<algo>` section (`tabs/tab_models.py:2195-2208`).

Caution. AICc is not computed; the program says so with the note `"AICc / delta-AICc is not computed: …"` (`sdm_tuning.py:62-66`). Because selection is made on the same folds that produce the reported scores, the scores of the tuned algorithm are optimistic; there is no nested validation (`sdm_tuning.py:68-78`). Searches exceeding 60 points or 600 fits ask for confirmation (`tabs/tab_tuning.py:36-38`, `318-327`). The result table is written automatically to the file `<output>/tuning/tuning_<algo>.csv` (`tabs/tab_tuning.py:397-413`).

6.8 Evaluation results (Evaluation Results)

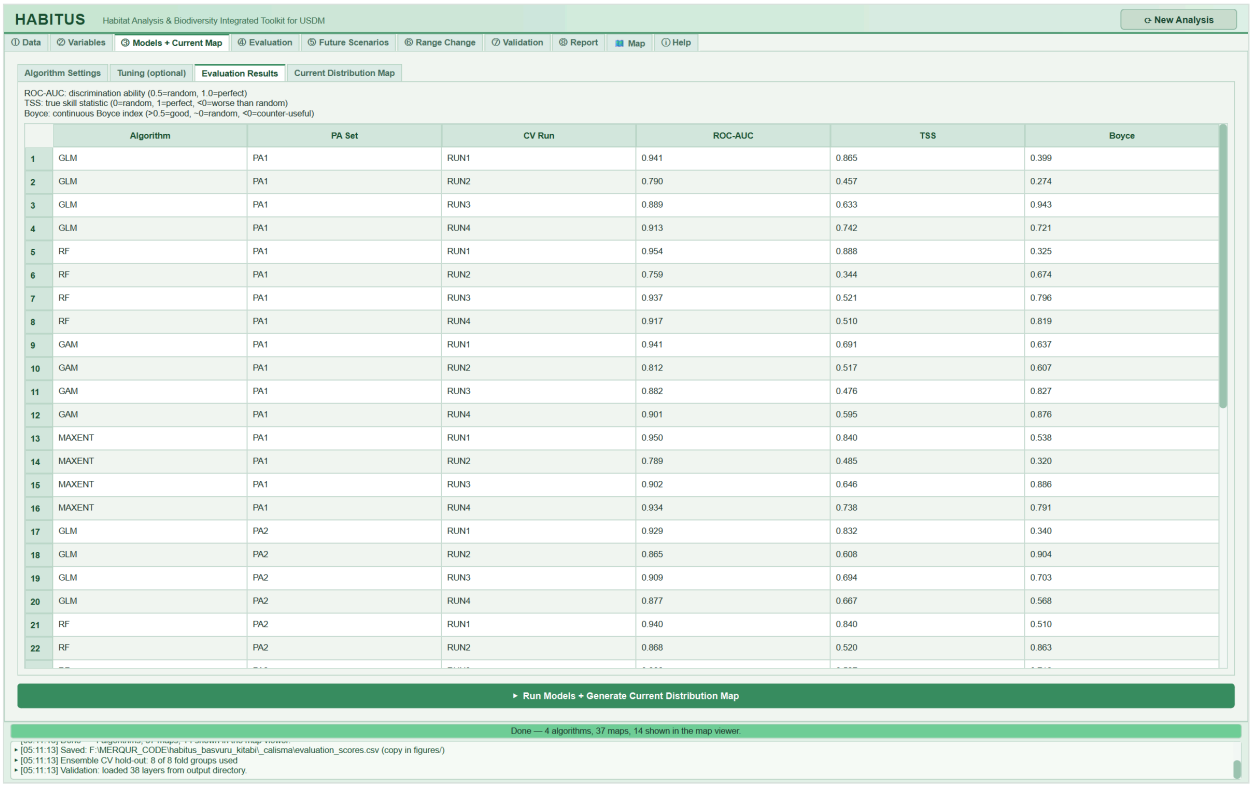


Figure 6.6. The Evaluation Results sub-tab after the example run: ROC-AUC, TSS and Boyce values for each algorithm, PA set and CV run.

Figure 6.6. The Evaluation Results sub-tab after the example run: ROC-AUC, TSS and Boyce values for each algorithm, PA set and CV run.

The table shows the first six columns of the `evaluation_scores_table` output: `Algorithm`, `PA Set`, `CV Run`, `ROC-AUC`, `TSS`, `Boyce` (`tabs/tab_models.py:618-634`, `1753-1782`). The rows come first as CV runs (`RUN1…`), then the full-data models (`Full`, carrying the CV mean), and finally the ensemble's hold-out rows (`EMwmean`, `EMca` …, `PA Set` = `Ensemble`). Values of 0.8 and above are shown in green, 0.6–0.8 in orange, below 0.6 in red; non-numeric values appear as `N/A`. The tooltip of the ensemble rows says that the score comes from the CV hold-outs, not from the map (`tabs/tab_models.py:1769-1774`).

The full table and the other columns (threshold, prevalence, Brier, calibration slope, Kappa, MCC, sensitivity, specificity, omission rates, `Threshold_source`, `Score_source` …) are in the `evaluation_scores.csv` file; their meanings are explained in Chapter 7.

> **Tip:** The colour thresholds are the same for all metrics; for Boyce, values above 0.5 are already considered good (explanation at the top of the tab). The colour is not a judgement but a marker that makes the table easier to scan.

6.9 Current distribution map (Current Distribution Map)

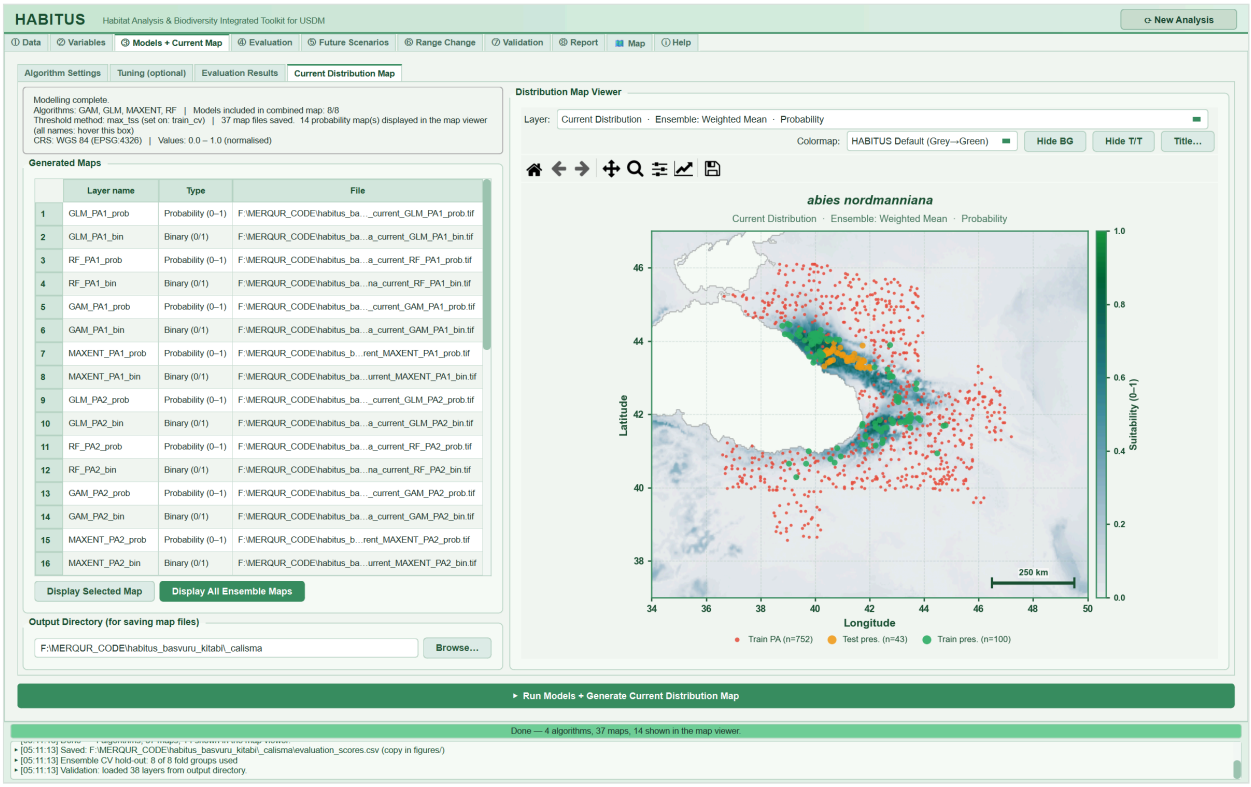


Figure 6.7. The Current Distribution Map sub-tab: on the left the status text, the Generated Maps table and the output folder; on the right the weighted-mean ensemble map, overlaid with training presences (green, n=100), test presences (orange, n=43) and training pseudo-absences (red, n=752).

Figure 6.7. The Current Distribution Map sub-tab: on the left the status text, the Generated Maps table and the output folder; on the right the weighted-mean ensemble map, overlaid with training presences (green, n=100), test presences (orange, n=43) and training pseudo-absences (red, n=752).

Name in the UI	What it does	Default	Code
Status text	Algorithms, number of models entering the ensemble, threshold rule and source, number of maps, CRS; the long layer list is in the tooltip	—	`tabs/tab_models.py:643-651`, `1817-1834`
`Generated Maps` table	`Layer name`, `Type` (`Probability (0–1)` / `Binary (0/1)`), `File`	—	`tabs/tab_models.py:654-666`, `1798-1812`
`Display Selected Map`	Loads the map of the selected rows into the viewer	—	`tabs/tab_models.py:1963-1977`
`Display All Ensemble Maps`	Loads all maps beginning with `EM`	—	`tabs/tab_models.py:1979-1984`
`Output Directory (for saving map files)`	Folder the maps are written to; if empty, the output folder from ① Data	from ① Data	`tabs/tab_models.py:679-686`, `1467-1477`

Name in the UI	What it does	Default	Code
`Distribution Map Viewer`	Layer selection, colour scale, `Hide BG`, `Hide T/T`, `Title…`	—	`tabs/tab_models.py:688-695`; `map_widget.py:346-401`

When the run finishes, all probability maps are loaded into the viewer automatically (`tabs/tab_models.py:1808-1812`); layer names have the form `Current Distribution · <algorithm> · PA-<n> · Probability` or `… · Ensemble: Weighted Mean · Probability` (`tabs/tab_models.py:35-47`). The split in the **first CV fold** of the first PA set is drawn over the map: training presences in green, test presences in orange, training pseudo-absences in red (`tabs/tab_models.py:1859-1907`; `sdm_core.py:5470-5513`). The same points are written to the output folder as `occurrence_train.csv`, `occurrence_test.csv` and `occurrence_all_splits.csv` (`tabs/tab_models.py:1914-1961`). `Hide BG` hides the pseudo-absence points, `Hide T/T` the training/test points; the viewer's other tools are described in Chapter 12.

Caution. In the example run the status text says `37 map files saved. 14 probability map(s) displayed`: 8 individual models × 2 (probability, binary) + 4 algorithms × 3 (mean probability, mean binary, SD) + 2 ensembles × 2 + `EMsd` + 4 novelty layers (`NT1`, `NT2`, `MIC_NT1`, `MIC_NT2`) = 37 files; the 14 maps loaded into the viewer are the 8 individual maps ending in `_prob`, the 4 algorithm means and the 2 ensemble maps. The "test" points on the map do not come from an 80/20 split but from the first spatial fold; this is why they form a contiguous cluster (the orange cluster in Figure 6.7). The values are raw probabilities clipped to [0, 1]; the word `normalised` in the status text does not mean that the map has been stretched to its own range (`sdm_core.py:7884-7889`).

6.10 Outputs and what passes to the next step

To the `<output>/current/` folder (`tabs/tab_models.py:1615`; `sdm_core.py:7895-8103`):

- `<species>_current_<ALGO>_PA<n>_prob.tif` and `_bin.tif`: each full-data model (`_PA<n>_RUN<k>_…` if the CV models are projected).
- `<species>_current_<ALGO>_mean_prob.tif`, `_mean_bin.tif`, `<ALGO>_sd.tif`: algorithm means and spreads (in two-level mode).
- `<species>_current_EMwmean_prob.tif`, `EMca_…`, the extra rules and `EMsd.tif` / `EMcv.tif` / `EMrange.tif`.
- `NT1`, `NT2`, `MIC_*` novelty (ExDet) layers and `projection_options.json` (Chapter 8).

In the output root: `evaluation_scores.csv` (and a copy in `figures/`), `habitus_run_config.json` (CV design, block size per fold, SSB, fold sizes, threshold settings, ensemble members and weights, hold-out ensemble scores), `occurrence_*.csv` and, if searches were run, `tuning/tuning_<algo>.csv` (`tabs/tab_models.py:2056-2240`).

What passes to the next tabs: the models and the ensemble object, the ensemble rules and weights, the two-level mode and the projected replicates (⑤ Future Scenarios uses the same rules, `tabs/tab_models.py:1539-1546`), the thresholds (⑥ Range Change), the map files (the ⑦ Validation layer lists are filled automatically, `tabs/tab_models.py:1853-1857`).

6.11 Expert questions

Q1. How is the spatial block size determined; is it the same as blockCV? Answer: No; it follows the same logic but is simplified. For each predictor at the presence points, HABITUS builds an omnidirectional empirical variogram with 15 classes, takes the median of the outer third of the classes as the sill, takes the distance at which 95% of the sill is first reached as the range, and uses the median across predictors as the block size (`sdm_spatial.py:166-281`). blockCV instead samples from raster cells or points and estimates the range by fitting a variogram model (Valavi et al. 2019). In HABITUS no variogram model is fitted and the value derives only from the predictor values at the presence points; the result may differ from blockCV. If a comparable value is wanted, the range found in blockCV should be entered manually in the `Block size:` field.

Q2. What happens if the block design cannot build enough blocks; do the scores silently revert to a random split? Answer: First the block is halved up to six times and this is noted; if there are still not as many blocks as the requested folds, or fewer than two folds contain both classes, the program falls back to random folds, flags `degraded`, the note starts with `WARNING (design not applied):`, and `[BLOCKING FAILED - scores are optimistic]` is added to the run summary (`sdm_spatial.py:454-495`; `sdm_core.py:5135-5140`, `5933-5934`). There is no silent fallback; however, the user needs to read the log or the report.

Q3. Does the full-data model that draws the map have an independent score? Answer: It does not, and cannot; it is fitted on all the data. The program writes into the `Full` row the mean of the CV fold scores of the same algorithm and PA set (and the mean of the fold thresholds), `Score_source` = `cv_mean` (`sdm_core.py:5818-5853`). It is also this mean that selects and weights the ensemble members. This is similar to the "score of the full model = summary of the CV scores" approach in biomod2.

Q4. Is the binary threshold chosen on the test data? Are the threshold-dependent scores optimistic? Answer: Not with the default `train_cv`: the threshold is chosen on the three-inner-fold out-of-fold predictions of the training rows and applied to the test fold (`sdm_core.py:4876-4927`, `1193-1220`). The `test` option optimizes the threshold on the test fold; the code and tooltip state that this typically inflates TSS by 0.1–0.2. This difference matters when comparing with workflows in R that search for the threshold on the evaluation data, such as `ecospat.max.tss` or biomod2: HABITUS's default TSS may come out systematically lower than theirs.

Q5. Is the ensemble score computed from the ensemble's own map? Answer: No. In each PA × fold group, CV models that have not seen that fold are combined with the ensemble rule and scored only on that fold; member selection and weights come from the scores of the other folds (leave one fold out) (`sdm_core.py:6463-6696`). The resubstitution score from the map (`evaluate`) does not enter this table at all (`sdm_core.py:6698-6706`). This design also brings the selection and weighting step of the ensemble inside the validation; it is more conservative than biomod2's ensemble evaluation.

Q6. Why is there a two-level ensemble; how does it differ from biomod2's EMwmean? Answer: In biomod2, each individual model (PA × run) enters the ensemble as a separate member. HABITUS by default first averages the replicates of an algorithm, then weights the algorithms (`sdm_core.py:6109-6132`, `6359-6376`). In this way an algorithm with many replicates does not dominate, and within-algorithm instability becomes visible in a separate SD map. If the box is unticked, the flat combination is restored; this option can be used for comparison with biomod2, but biomod2's weight definition (including the rank-based `decay`) is not identical.

Q7. How are the weights computed; what happens to a negative score? Answer: $w_i = \max(s_i, 0)^p / \sum_j \max(s_j, 0)^p$; negative scores are set to 0, non-finite ones receive an equal share, and if all weights are zero equal weights are given (`sdm_core.py:5988-6005`). In the example run the TSS means are

0.687 (GLM), 0.566 (RF), 0.611 (GAM), 0.672 (MaxEnt), while the weights are 0.271/0.223/0.241/0.265; that is, at $p = 1$ the weights are close to one another. If more discrimination is wanted, p is increased.

Q8. How is SSB computed, and is it the same as dismo::ssb? Answer: For each fold, the distances of the test presences and test absences to the nearest training presence are taken, pooled and put in a ratio (`sdm_spatial.py:808-881`). dismo::ssb also divides the test-presence→training-presence distance by the test-background→training-presence distance (Hijmans 2012); HABITUS uses the pseudo-absences in the test fold as background and draws a seeded subsample when there are more than 3000 points. The interpretation thresholds (0.7 and 1.3) are specific to HABITUS and are not a published cut-off.

Q9. Are the scores optimistic after a tuning search? Is nested validation performed? Answer: Yes, they are optimistic; no nested validation is performed. The best point is chosen on the same folds that produce the reported scores; the program writes this to the log and the UI with `SELECTION_BIAS_NOTE` and states that the size of the optimism is not measured (`sdm_tuning.py:68-78`; `tabs/tab_tuning.py:396`). The user can partly remove this bias by testing the tuned algorithm with an independent test set (`External test points (CSV)`) or with the independent map in ⑦ Validation.

Q10. Why is ENMeval's delta-AICc criterion absent? Answer: elapid fits MaxEnt as a penalized, weighted logistic regression and does not provide the Maxent likelihood; HABITUS therefore does not compute AICc at all rather than approximating it (`sdm_tuning.py:38-46`, `62-66`). Selection rests on out-of-fold omission and AUC (the complexity warning of Warren and Seifert 2011 is addressed only indirectly in this way).

Q11. With an external test CSV, is the standard deviation across `RUN1…RUN4` a measure of uncertainty? Answer: No. Each run is trained on the same full data and tested on the same test points (`sdm_core.py:5417-5427`); the difference comes only from the algorithm's own randomness. The program writes this as a note in the report (`sdm_core.py:5341-5347`). For uncertainty, the difference between PA sets or blocked CV should be used.

Q12. Which setting changes the map most? Answer: For the probability map, the choice of algorithms and the threshold for entry into the ensemble (§6.3, §6.6.4); for the binary map and area, the threshold rule and the threshold source (§6.5.1–6.5.2), because the threshold on the map is the mean of the thresholds chosen in the CV folds. The CV design does not change the map directly, but through the scores it changes member selection, the weights and the threshold. In the example run the thresholds lie in the 0.06–0.21 range, so small differences in the threshold rule can change the area markedly; if the sensitivity to this is to be measured, the `Classification Thresholds` table in ④ Evaluation can be used (Chapter 7).

6.12 Common mistakes and pitfalls

- Comparing a high AUC obtained with the `random` design with spatially blocked results in the same table. The design is written in the `_cv_strategy` field of every row and in the run record.
- Keeping `Min score to include model:` high, e.g. 0.7 out of AUC habit, and switching the metric to TSS: no model passes and `WARNING - NO COMBINED MAP:` appears. If the user has not typed a value, the default follows the metric; if they have, it does not.
- Lowering `Cross-validation folds:` below 4 with `checkerboard2`: some of the four folds are never tested.
- Expecting the scores to change by changing `Training data %:`; if CV succeeds, they do not change (§6.4.6).
- Reporting in a paper the TSS produced with `Test-fold threshold:` = `test`; the UI shows an orange warning for this choice.
- Taking the orange "test" points on the map for a random test set; they are the points of the first spatial fold.
- Not understanding why the ensemble is not written when `Also project the cross-validation models` is off and only CV models pass the threshold; the message suggests ticking this box.
- Looking for `EMsd` in a single-algorithm run: at least two members are needed.
- Starting the main run without applying the point selected in the tuning search with `Apply selected setting to Models tab`; the search result does not change the settings by itself. The `selected_options_used_in_this_run` field in the run record shows this.

6.13 References

Allouche, O., Tsoar, A., Kadmon, R., 2006. Assessing the accuracy of species distribution models: prevalence, kappa and the true skill statistic (TSS). Journal of Applied Ecology 43, 1223–1232. https://doi.org/10.1111/j.1365-2664.2006.01214.x

Araújo, M.B., New, M., 2007. Ensemble forecasting of species distributions. Trends in Ecology & Evolution 22, 42–47. https://doi.org/10.1016/j.tree.2006.09.010

Boyce, M.S., Vernier, P.R., Nielsen, S.E., Schmiegelow, F.K.A., 2002. Evaluating resource selection functions. Ecological Modelling 157, 281–300. https://doi.org/10.1016/S0304-3800(02)00200-4

Fielding, A.H., Bell, J.F., 1997. A review of methods for the assessment of prediction errors in conservation presence/absence models. Environmental Conservation 24, 38–49. https://doi.org/10.1017/S0376892997000088

Hijmans, R.J., 2012. Cross-validation of species distribution models: removing spatial sorting bias and calibration with a null model. Ecology 93, 679–688. https://doi.org/10.1890/11-0826.1

Hirzel, A.H., Le Lay, G., Helfer, V., Randin, C., Guisan, A., 2006. Evaluating the ability of habitat suitability models to predict species presences. Ecological Modelling 199, 142–152. https://doi.org/10.1016/j.ecolmodel.2006.05.017

Kass, J.M., Muscarella, R., Galante, P.J., Bohl, C.L., Pinilla-Buitrago, G.E., Boria, R.A., Soley-Guardia, M., Anderson, R.P., 2021. ENMeval 2.0: Redesigned for customizable and reproducible modeling of species' niches and distributions. Methods in Ecology and Evolution 12, 1602–1608. https://doi.org/10.1111/2041-210X.13628

Li, W., Guo, Q., 2013. How to assess the prediction accuracy of species presence–absence models without absence data? Ecography 36, 788–799. https://doi.org/10.1111/j.1600-0587.2013.07585.x

Liu, C., Berry, P.M., Dawson, T.P., Pearson, R.G., 2005. Selecting thresholds of occurrence in the prediction of species distributions. Ecography 28, 385–393. https://doi.org/10.1111/j.0906-7590.2005.03957.x

Marmion, M., Parviainen, M., Luoto, M., Heikkinen, R.K., Thuiller, W., 2009. Evaluation of consensus methods in predictive species distribution modelling. Diversity and Distributions 15, 59–69. https://doi.org/10.1111/j.1472-4642.2008.00491.x

Muscarella, R., Galante, P.J., Soley-Guardia, M., Boria, R.A., Kass, J.M., Uriarte, M., Anderson, R.P., 2014. ENMeval: An R package for conducting spatially independent evaluations and estimating optimal model complexity for Maxent ecological niche models. Methods in Ecology and Evolution 5, 1198–1205. https://doi.org/10.1111/2041-210X.12261

Pearson, R.G., Raxworthy, C.J., Nakamura, M., Peterson, A.T., 2007. Predicting species distributions from small numbers of occurrence records: a test case using cryptic geckos in Madagascar. Journal of Biogeography 34, 102–117. https://doi.org/10.1111/j.1365-2699.2006.01594.x

Roberts, D.R., Bahn, V., Ciuti, S., Boyce, M.S., Elith, J., Guillera-Arroita, G., Hauenstein, S., Lahoz-Monfort, J.J., Schröder, B., Thuiller, W., Warton, D.I., Wintle, B.A., Hartig, F., Dormann, C.F., 2017. Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure. Ecography 40, 913–929. https://doi.org/10.1111/ecog.02881

Valavi, R., Elith, J., Lahoz-Monfort, J.J., Guillera-Arroita, G., 2019. blockCV: An R package for generating spatially or environmentally separated folds for k-fold cross-validation of species distribution models. Methods in Ecology and Evolution 10, 225–232. https://doi.org/10.1111/2041-210X.13107

Velazco, S.J.E., Rose, M.B., de Andrade, A.F.A., Minoli, I., Franklin, J., 2022. flexsdm: An R package for supporting a comprehensive and flexible species distribution modelling workflow. Methods in Ecology and Evolution 13, 1661–1669. https://doi.org/10.1111/2041-210X.13874

Warren, D.L., Seifert, S.N., 2011. Ecological niche modeling in Maxent: the importance of model complexity and the performance of model selection criteria. Ecological Applications 21, 335–342. https://doi.org/10.1890/10-1171.1

7. Evaluation (④ Evaluation)

Tab: ④ Evaluation · Code: `tabs/tab_evaluation.py` (UI); `sdm_core.py` `evaluate_all`, `boyce_ecospat`, `compute_calibration`, `cv_roc_curves`, `cv_omission_pr_calibration`, `cv_boyce_curves`, `cv_threshold_performance`, `compute_response_curves`, `evaluation_scores_table` (computation)

7.1 What this tab does

④ Evaluation shows the performance of the models built in ③ Models in charts and tables. It does not build new models or re-split the data: all curves and scores are produced from the **out-of-fold (CV hold-out) predictions** saved during the ③ Models run (`sdm_core.py:5547-5556`, `6826-7037`). Only two things are computed from the full-data (`Full`) models: the response curves (§7.7) and the map thresholds in the `Classification Thresholds` table (§7.9).

- **Input:** the modeller (`modeler`) and ensemble (`ensemble`) objects of ③ Models, the PA sets, the output folder.
- **Output:** seven sub-tabs on screen; PNG figures (300 dpi) in the `figures/` folder, `evaluation_scores.csv`, `evaluation_omission_pr_calibration.csv`, `classification_thresholds.csv` (§7.11).
- **Locking:** unlocked when the ③ Models run finishes (`tabs/tab_models.py:1848`). When the tab is first shown, all charts are computed once, accompanied by a progress window; switching tabs does not recompute for the same run (`tabs/tab_evaluation.py:1344-1371`).

7.2 Screen layout

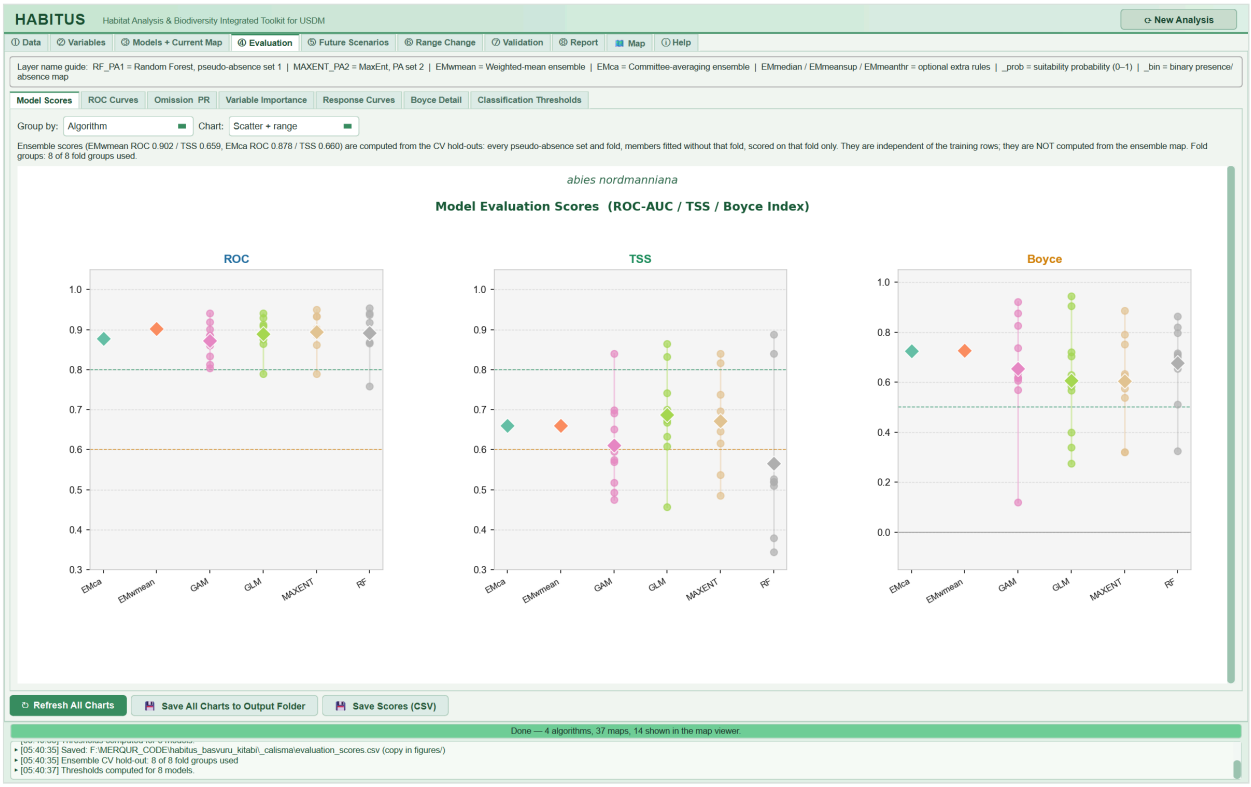


Figure 7.1. ④ Evaluation tab with the Model Scores sub-tab open: the layer name guide at the top, seven sub-tabs in the middle, ROC, TSS and Boyce panels with Group by = Algorithm and Chart = Scatter + range (including the EMca and EMwmean ensemble rows), and the Refresh All Charts, Save All Charts to Output Folder and Save Scores (CSV) buttons at the bottom.

Figure 7.1. ④ Evaluation tab with the Model Scores sub-tab open: the layer name guide at the top, seven sub-tabs in the middle, ROC, TSS and Boyce panels with Group by = Algorithm and Chart = Scatter + range (including the EMca and EMwmean ensemble rows), and the Refresh All Charts, Save All Charts to Output Folder and Save Scores (CSV) buttons at the bottom.

Area	Content	In this chapter
Top strip	`Layer name guide:` meaning of the layer names (`RF_PA1`, `EMwmean`, `_prob`, `_bin` …)	§7.2
`Model Scores`	ROC, TSS, Boyce distributions	§7.3
`ROC Curves`	Out-of-fold ROC curves	§7.4
`Omission & PR`	Omission, precision-recall, calibration	§7.5
`Variable Importance`	Permutation importance	§7.6
`Response Curves`	Response curves (marginal / partial / ice)	§7.7
`Boyce Detail`	Boyce P/E curves per fold	§7.8
`Classification Thresholds`	Thresholds of eleven threshold rules and their out-of-fold performance	§7.9

Area	Content	In this chapter
Bottom buttons	`↻ Refresh All Charts`, `💾 Save All Charts to Output Folder`, `💾 Save Scores (CSV)`	§7.10

In the sub-tab strip, the name of the third sub-tab appears on screen as `Omission PR`; in the code its name is `Omission & PR` (`tabs/tab_evaluation.py:201`). This chapter uses the name in the code.

Multi-row figures (scores, ROC, omission, response, Boyce) sit in their own scroll area; if the figure continues below, the hint `▼ Scroll down for more of this figure` appears (`tabs/tab_evaluation.py:69-101`).

7.3 Model scores (Model Scores)

This sub-tab (Figure 7.1) plots the ROC, TSS and Boyce columns of the `evaluation_scores.csv` table in three panels (`tabs/tab_evaluation.py:229-353`). The table itself contains every metric; the metrics are defined in §7.3.2.

Name in the UI	What it does	Default	Options	Code
`Group by:`	Column used to group the points	`Algorithm`	`Algorithm` / `PA Set` / `CV Run`	`tabs/tab_evaluation.py:233-235`
`Chart:`	Chart type	`Scatter + range`	`Scatter + range` / `Box plot` / `Bar mean±SD`	`tabs/tab_evaluation.py:237-239`
Ensemble note	Writes the hold-out scores and fold groups of the ensemble rows	—	—	`tabs/tab_evaluation.py:261-281`

7.3.1 Chart and grouping

What it does. `Scatter + range` draws each row as a point, the group's minimum–maximum range as a line and the mean as a diamond; `Box plot` draws a box plot; `Bar mean±SD` draws a mean and standard deviation bar (`tabs/tab_evaluation.py:315-331`). The ROC and TSS panels have dashed lines at 0.8 (green) and 0.6 (orange), the Boyce panel lines at 0.5 and 0. If there are more than six groups, the labels are written vertically.

When and how to set it. `Algorithm` compares algorithms; `PA Set` shows the difference between pseudo-absence sets (the effect of background sampling); `CV Run` shows the difference between folds (whether some folds are systematically difficult).

Effect on results. Changes only the view; the table and files do not change.

Caution. The chart uses **all** rows of the table: CV runs, `Full` rows carrying the CV mean, and ensemble rows (`PA Set` = `Ensemble`). Because `Full` rows are copies of the mean, in `Algorithm` grouping one extra mean point per PA set is added to an algorithm's distribution; when reading the spread, rely on the `Score_source` = `cv_fold` rows in `evaluation_scores.csv`. The lower axis limits are 0.3 for ROC and TSS and −0.15 for Boyce (`tabs/tab_evaluation.py:314`); values below these are not visible in the panel.

7.3.2 Metrics in the score table

Each row is computed by `evaluate_all` from a model's predictions (`sdm_core.py:1122-1257`). Threshold-dependent metrics are computed at the threshold found with the threshold rule and threshold source selected in ③ Models (`train_cv` by default) (Chapter 6, §6.5). TP, FN, FP, TN are the cells of the confusion matrix; if a denominator is zero, the metric becomes `NaN`, not 0 (`sdm_core.py:693-719`).

Column	Definition	Threshold-dependent	Code
`ROC`	Area under the ROC curve (AUC)	no	`sdm_core.py:951-959`
`TSS`	Sensitivity + specificity − 1 (Allouche et al. 2006)	yes	`sdm_core.py:671-679`
`Boyce`	Continuous Boyce index (ecospat or legacy definition)	no	`sdm_core.py:475-664`
`Threshold`	Binary threshold used	—	`sdm_core.py:1242`
`Prevalence`, `N_pres`, `N_abs`	Proportion and numbers of presences in the evaluation set	no	`sdm_core.py:1249-1252`
`PredPresFrac`	Proportion of rows above the threshold	yes	`sdm_core.py:1235-1236`
`Brier`	$\overline{(p-y)^2}$; smaller is better	no	`sdm_core.py:998`
`CalSlope`	Slope of the logistic regression of y on $\text{logit}(p)$; 1 good calibration, <1 overconfident (Pearce and Ferrier 2000)	no	`sdm_core.py:1000-1013`
`Sensitivity`, `Specificity`	TP/(TP+FN), TN/(TN+FP)	yes	`sdm_core.py:709-710`
`Kappa`	Cohen's kappa (Fielding and Bell 1997)	yes	`sdm_core.py:711-712`
`MCC`	Matthews correlation coefficient	yes	`sdm_core.py:713-714`
`Jaccard`, `Sorensen`, `FPB`	TP/(TP+FP+FN); 2TP/(2TP+FP+FN); 2 × Jaccard (Li and Guo 2013)	yes	`sdm_core.py:715-719`
`OR_MTP`, `OR_10p`	Proportion of test presences below the lowest / 10th percentile prediction of the training presences	depends on reference threshold	`sdm_core.py:1095-1099`
`AUC_train`, `dAUC`	AUC on the training data; `AUC_train − ROC` (overfitting gap)	no	`sdm_core.py:1074-1084`
`OR_MTP_degenerate`, `OR_10p_degenerate`	1 if the omission threshold is equal to or below the smallest prediction (omission is 0 by construction)	—	`sdm_core.py:1100-1114`
`Threshold_source`, `OR_reference`, `Boyce_formula`	Source of the threshold, the omission reference and the Boyce definition	—	`sdm_core.py:5877-5879`
`Score_source`	`cv_fold`, `cv_mean`, `random_split`, `cv_holdout`	—	`sdm_core.py:1038-1053`

`Score_source` values: `cv_fold` is the score of a CV run on its own held-out fold; `cv_mean` is the CV mean carried over to the full-data model; `random_split` is the random-split score of a full-data model for which no CV score was produced (optimistic); `cv_holdout` is an ensemble row (Chapter 6, §6.6.7).

> **Note:** `compute_calibration` also computes the calibration intercept (`CalIntercept`) and the maximum and expected calibration errors (`MCE`, `ECE`) (`sdm_core.py:981-1033`); these three values are not written to `evaluation_scores.csv`.

A row from the example run (GLM, PA1, RUN1): ROC 0.941, TSS 0.865, Boyce 0.399, threshold 0.177, prevalence 0.148 (43 presences, 248 absences), sensitivity 0.953, specificity 0.911, calibration slope 1.28, `AUC_train` 0.900 (`dAUC` = −0.041). In the RUN2 fold of the same algorithm, ROC is 0.790 and TSS 0.457; the difference between folds is larger than the difference between algorithms.

7.4 ROC curves (ROC Curves)



Figure 7.2. ROC Curves sub-tab: a separate panel for each algorithm and ensemble rule; thin lines are individual folds, the thick line is the mean curve, the shading is the ±1 SD band.

Figure 7.2. ROC Curves sub-tab: a separate panel for each algorithm and ensemble rule; thin lines are individual folds, the thick line is the mean curve, the shading is the ±1 SD band.

What it does. For each algorithm, one ROC curve is drawn from the out-of-fold predictions of each PA set × fold group; the curves are interpolated onto a common 101-point false positive rate grid, and the mean curve and ±1 SD band are computed; the legend has the form `AUC = mean ± SD (n = number of fold groups)` (`cv_roc_curves`, `sdm_core.py:6826-6891`; `tabs/tab_evaluation.py:372-435`). Ensemble rules (`EMwmean (ensemble)` …) are drawn from the ensemble's hold-out surfaces (Chapter 6, §6.6.7). There are no settings.

Effect on results. Screen and `evaluation_roc_curves.png`; the scores do not change. In the example run, AUC over eight fold groups: GAM 0.872 ± 0.050, GLM 0.889 ± 0.047, MaxEnt 0.895 ± 0.051.

Caution. A single-class hold-out is skipped; an algorithm with no out-of-fold predictions at all is listed below the figure with the note `Not shown: …` (`sdm_core.py:6872-6876`; `tabs/tab_evaluation.py:427-429`). Because the absence class consists of pseudo-absences, AUC is the ability to "discriminate presence points from the background"; it is not a measure of performance against true absences.

7.5 Omission, precision-recall and calibration (Omission & PR)

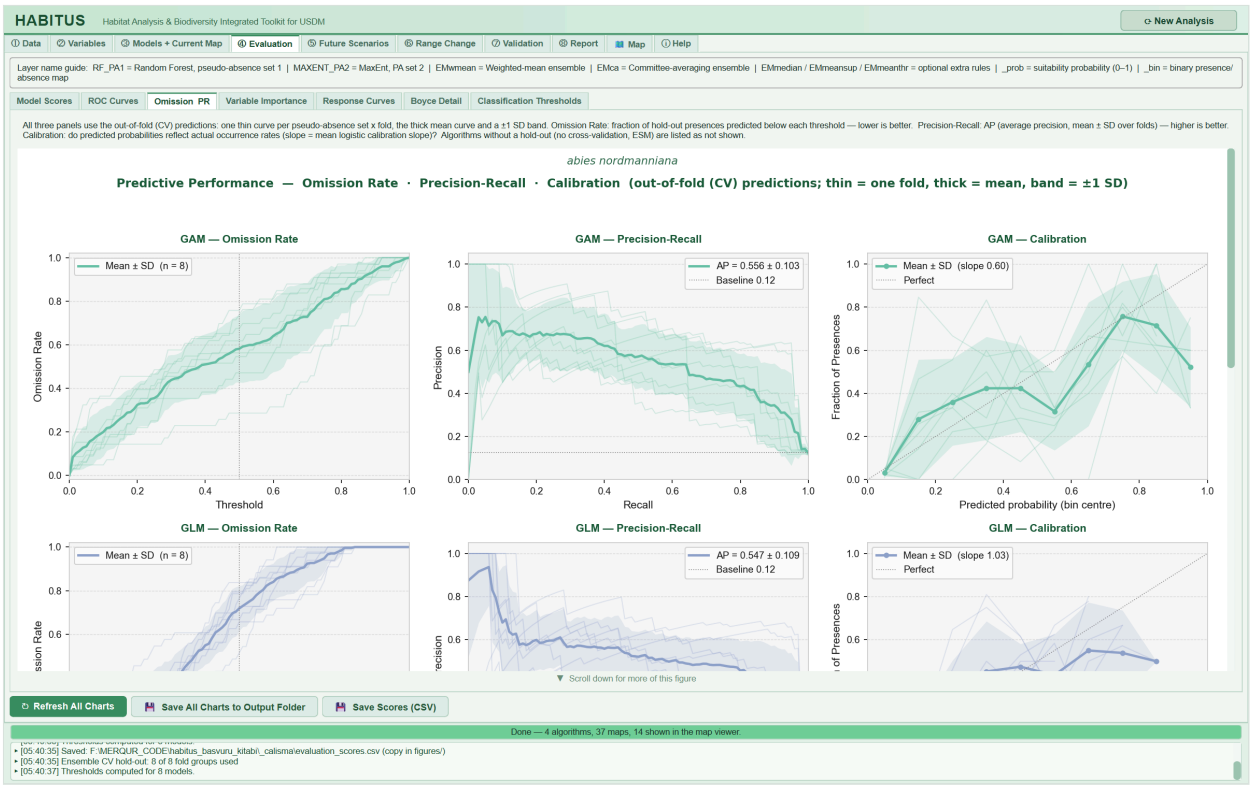


Figure 7.3. Omission & PR sub-tab: omission rate, precision-recall and calibration panels in one row for each algorithm, thin curves per fold and the mean.

Figure 7.3. Omission & PR sub-tab: omission rate, precision-recall and calibration panels in one row for each algorithm, thin curves per fold and the mean.

What it does. For each algorithm, three panels are drawn from the out-of-fold predictions (`cv_omission_pr_calibration`, `sdm_core.py:6926-6995`; `tabs/tab_evaluation.py:459-571`):

- **Omission Rate:** at 101 thresholds between 0 and 1, the proportion of hold-out presences whose prediction falls below the threshold. Lower is better; there is a dashed vertical line at 0.5.
- **Precision-Recall:** the precision-recall curve, interpolated onto a common recall grid; the legend is average precision (AP) ± SD. The grey horizontal line is the baseline precision (mean of the fold prevalences).
- **Calibration:** predictions are divided into 10 equal-width bins; the proportion of presences is computed in each bin (bins with fewer than 3 rows are left empty); the legend is the mean calibration slope of the folds. The diagonal indicates perfect calibration.

Effect on results. Screen, `evaluation_omission_pr_calibration.png`, and `figures/evaluation_omission_pr_calibration.csv`, which contains for each algorithm the mean omission at the 0.5 threshold, AP, baseline precision and mean calibration slope (`tabs/tab_evaluation.py:573-597`).

In the example run, AP is 0.556 ± 0.103 (baseline 0.12) with a mean calibration slope of 0.60 for GAM, and AP 0.547 ± 0.109 with a slope of 1.03 for GLM: the two models discriminate similarly, but GAM's probabilities are overconfident.

Caution. With pseudo-absence data, precision and calibration cannot be interpreted as absolute values: the "absence" class is background, and the probability scale depends on the PA:presence ratio (7:1 in the example run, baseline precision about 0.12). These panels are for comparing algorithms and folds **with each other**. Because the hold-out sets are small (32–43 presences per fold in the example run), the ±1 SD band is not a confidence interval.

7.6 Variable importance (Variable Importance)

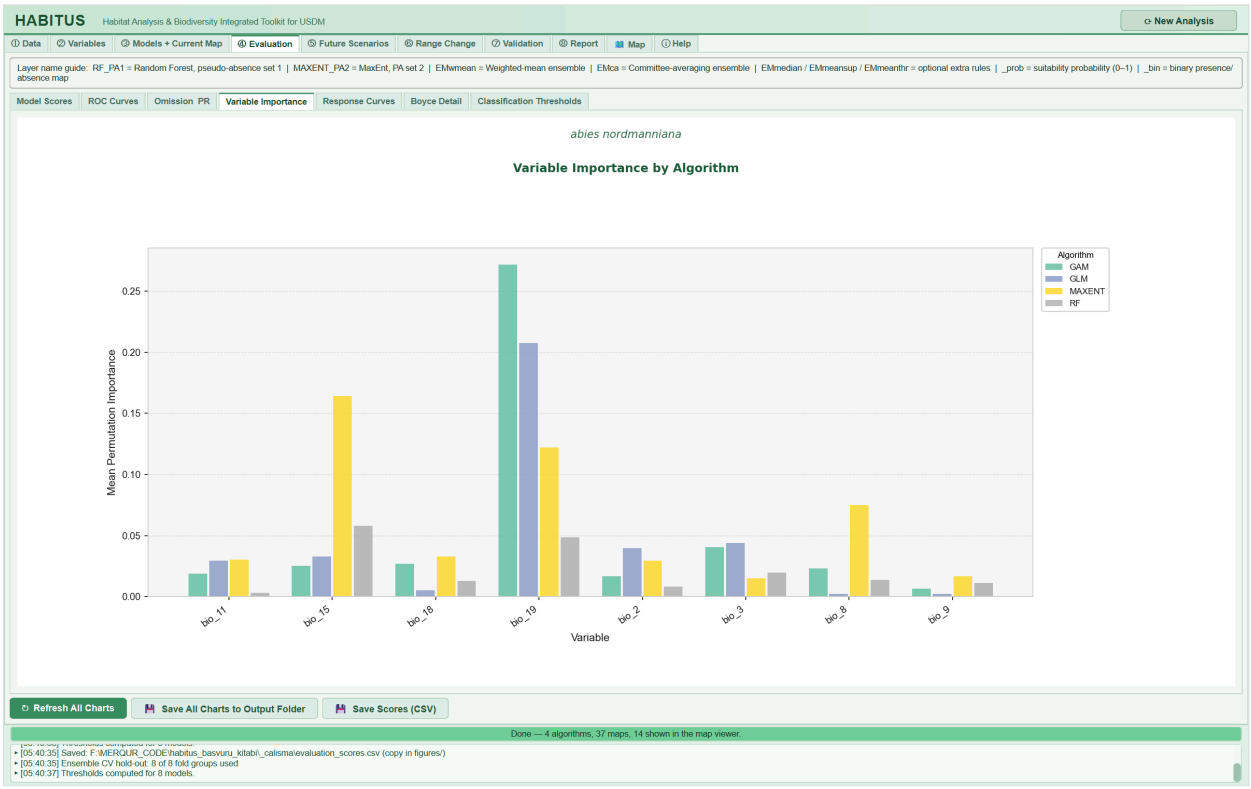


Figure 7.4. Variable Importance sub-tab: bars of mean permutation importance per algorithm for each variable; the legend box is to the right of the chart.

Figure 7.4. Variable Importance sub-tab: bars of mean permutation importance per algorithm for each variable; the legend box is to the right of the chart.

What it does. The permutation importance computed on the test fold in each CV run in ③ Models (the drop in AUC when a predictor is shuffled) is averaged for each algorithm × variable over all folds and PA sets and plotted in a grouped bar chart (`tabs/tab_evaluation.py:607-647`; computation: `sdm_core.py:5562-5636`). There are no settings; the number of repetitions is the `Variable importance reps:` field in ③ Models (Chapter 6, §6.4.7).

When and how to set it. Importance values are on the same scale across algorithms (AUC drop); nevertheless, tree-based and linear models may rank importance differently, so look for consistency across algorithms for a common ranking.

Effect on results. Screen and `evaluation_variable_importance.png`. In the example run, the most important variable in GAM and GLM is bio_19 (AUC drop of about 0.27 and 0.21, respectively), and in MaxEnt bio_15 (about 0.16); in RF no variable exceeds 0.06.

Caution. Correlated predictors share importance: when one is shuffled, the other continues to carry the information, and both may appear unimportant. Importance can be negative (shuffling increased AUC by chance). If `Variable importance reps:` = 0, the chart stays empty. Importance is not computed for full-data models; the chart comes only from CV runs.

7.7 Response curves (Response Curves)

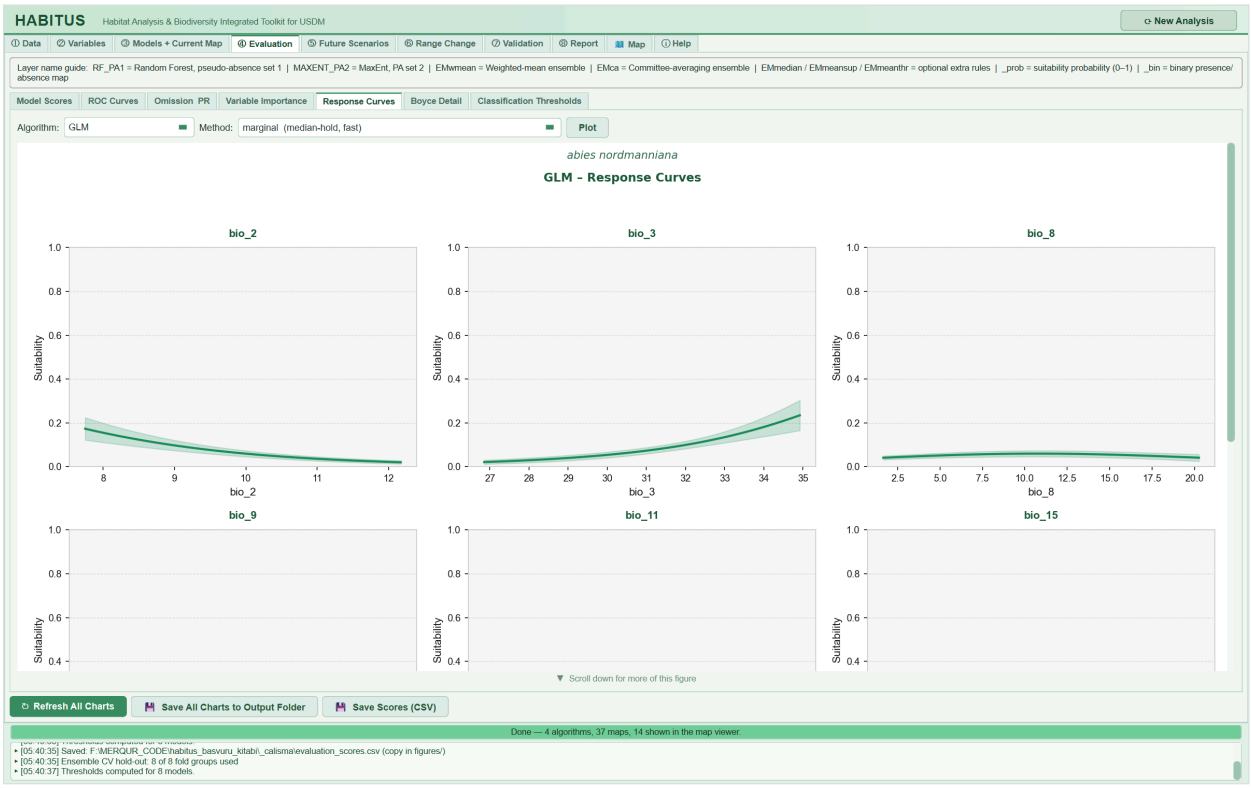


Figure 7.5. Response Curves sub-tab: Algorithm, Method and Plot controls; one panel per continuous predictor for the selected algorithm, the thick line is the mean response, the shading is the standard deviation across replicates; only the upper part of the second-row panels is visible, the curves lie in the region below 0.4 that becomes visible on scrolling.

Figure 7.5. Response Curves sub-tab: Algorithm, Method and Plot controls; one panel per continuous predictor for the selected algorithm, the thick line is the mean response, the shading is the standard deviation across replicates; only the upper part of the second-row panels is visible, the curves lie in the region below 0.4 that becomes visible on scrolling.

Name in the UI	What it does	Default	Options	Code
`Algorithm:`	Algorithm whose curves are drawn	first algorithm	algorithms in the run	`tabs/tab_evaluation.py: 654-655, 885-892`

Name in the UI	What it does	Default	Options	Code
`Method:`	Response curve method	`marginal (median-hold, fast)`	`marginal` / `partial` / `ice`	`tabs/tab_evaluation.py: 656-671`
`Plot`	Computes and draws the curves	—	—	`tabs/tab_evaluation.py: 672-673`

7.7.1 Response curve method (Method)

What it does. Each continuous predictor is varied at 100 points between its 5th and 95th percentiles in the pooled data of all PA sets (presences and pseudo-absences) (`compute_response_curves`, `sdm_core.py:8354-8510`):

- `marginal`: all other predictors are held at the median of the pooled data (the "median-hold" approach of biomod2).
- `partial`: partial dependence (PDP; Friedman 2001): in at most 500 seeded randomly selected real rows, the focal variable is set to each grid value and the predictions are averaged.
- `ice`: individual conditional expectation (ICE; Goldstein et al. 2015): a separate curve (thin lines) is drawn for each of 60 seeded randomly selected rows; the thick line is the mean of these curves, i.e. the PDP of those rows.

In every method, **all fitted models** of the algorithm (the CV runs of each PA set and the full-data model) are predicted separately; the thick line is their mean and the shading their standard deviation (`sdm_core.py:8461-8506`). For categorical predictors, one bar is drawn per class (other predictors at the median; variables with 2–60 classes) (`sdm_core.py:8512-8553`).

When and how to set it. `marginal` is fast and comparable with biomod2/maxnet outputs. If there are strong interactions or correlations between predictors, the median row may be an unrealistic combination of predictors; `partial` averages over the real data distribution. `ice` is used to see interactions: thin curves of differing shapes indicate that the effect of the focal variable depends on other variables.

Effect on results. Screen and `evaluation_response_<ALGO>.png` only. `partial` requires about 500 × 100 predictions, `ice` 60 × 100 predictions × the number of models; it is noticeably slow for MaxEnt and RF. The result is cached for the same run, method and algorithm; redrawing does not recompute (`tabs/tab_evaluation.py:690-712`).

Caution. The curves are computed without applying the `Clamp predictors …` setting of ③ Models (the prediction functions are called directly, `sdm_core.py:8419-8437`). The y axis is fixed to the range 0–1; with pseudo-absence data the probability scale is low, so the curves may look compressed (in the example run the GLM curves stay below 0.25). The shading of a curve shows the difference between CV and full-data models; it is not a confidence interval. The figure has 3 columns; with more than nine predictors, scroll the figure for the lower rows.

7.8 Boyce detail (Boyce Detail)

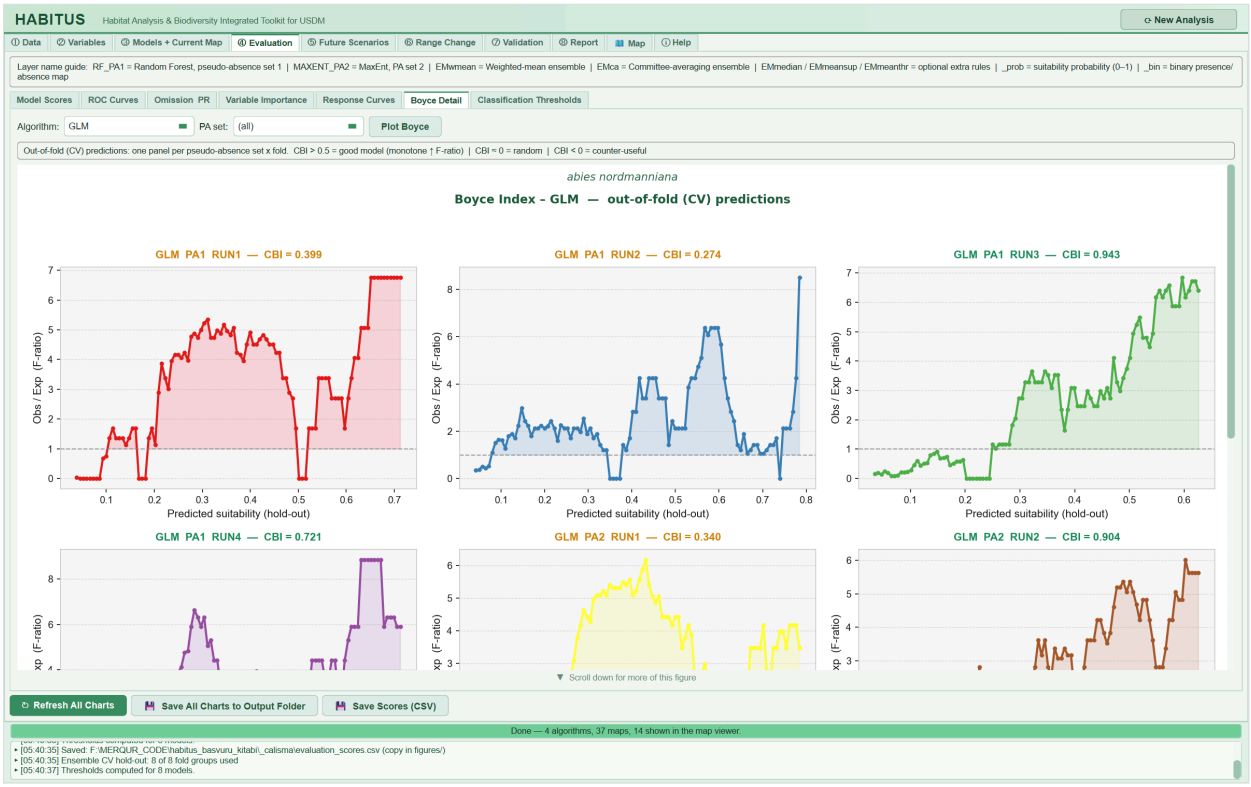


Figure 7.6. Boyce Detail sub-tab: one panel per PA set × fold group for the selected algorithm; P/E (F-ratio) curve, dashed line at level 1 and the CBI value of that fold in the title.

Figure 7.6. Boyce Detail sub-tab: one panel per PA set × fold group for the selected algorithm; P/E (F-ratio) curve, dashed line at level 1 and the CBI value of that fold in the title.

Name in the UI	What it does	Default	Options	Code
`Algorithm:`	Algorithm	first algorithm	algorithms in the run	`tabs/tab_evaluation.py:774`
`PA set:`	PA set to display	`(all)`	`(all)`, `PA1`, `PA2`, …	`tabs/tab_evaluation.py:775`, `893-897`
`Plot Boyce`	Draws the panels	—	—	`tabs/tab_evaluation.py:776-777`

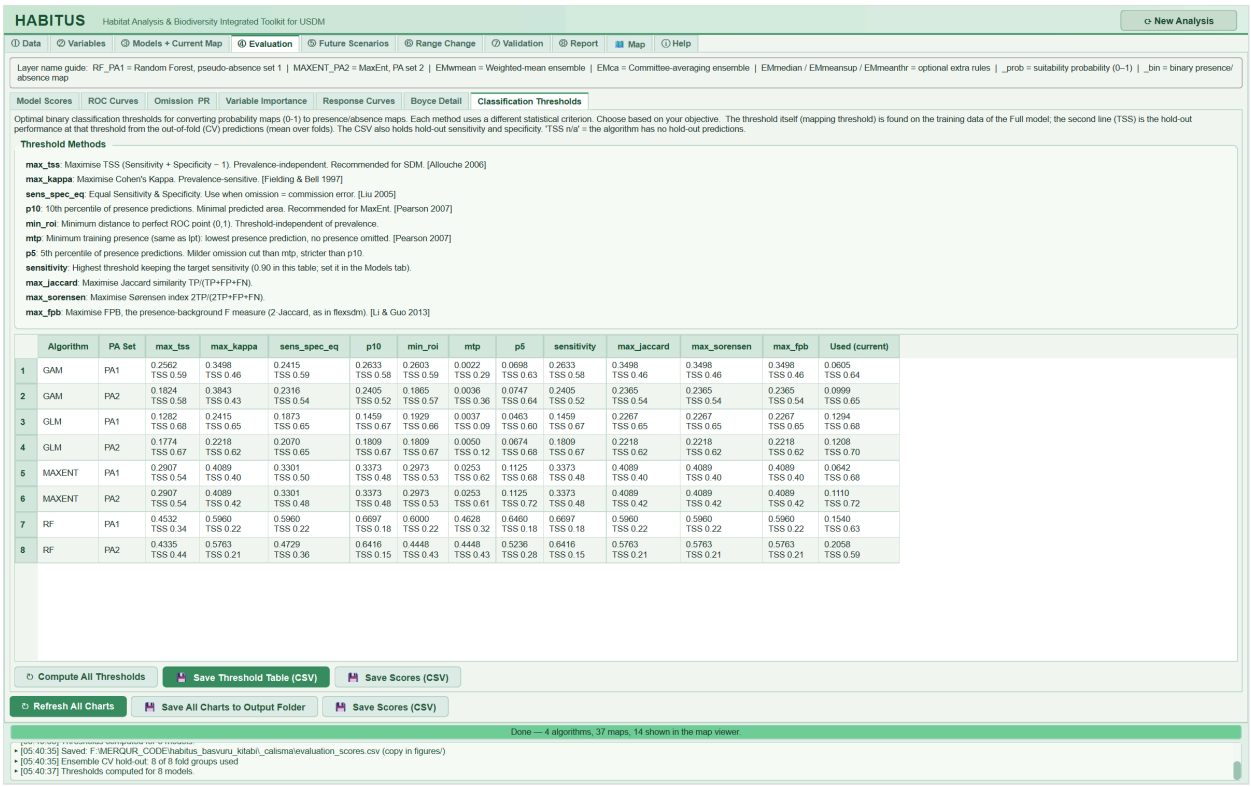
What it does. For each PA set × fold, a Boyce curve is drawn from the hold-out predictions: the x axis is the window midpoint, the y axis the ratio $F = P_i/E_i$; the region above 1 is shaded; the title colour is green for CBI ≥ 0.5, orange for 0–0.5 and red below 0 (`cv_boyce_curves`, `sdm_core.py:6998-7015`; `tabs/tab_evaluation.py:791-847`). The function called is the same call that produces the Boyce value in the score table; the CBI in the panel title matches the value in the table.

When and how to set it. In a good model, the P/E curve rises monotonically with suitability and lies below 1 at low suitability and above 1 at high suitability (Hirzel et al. 2006). A curve that falls after a peak indicates that the model assigns its highest values to too large an area; a flat curve indicates weak discrimination.

Effect on results. Screen and `evaluation_boyce_<ALGO>.png`. In the example run, the fold CBI values of GLM range from 0.274 (PA1 RUN2) to 0.943 (PA1 RUN3); in the RUN1 and RUN2 panels the P/E curve peaks at intermediate suitability and then falls.

Caution. The window width is one tenth of the prediction range (ecospat definition); in models with a very narrow prediction range (e.g. BIOCLIM) the windows fall very close to each other. If the number of presences per fold is small, the curve is stepped; read the CBI as the mean over folds, not from a single fold.

7.9 Classification thresholds (Classification Thresholds)



	Algorithm	PA Set	max_tss	max_kappa	sens_spec_eq	p10	min_roi	mtp	p5	sensitivity	max_jaccard	max_sorensen	max_fpb	Used (current)
1	GAM	PA1	0.2562 TSS 0.59	0.3498 TSS 0.46	0.2415 TSS 0.59	0.2633 TSS 0.58	0.2603 TSS 0.59	0.0022 TSS 0.29	0.0698 TSS 0.63	0.2633 TSS 0.58	0.3498 TSS 0.46	0.3498 TSS 0.46	0.3498 TSS 0.46	0.0605 TSS 0.64
2	GAM	PA2	0.1824 TSS 0.58	0.3843 TSS 0.43	0.2316 TSS 0.54	0.2405 TSS 0.52	0.1865 TSS 0.57	0.0036 TSS 0.36	0.0747 TSS 0.64	0.2405 TSS 0.52	0.2365 TSS 0.54	0.2365 TSS 0.54	0.2365 TSS 0.54	0.0999 TSS 0.65
3	GLM	PA1	0.1282 TSS 0.68	0.2415 TSS 0.65	0.1873 TSS 0.65	0.1459 TSS 0.67	0.1929 TSS 0.66	0.0037 TSS 0.09	0.0463 TSS 0.60	0.1459 TSS 0.67	0.2267 TSS 0.65	0.2267 TSS 0.65	0.2267 TSS 0.65	0.1294 TSS 0.68
4	GLM	PA2	0.1774 TSS 0.67	0.2218 TSS 0.62	0.2070 TSS 0.65	0.1809 TSS 0.67	0.1809 TSS 0.67	0.0050 TSS 0.12	0.0674 TSS 0.68	0.1809 TSS 0.67	0.2218 TSS 0.62	0.2218 TSS 0.62	0.2218 TSS 0.62	0.1208 TSS 0.70
5	MAXENT	PA1	0.2907 TSS 0.54	0.4089 TSS 0.40	0.3301 TSS 0.50	0.3373 TSS 0.48	0.2973 TSS 0.53	0.0253 TSS 0.62	0.1125 TSS 0.68	0.3373 TSS 0.48	0.4089 TSS 0.40	0.4089 TSS 0.40	0.4089 TSS 0.40	0.0642 TSS 0.68
6	MAXENT	PA2	0.2907 TSS 0.54	0.4089 TSS 0.42	0.3301 TSS 0.48	0.3373 TSS 0.48	0.2973 TSS 0.53	0.0253 TSS 0.61	0.1125 TSS 0.72	0.3373 TSS 0.48	0.4089 TSS 0.42	0.4089 TSS 0.42	0.4089 TSS 0.42	0.1110 TSS 0.72
7	RF	PA1	0.4532 TSS 0.34	0.5960 TSS 0.22	0.5960 TSS 0.22	0.6697 TSS 0.18	0.6000 TSS 0.22	0.4628 TSS 0.32	0.6460 TSS 0.18	0.6697 TSS 0.18	0.5960 TSS 0.22	0.5960 TSS 0.22	0.5960 TSS 0.22	0.1540 TSS 0.63
8	RF	PA2	0.4335 TSS 0.44	0.5763 TSS 0.21	0.4729 TSS 0.36	0.6416 TSS 0.15	0.4448 TSS 0.43	0.4448 TSS 0.43	0.5236 TSS 0.28	0.6416 TSS 0.15	0.5763 TSS 0.21	0.5763 TSS 0.21	0.5763 TSS 0.21	0.2058 TSS 0.59

Figure 7.7. Classification Thresholds sub-tab: descriptions of the threshold rules and, for each full-data model, the threshold of each of the eleven rules (upper line) with the out-of-fold TSS at that threshold (lower line); the last column is the threshold used in the map.

Figure 7.7. Classification Thresholds sub-tab: descriptions of the threshold rules and, for each full-data model, the threshold of each of the eleven rules (upper line) with the out-of-fold TSS at that threshold (lower line); the last column is the threshold used in the map.

Name in the UI	What it does	Code
`Threshold Methods` box	Short definitions of the eleven rules	`tabs/tab_evaluation.py: 1132-1160`
Table	Rows: full-data model (algorithm × PA set); columns: 11 rules and `Used (current)`	`tabs/tab_evaluation.py: 1162-1180`
`↻ Compute All Thresholds`	Recomputes the table	`tabs/tab_evaluation.py: 1183-1185`

Name in the UI	What it does	Code
💾 Save Threshold Table (CSV)	Writes figures/classification_thresholds.csv	tabs/tab_evaluation.py:1316-1342
💾 Save Scores (CSV)	Writes evaluation_scores.csv (§7.10)	tabs/tab_evaluation.py:1188-1194

What it does. For each full-data model, a **map threshold** is found with each of the eleven rules (Chapter 6, §6.5.1), and two lines are written in the cell: the threshold and the mean over folds of the TSS this threshold gives on the out-of-fold predictions (tabs/tab_evaluation.py:1200-1314; sdm_core.py:7018-7037). The tooltip additionally shows the out-of-fold sensitivity and specificity at that threshold, the criterion value on the training data and the rule "pixel >= t -> Present (1)". The Used (current) column is the threshold actually used in the map. The CSV contains, for each rule, the threshold and the _holdout_Sens, _holdout_Spec, _holdout_TSS columns, the threshold used, the name of the rule and the number of folds.

When and how to set it. To support the choice of threshold rule: you see which rule gives a wider or narrower map while preserving out-of-fold TSS. The sensitivity column is always computed with a target of 0.90 (method description).

Effect on results. This table does not change the maps; the maps change when the rule is changed in ③ Models and the models are rerun.

Caution. The thresholds in this table and the threshold used in the map are found in different ways:

- The rule columns of the table are found from the full-data model's own (seen in training) predictions on the **pooled data of all PA sets** (tabs/tab_evaluation.py:1217-1259).
- Used (current), in contrast, is the mean of the thresholds found in the CV folds in ③ Models with the selected threshold source (default train_cv) (sdm_core.py:5852-5853).

For this reason the max_tss column and Used (current) differ even for the same rule. In the example run, for GAM PA1 the max_tss column is 0.2562 (out-of-fold TSS 0.59) and the threshold used is 0.0605 (out-of-fold TSS 0.64). The difference grows for algorithms that memorise the training data: for RF PA1 the max_tss column is 0.4532 (out-of-fold TSS 0.34) and the threshold used is 0.1540 (out-of-fold TSS 0.63). To compare rules, look at the out-of-fold TSS in the lower line; do not read the threshold values as the map threshold.

7.10 Bottom buttons: refresh, save

Name in the UI	What it does	Code
↻ Refresh All Charts	Updates the drop-down lists and recomputes all seven sub-tabs	tabs/tab_evaluation.py:851-872
💾 Save All Charts to Output Folder	Writes all figures and the score table to the figures/ folder; can be cancelled	tabs/tab_evaluation.py:939-1058
💾 Save Scores (CSV)	Writes evaluation_scores.csv to the output root and a copy to figures/	tabs/tab_evaluation.py:1064-1108

What it does. Save All Charts … writes, in order, the score, ROC, omission/PR/calibration and importance figures; then, for each algorithm with CV models, the response curve figure (with the currently selected Method:) and the Boyce figure; and finally the score table (tabs/tab_evaluation.py:993-1030). Each figure is saved once as a 300 dpi PNG (tabs/tab_evaluation.py:923-937). When a figure is drawn on screen, it is also saved under the same name; therefore, before the button is pressed, the figures/ folder contains only the

response and Boyce figures of the displayed algorithm (in the example run only `evaluation_response_GLM.png` and `evaluation_boyce_GLM.png`).

Caution. `evaluation_scores.csv` is written automatically when ③ Models finishes, when the tab is first opened and when the ensemble is rebuilt (`tabs/tab_evaluation.py:1071-1081`; `tabs/tab_models.py:1840-1845`); `Save Scores (CSV)` is only for repeating this manually. If there is no output folder, figures cannot be saved and `"Output directory not set — cannot save figure."` is written to the log. The information window after saving mentions a `figures/response_curves/` subfolder; no such folder is created, and a single combined figure is written per algorithm.

7.11 Outputs and what passes to the next step

In the `<output>/figures/` folder:

- `evaluation_model_scores.png`, `evaluation_roc_curves.png`, `evaluation_omission_pr_calibration.png`, `evaluation_variable_importance.png`
- `evaluation_response_<ALGO>.png`, `evaluation_boyce_<ALGO>.png`
- `evaluation_omission_pr_calibration.csv`, `classification_thresholds.csv`, `evaluation_scores.csv` (copy)

`evaluation_scores.csv` in the output root. This tab does not change models, thresholds or maps; ⑧ Report uses the same score table and figures (Chapter 11).

7.12 Expert questions

Q1. Are the ROC, omission and Boyce curves on this tab based on independent data? Is the model tested on its training data? Answer: The curves are drawn only from the CV hold-out predictions saved in ③ Models (`modeler.cv_holdout`; `sdm_core.py:5547-5556`); no new split is made and no predictions are made on training rows (`sdm_core.py:6926-6932`). The degree of independence depends on the CV design in ③ Models: with spatial blocks, the tests are geographically separate from training; with the `random` design they are not. The exceptions are the response curves and the threshold columns of the `Classification Thresholds` table.

Q2. Is the AUC reported for the ensemble (EMwmean) computed from the ensemble map? Answer: No. The ensemble rows and ROC panels are produced by combining, in each fold, the CV models that have not seen that fold with the ensemble rule and scoring them only on that fold; member selection and weights come from the scores of the other folds (`sdm_core.py:6516-6696`). The note on the scores sub-tab states this explicitly: `"… they are NOT computed from the ensemble map."` (`tabs/tab_evaluation.py:269-274`).

Q3. At how many thresholds and on which data is TSS computed; is it comparable with ecospat or biomod2? Answer: TSS is computed on the test fold, at the threshold that the selected rule (default `max_tss`) finds on the inner out-of-fold predictions of the training rows (`sdm_core.py:1193-1217`). biomod2 and `ecospat.max.tss` usually optimise the threshold on the evaluation data itself; this "oracle" threshold raises TSS. If `Test-fold threshold: = test` is selected in ③ Models, HABITUS follows the same path and the numbers become comparable; however, this choice is optimistic (Chapter 6, §6.5.2).

Q4. Is the Boyce index exactly the same as in ecospat? Answer: The default `ecospat` formula is a line-by-line port of the `ecospat.boyce` function (ecospat 4.1.4): window width 1/10 of the prediction range, 101 positions, closed windows, removal of consecutive repeated P/E values, and Spearman correlation (`sdm_core.py:475-543`). The difference lies in the choice of the `fit` set: HABITUS uses all rows of the evaluation fold (test presences and test pseudo-absences) (`sdm_core.py:605-610`); in ecospat, `fit` is in most

applications the raster predictions of the whole study area. The values therefore match exactly for the same predictions, and approximately for the full map.

Q5. How is the calibration slope computed, and what does it mean with pseudo-absence data? Answer: It is the slope of the logistic regression (practically unpenalised, `C = 1e12`) of the observation on the logit of the prediction; predictions are clipped to the range $[10^{-6}, 1 - 10^{-6}]$, and at least 10 rows and two classes are required (`sdm_core.py:989-1013`; Pearce and Ferrier 2000). With pseudo-absence data, the scale of the probability depends on the PA:presence ratio; the slope is an indicator not of the true probability of presence but of the model's over- or underconfidence on out-of-fold data. A slope much smaller than 1 (overfitting) is meaningful in comparisons between algorithms.

Q6. Why does OR_10p sometimes come out as 0, and is this a good result? Answer: Not always. If the omission threshold is equal to or below the lowest prediction of the evaluation set, no presence can fall below the threshold; omission is 0 by construction. This case is flagged with `OR_10p_degenerate` = 1; if it occurs in at least half of the folds, a warning is written for the algorithm (`sdm_core.py:1100-1114`, `5787-5816`). It is usually seen with BIOCLIM, shallow CART and strongly regularised GLMNET; the omission of these algorithms should not be compared with that of the others.

Q7. Which method is used to compute variable importance; is it like MaxEnt's percent contribution? Answer: No, it is permutation importance: a predictor is shuffled on the test fold and the drop in AUC is measured (`sdm_core.py:5564-5636`). MaxEnt's "percent contribution" and "permutation importance" outputs are normalised to percentages and computed on the training data; HABITUS does not normalise and computes on the test fold. For scikit-learn models, because of the fixed seed, the `Variable importance reps:` repetitions give the same result; the effective repetition is `n_repeats` = 3.

Q8. From which models and which data are the response curves produced? Answer: From all fitted models of the algorithm (the CV runs of each PA set and the full-data model); the thick line is their mean, the shading their standard deviation (`sdm_core.py:8461-8506`). The range of the variable and the median row are taken from the pooled presence and pseudo-absence rows of all PA sets; the range is the 5th–95th percentile, and there is no extrapolation at the ends (`sdm_core.py:8405-8457`). The default approach in biomod2's `bm_PlotResponseCurves` function is also to hold the other variables at a fixed value (the median); because the variable range and summarising details are not the same in the two programs, the curves may be similar in shape but not numerically identical.

Q9. When do marginal, PDP and ICE give different results? Answer: When there are interactions or strong correlations between predictors. Marginal shows only the neighbourhood of the median row; PDP averages over real data rows (Friedman 2001); ICE shows the curve of each row separately and reveals the heterogeneity that PDP hides by averaging (Goldstein et al. 2015). In a linear additive model (e.g. a GLM without interactions) all three give the same shape, though their levels may differ; they diverge in tree models and in GLMs with `interactions` terms.

Q10. Why is `max_tss` in the Classification Thresholds table not the same as the threshold in the map? Answer: The table finds the threshold from the full-data model's predictions (seen in training) on the pooled data of all PA sets (`tabs/tab_evaluation.py:1217-1259`); the threshold in the map, in contrast, is the mean of the thresholds found in the CV folds according to the threshold source (`sdm_core.py:5852-5853`). In the example run, for RF PA1 the table gives 0.4532 and the map 0.1540; because RF's training predictions are close to 1 at presences, the table threshold comes out high and out-of-fold TSS drops to 0.34, whereas the map's threshold gives 0.63. The reliable part of the table is the out-of-fold TSS in the lower line.

Q11. Is the difference between algorithms significant? Is a test provided? Answer: No, there is no statistical comparison test. The program gives the mean, SD and range over folds and PA sets (`sdm_core.py:5883-5919`), and the charts show the distribution. Because the folds derive from the same data,

they are not independent; a simple t test would not be valid either. In the example run, the difference between folds of the same algorithm (GLM PA1 ROC 0.790–0.941) is larger than the difference between algorithms; in this situation it is more defensible to include several algorithms in the ensemble than to rank them.

Q12. How does an algorithm run without ESM or CV appear on this tab? Answer: Algorithms without out-of-fold predictions are listed with a "Not shown" note in the ROC, omission and Boyce figures (`sdm_core.py:6872-6876`, `6974-6977`); the `Classification Thresholds` cells show `TSS n/a`. ESM's hold-out predictions are saved and appear normally; only because it does not produce inner out-of-fold training predictions may its threshold source be labelled differently (Chapter 6, §6.5.2).

7.13 Common mistakes and pitfalls

- Taking the distribution in the `Model Scores` chart for the spread between folds: the chart also includes `Full` (CV mean) and ensemble rows.
- Not being able to see a model with TSS below 0.3 in the chart: the axis starts at 0.3; read the value in the table or CSV.
- Taking the threshold values in the `Classification Thresholds` table for the map threshold; the map threshold is the `Used (current)` column.
- Interpreting calibration and precision values as true probabilities of presence; the scale depends on the PA:presence ratio.
- Reporting the shading of a response curve as a confidence interval; it is the SD across replicates.
- Expecting response and Boyce figures for every algorithm in the `figures/` folder without using `Save All Charts …`.
- Running with `Variable importance reps:` = 0 and searching for why the importance chart is empty.
- Assuming that clamping (`Clamp`) is applied to the response curves.

7.14 References

Allouche, O., Tsoar, A., Kadmon, R., 2006. Assessing the accuracy of species distribution models: prevalence, kappa and the true skill statistic (TSS). Journal of Applied Ecology 43, 1223–1232. https://doi.org/10.1111/j.1365-2664.2006.01214.x

Fielding, A.H., Bell, J.F., 1997. A review of methods for the assessment of prediction errors in conservation presence/absence models. Environmental Conservation 24, 38–49. https://doi.org/10.1017/S0376892997000088

Friedman, J.H., 2001. Greedy function approximation: A gradient boosting machine. The Annals of Statistics 29, 1189–1232. https://doi.org/10.1214/aos/1013203451

Goldstein, A., Kapelner, A., Bleich, J., Pitkin, E., 2015. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. Journal of Computational and Graphical Statistics 24, 44–65. https://doi.org/10.1080/10618600.2014.907095

Hirzel, A.H., Le Lay, G., Helfer, V., Randin, C., Guisan, A., 2006. Evaluating the ability of habitat suitability models to predict species presences. Ecological Modelling 199, 142–152. https://doi.org/10.1016/j.ecolmodel.2006.05.017

Li, W., Guo, Q., 2013. How to assess the prediction accuracy of species presence–absence models without absence data? Ecography 36, 788–799. https://doi.org/10.1111/j.1600-0587.2013.07585.x

Pearce, J., Ferrier, S., 2000. Evaluating the predictive performance of habitat models developed using logistic regression. Ecological Modelling 133, 225–245. https://doi.org/10.1016/S0304-3800(00)00322-7

8. Future scenarios (⑤ Future Scenarios)

Tab: ⑤ Future Scenarios · Code: `tabs/tab_projection.py` (UI), `sdm_core.py` `Projector` (computation), `sdm_novelty.py` (ExDet, MESS), `sdm_grid.py` (grid alignment), `sdm_mask.py` (clipping to the study boundary)

8.1 What this tab does

The ⑤ Future Scenarios tab applies the models built in ③ Models to another stack of environmental layers; this is called **projection** (transfer). The most common use is to apply a model built with current climate to the future-period layers of a climate model (GCM, general circulation model). The same tab is also used to transfer the model to another region or to a past period: the program only expects "continuous rasters with the same names and in the same order as the variables used in training".

The tab is unlocked when ③ Models has finished (`tabs/tab_models.py:1849`). The map for current climate has already been produced in ③ (the `current\` subfolder of the output folder, `tabs/tab_models.py:1719`); ⑤ produces the future counterpart of this map.

- **Input:** one continuous raster for each model variable (scenario layers), the selection of algorithms to project, and the optional MESS/MoD, MESS mask and clamping options.
- **Computation:** each full-data model (each algorithm × pseudo-absence set) is applied to the new layers; within-algorithm mean and spread maps, ensemble maps, binary (0/1) maps and ExDet novelty surfaces are written (`sdm_core.py:7733-8156`).
- **Output:** GeoTIFF files and the `<species>_<scenario>_projection_options.json` record in the scenario subfolder; the list of files in the `Completed Projections` table; the maps in the map viewer on the right.
- **Tab unlocked:** when the job finishes, ⑥ Range Change is unlocked (`tabs/tab_projection.py:672`).

8.2 Screen layout

The tab has two panes: the settings on the left, the map viewer and the `▶ Run Projection` button on the right (`tabs/tab_projection.py:98`, `ui_style.two_pane`).

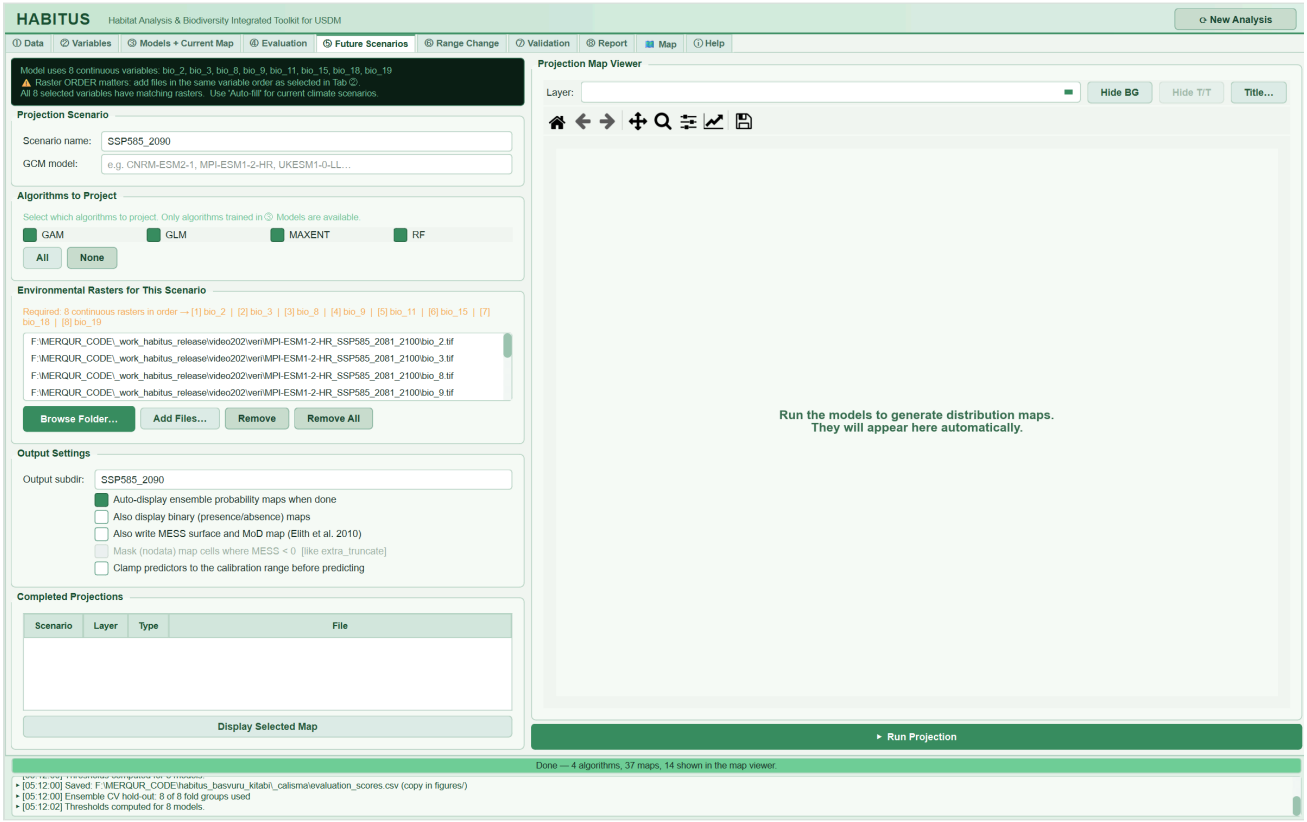


Figure 8.1. The Future Scenarios tab before running: at the top the information banner showing the eight variables expected by the model and the order warning; on the left Projection Scenario, Algorithms to Project, Environmental Rasters for This Scenario, Output Settings and the empty Completed Projections table; on the right the empty Projection Map Viewer.

The regions of the left pane, from top to bottom, are:

Region	Name in the UI	Described in
Information banner	(untitled, dark green box)	§8.3
Scenario name and GCM	`Projection Scenario`	§8.4
Algorithm selection	`Algorithms to Project`	§8.5
Raster list	`Environmental Rasters for This Scenario`	§8.6
Output and method options	`Output Settings`	§8.7, §8.8
Completed outputs	`Completed Projections`	§8.11
Map	`Projection Map Viewer` (right pane)	§8.12

8.3 Information banner

The dark box at the top of the tab is rewritten every time the tab is shown (`tabs/tab_projection.py:308-345`). It gives three pieces of information:

1. The continuous variables used by the model (`Model uses 8 continuous variables: bio_2, bio_3, …`).
2. The order warning: `⚠ Raster ORDER matters: add files in the same variable order as selected in Tab ②.`
3. Whether the training rasters match the variables (`All 8 selected variables have matching rasters.` or `Warning: only k/n variables matched …`). If there are categorical variables, a line is added saying that they are not carried into the future and that the current layer is reused.

The orange line immediately below (`Required: 8 continuous rasters in order → [1] bio_2 | [2] bio_3 | …`) shows the order of the files to be added, numbered (`tabs/tab_projection.py:340-343`).

> **Note:** There is no button in this tab corresponding to the phrase `Use 'Auto-fill' for current climate scenarios.` that appears in the banner. The current climate map is produced automatically in ③ Models; a user who wants to re-project current climate with other settings (e.g. with MESS on) selects the folder of the training rasters with `Browse Folder…` (§8.6.2).

8.4 Scenario name and GCM (Projection Scenario)

Name in the UI	What it does	Default	Range / options	Code
`Scenario name:`	Name of the projection; enters the file names, the subfolder, the report and the list in ⑥	`2050_SSP245`	free text; `current` if empty	`tabs/tab_projection.py:115`, `:505`
`GCM model:`	Climate model name to be written in the map title	empty	free text	`tabs/tab_projection.py:119`, `:31-44`

8.4.1 Scenario name

- **What it does.** The scenario name (`proj_name`) enters the name of every output file: `<species>_<scenario>_<layer>_prob.tif` (`sdm_core.py:7895`). If `Output subdir:` is empty, it is also the name of the output folder (`tabs/tab_projection.py:544`). The name is kept as a key in the session state (`state["projections"][name]`); the ⑥ Range Change lists, the novelty table in the report and the multi-species batch run work with this key (`tabs/tab_projection.py:580`, `:636`).
- **When and how to set it.** The name should state both the period and the scenario: `SSP585_2090`, `SSP245_2050_MPI`. If more than one GCM is used, include the GCM in the name as well; otherwise the outputs of two GCMs are written to the same folder and the same file names.
- **Effect on results.** It has no effect on the computation; it determines the organisation of files and records.
- **Caution.** If an SSP5-8.5 2090 layer is projected without changing the default text `2050_SSP245`, the files are written with the wrong name; the program does not check the name against the content. Running a second time with the same name overwrites the same files and duplicates the rows in the table. Naming it `current` overwrites the current maps produced by ③ (the same `current\` folder) and the `current` record in the session.

8.4.2 GCM name

- **What it does.** It is only added to the map title in the viewer (`Future Distribution · MPI-ESM1-2-HR · SSP585_2090 · …`, `tabs/tab_projection.py:690-691`). It does not enter the file names, the report or ODMAP.
- **Caution.** If the GCM information is to appear in the report and files, it must be written into the scenario name.

8.5 Algorithms to project (Algorithms to Project)

Name in the UI	What it does	Default	Range / options	Code
algorithm boxes (e.g. `GAM`, `GLM`, `MAXENT`, `RF`)	Which algorithms' models are projected	all checked	algorithms trained in ③	`tabs/tab_projection.py:288-306`
`All` / `None`	Checks all / unchecks all	—	—	`tabs/tab_projection.py:142-145`

- **What it does.** Each time the tab is opened, the boxes are built as a grid of four per row from the models trained in ③ (`modeler.fitted_models`). The **full-data** models of the selected algorithms ("Full", the model refitted with all data; one for each pseudo-absence set) are projected (`sdm_core.py:7850-7855`). If `Also project the cross-validation models (more replicates, slower)` was checked in ③, the cross-validation models are projected as well (`tabs/tab_models.py:1537`, `:1546`).
- **When and how to set it.** If the current map and the future map are to be compared (⑥ Range Change), the two maps must be produced with **the same set of algorithms**; therefore leaving all algorithms used in ③ checked is the default and recommended choice. If an algorithm is known to behave poorly in transfer (e.g. a GAM that jumps outside the range), leaving it out can be done as a sensitivity test.
- **Effect on results.** An unselected algorithm does not enter the ensemble map; the ensemble weights are computed only over the projected models (`sdm_core.py:8028-8035`). In the example run, with four algorithms selected, 8 models (4 algorithms × 2 pseudo-absence sets) were projected and 37 rasters were written.
- **Caution.** If no box is checked, the message `Select at least one algorithm to project.` appears. If none of the selected algorithms passes the ensemble criterion in ③ (`minimum score`), no ensemble map is written and the warning `WARNING: No full-data model reached the minimum …` appears in the log (`sdm_core.py:8036-8054`).

8.6 Raster list and the order rule (Environmental Rasters for This Scenario)

Name in the UI	What it does	Default	Range / options	Code
raster list	Continuous layers of the scenario, in variable order	empty	multiple selection	`tabs/tab_projection.py:159-162`
`Browse Folder…`	Matches the rasters in a folder to the variable names and fills the list	—	`.tif .tiff .asc .img .nc .vrt`	`tabs/tab_projection.py:166-170`, `:382-449`
`Add Files…`	Appends the selected files to the end of the list	—	same extensions	`tabs/tab_projection.py:172`, `:451-457`
`Remove`	Deletes the selected rows	—	—	`tabs/tab_projection.py:174`, `:459-461`
`Remove All`	Deletes the list and **all projection results**, locks ⑥	—	asks for confirmation	`tabs/tab_projection.py:177-180`, `:463-489`

8.6.1 The order rule

- **What it does.** The program matches continuous rasters **by position**: the i-th file in the list is read as the i-th continuous variable of the model (`sdm_core.py:7335`, `:7407-7412`). There is a single exception: if the names of the files in the list (in cleaned form, e.g. `bio_ (12)` → `bio_12`) form exactly the same set as the model variables, the files are matched by name even if the order differs (`sdm_core.py:7375-7379`). This prevents alphabetically sorted lists such as `bio_11, bio_15, bio_18, bio_19, bio_2, …` from swapping two layers.
- **When and how to set it.** Storing the future layers with **the same file names** as the training layers (`bio_2.tif`, `bio_3.tif` …) is the safest course; then name matching takes effect and an order error becomes impossible. If the file names differ (such as `wc2.1_2.5m_bioc_MPI_ssp585_2081-2100_02.tif`), the list must be built exactly in the order of the orange line.
- **Effect on results.** An order error does not stop the program: if two variables are swapped, the model takes, for example, precipitation seasonality for temperature and the map becomes meaningless. The only trace revealing this is an unusually high ExDet NT1 proportion (§8.9).
- **Caution.** The number of files must equal the number of continuous variables of the model; otherwise the message `Raster count mismatch!` appears and the projection does not start (`tabs/tab_projection.py:525-534`, `sdm_core.py:7360-7366`). Categorical variables (land cover, soil class) are **not added** to this list: the program takes them from the current categorical rasters loaded in ① Data and assumes them to be the same in the future (`sdm_core.py:7474-7483`, `:7414-7442`). That is, land cover is assumed not to change.

8.6.2 Adding from a folder (Browse Folder…)

- **What it does.** The names of the raster files in the selected folder are converted to lower case and spaces, parentheses, hyphens and dots are reduced to underscores; for each model variable, the program searches in turn for (1) an exact name match, (2) a file name that starts with the variable name, (3) the variable name occurring within the file name (`tabs/tab_projection.py:399-436`). The matches are written to the list **in variable order**.
- **When and how to set it.** With a layout in which each scenario is kept in a separate folder and with the same names as in training, the correct list is built with a single click (in the example run `MPI-ESM1-2-HR_SSP585_2081_2100\bio_2.tif …`).
- **Effect on results.** With a correct match, it gives the same result as a manually built list.
- **Caution.** The "prefix" and "contains" rules are loose: for the variable `bio_1` the exact name is searched first; if it is not found, the first file starting with `bio_1`, such as `bio_10.tif` or `bio_11.tif`, may be taken. Unmatched variables are listed in the `Unmatched Variables` dialog and the user is asked to add them with `Add Files…`. After the list has been built, check it visually against the orange line.

8.6.3 Adding and removing files

`Add Files…` appends the selected files to the **end** of the list; keeping the order is up to the user. `Remove` deletes only rows in the list. `Remove All`, however, after asking for confirmation, clears the list, resets all projection and range change results in the session, empties the maps and tables of both tabs and locks ⑥ Range Change again (`tabs/tab_projection.py:475-489`). Files on disk are not deleted.

8.7 Output and display settings (Output Settings)

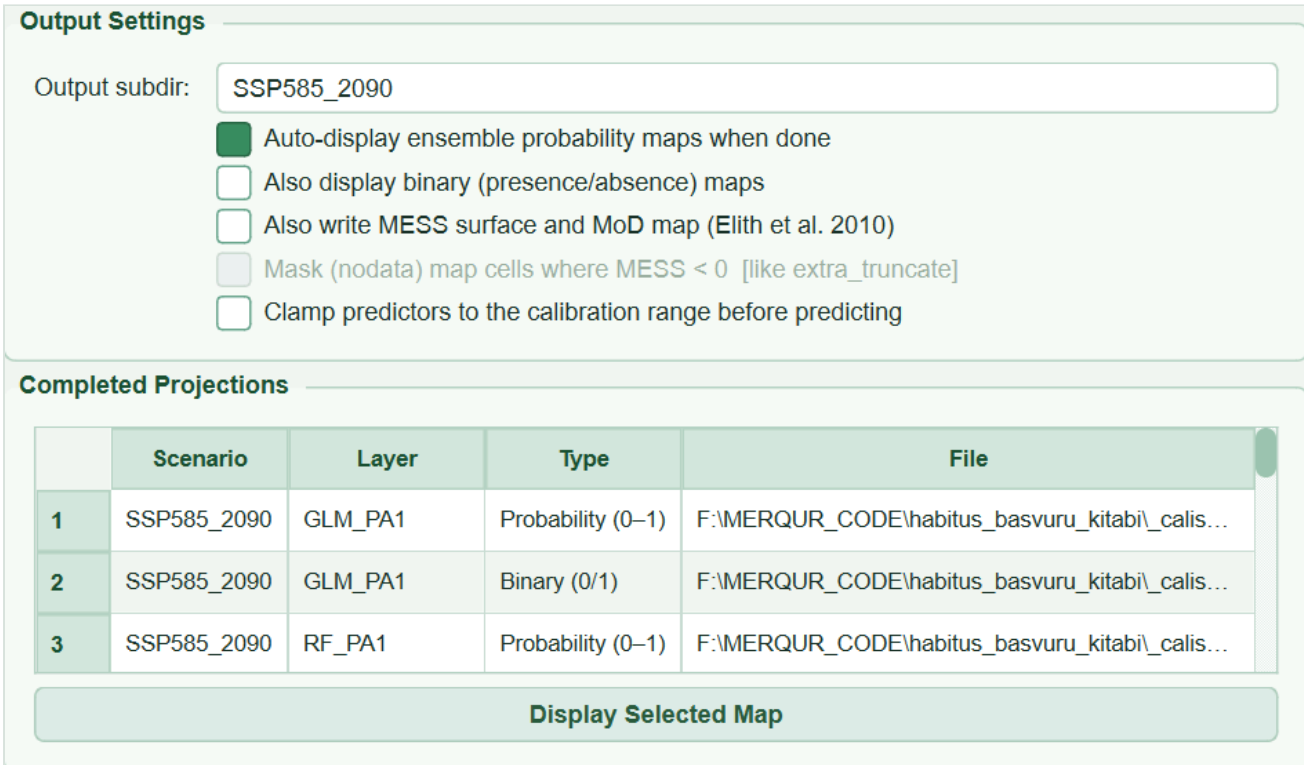


Figure 8.2. Lower part of the left pane after projection: the options used in the example run in the Output Settings box (only Auto-display on; the MESS mask greyed out because MESS is off) and the first rows of the Completed Projections table.

Name in the UI	What it does	Default	Range / options	Code
`Output subdir:`	Subfolder to which the scenario files are written	empty (scenario name is used)	free text	`tabs/tab_projection.py:197-198`, `:543-545`
`Auto-display ensemble probability maps when done`	Loads the probability maps into the viewer when the job finishes	on	on / off	`tabs/tab_projection.py:199-201`, `:669-670`
`Also display binary (presence/absence) maps`	Also loads the binary maps	off	on / off	`tabs/tab_projection.py:202-203`, `:683-684`
`Also write MESS surface and MoD map (Elith et al. 2010)`	Also writes the MESS and MoD rasters	off	on / off	`tabs/tab_projection.py:204-214`
`Mask (nodata) map cells where MESS < 0 [like extra_truncate]`	Sets cells with MESS < 0 to nodata in the maps	off (disabled while MESS is off)	on / off	`tabs/tab_projection.py:215-224`
`Clamp predictors to the calibration range before predicting`	Clips the predictors to the training range before prediction	off	on / off	`tabs/tab_projection.py:225-233`

8.7.1 Output subfolder

- **What it does.** Files are written under `<output folder>\<Output subdir>`; if the field is empty, the scenario name is used (`tabs/tab_projection.py:544-545`).
- **When and how to set it.** In most cases it is left empty. A different subfolder can be given to compare two runs of the same scenario with different settings (e.g. clamping on/off); however, because the file names are generated from the scenario name, ⑥ and the report distinguish the two runs only if the scenario names also differ.
- **Caution.** The report (⑧) looks for the PNGs of the future maps in these subfolders (§11.5.7); saving the scenario PNGs here determines the layout of the report.

8.7.2 Display options

While `Auto-display …` is on, the viewer is cleared when the job finishes and **all** probability maps whose names end in `_prob` are loaded: not only the ensemble but also the single models and the algorithm means (14 maps in the example run, `tabs/tab_projection.py:675-705`). If `Also display binary …` is on, the `_bin` maps are added as well. These two options do not change any file; they only fill the viewer's `Layer:` list.

8.8 Transfer safety options: MESS, MESS mask, clamping

A model produces a value between 0 and 1 even under environmental conditions it never saw in training; this value appears as "certain" as the others. The narrower the environmental range of the calibration area, the more often this happens in transfer (Owens et al. 2013). HABITUS handles this problem at three levels: ExDet is computed automatically in every projection (§8.9); MESS is an optional additional diagnostic; the MESS mask and clamping change the map itself.

All three use the same **calibration reference set**: the rows on which the models were actually built, i.e. the union of the presence and absence points in all pseudo-absence sets (`sdm_core.py:7142-7171, :4929-4952`).

Categorical variables do not enter the reference.

8.8.1 MESS and MoD (Also write MESS surface and MoD map)

- **What it does.** Following the definition of Elith et al. (2010), a similarity value is computed for each cell and each variable. With F the percentage of reference points whose value of variable j is less than or equal to p, $\min_j$ the smallest value and $\max_j$ the largest (`sdm_novelty.py:377-449`):

$$s_j = \begin{cases} 100\,(p - \min_j)/(\max_j - \min_j) & F = 0 \\ 2F & 0 < F \le 50 \\ 2(100 - F) & 50 < F < 100 \\ 100\,(\max_j - p)/(\max_j - \min_j) & F = 100 \end{cases}$$

MESS is the smallest of the s_j; MoD (most dissimilar variable) is the index of the variable giving this smallest value. The files written are `<species>_<scenario>_MESS.tif` (float32) and `<species>_<scenario>_MoD.tif` (int16; the variable names are stored in the raster labels as `CLASS_i`) (`sdm_core.py:7263-7281`). - **When and how to set it.** MESS is turned on in studies that require comparison with the outputs of the MaxEnt software and dismo/flexsdm, or where a reviewer asks for a "MESS map". In terms of method, ExDet NT1 already gives the same information (§8.9). - **Effect on results.** It does not change the suitability maps. The line `MESS: x% of projected cells have MESS < 0 …` is written to the log (`sdm_core.py:7286-7289`). Because the same reference set and the same minimum/maximum values are used, **the cells with MESS < 0 are the same as the cells with ExDet NT1 < 0** (if a value is exactly equal to the maximum, both give 0): in the example run, the NT1 < 0 proportion for SSP585_2090 is 55.4%; had MESS been turned on, the same proportion would have been reported for MESS < 0. The only difference is a variable that is constant in training: NT1 ignores it, whereas MESS gives −100 to every cell that differs from the constant value (`sdm_novelty.py:216-225`, `:436-437`). - **Caution.** MESS is univariate: it cannot see cells where the combination is novel while all variables are individually within range; for these, ExDet NT2 is consulted. The tooltip text says this too: `MESS cannot see novel combinations of in-range values; ExDet NT2 can.`

8.8.2 MESS < 0 mask (Mask (nodata) map cells where MESS < 0)

- **What it does.** After all outputs have been written, cells with MESS < 0 are set to nodata (−9999) in the probability, binary, algorithm mean, algorithm spread (`_sd`) and ensemble spread (`EMsd`, `EMcv`, `EMrange`) maps (`sdm_core.py:7293-7325`, `:8112-8123`). The ExDet and MESS/MoD rasters are not masked. It is the counterpart of the `extra_truncate` function in flexsdm (Velazco et al. 2022). The box is not enabled unless the MESS box is checked (`tabs/tab_projection.py:224`).
- **When and how to set it.** Turned on when the map will be used directly for a decision (conservation priority, area computation) and extrapolated cells should not be interpreted at all. In a scientific report, the unmasked map and the NT1 map are usually given side by side and the masked computation is added as a sensitivity analysis.
- **Effect on results.** The effect is large. Because MESS < 0 and NT1 < 0 are the same cells, in the example run the mask would have set 55.4% of the cells of the SSP585_2090 map to nodata. ⑥ Range Change counts only cells that are valid in both maps (`sdm_core.py:8292`); a masked future map does not count the masked cells as "lost" or "gained", it removes them from the computation entirely. Thus the current area, too, is computed only from current presence cells lying outside the mask.
- **Caution.** If MESS cannot be computed, no masking is done and `WARNING: MESS truncation was requested but MESS could not be computed; the maps of <scenario> were NOT truncated.` is written to the log (`sdm_core.py:8114-8119`). Whether masking was done can be read from the `mess_truncate` and `mess_truncated_files` fields of the `_projection_options.json` file.

8.8.3 Clamping (Clamp predictors to the calibration range before predicting)

- **What it does.** Before prediction, each continuous variable in the projection is clipped to the minimum and maximum values of the reference set (`np.clip` logic; `sdm_core.py:7808-7814`, `:4929-4960`). Single models, algorithm means and the ensemble predict with these clipped values. ExDet and MESS, however, are computed with the unclipped raw values; that is, the novelty maps continue to show the actual extrapolation.
- **When and how to set it.** Algorithms that fit flexible response curves (quadratic terms of GLM, GAM curves, MaxEnt hinge/product features) can rise and fall outside the range and produce unrealistic values; tree-based methods (RF, GBM, BRT), on the other hand, already stay constant at the last value outside the range. If the NT1 proportion in the future layers is high (55.4% in the example run), making two projections, with and without clamping, and reporting the difference is good practice. This is the logic of the "clamping" option in the MaxEnt software and of the `do.clamping` / `clamping.mask` approach in biomod2.
- **Effect on results.** In out-of-range cells, suitability is fixed at the suitability of the value at the edge of the range. In tree-based methods the effect is small; in GLM/GAM/MaxEnt it can be marked in extrapolated cells. The gain/loss proportions in ⑥ are affected by these cells.
- **Caution.** Clamping does not remove extrapolation, it only holds the response constant; the assumption is made that "the condition at the edge of the range continues outside the range". The clamping setting in ③ Models (`settings.models.clamp`) governs the current map; the box in ⑤ is chosen separately for each scenario and written to the `_projection_options.json` file. For comparability, use the same choice for the current and future maps.

8.9 ExDet novelty surfaces (always written)

There is no box for this in the UI: `Projector` runs ExDet in every projection (`write_novelty=True`, `sdm_core.py:7088`, `:7937-7943`). The method is the ExDet of Mesgaran et al. (2014) and it distinguishes two kinds of novelty.

NT1 (univariate range novelty). With $\min_j$ the smallest and $\max_j$ the largest value of variable j in the reference set, for cell i (`sdm_novelty.py:202-228`):

$$UD_{ij} = \frac{\min\left(P_{ij} - \min_j,\ \max_j - P_{ij},\ 0\right)}{\max_j - \min_j}, \qquad NT1_i = \sum_j UD_{ij}$$

Within the range NT1 = 0, outside it is negative; the larger the absolute value, the further outside the range the cell lies.

NT2 (combinational novelty). Only in cells with NT1 = 0, the squared Mahalanobis distance D^2 is computed with the mean and covariance of the reference set and divided by the largest D^2 in the reference set: $NT2_i = D^2_i / \max(D^2_{ref})$ (`sdm_novelty.py:249-290`). NT2 > 1 marks a cell where each variable on its own is within range but the combination was never seen in training.

MIC (most influential covariate). For NT1, the variable giving the most negative UD_{ij}; for NT2, the variable whose removal reduces D^2 the most (Mesgaran et al. 2014, eq. 7; `sdm_novelty.py:231-235`, `:292-317`).

Files written: `<species>_<scenario>_NT1.tif`, `_NT2.tif`, `_MIC_NT1.tif`, `_MIC_NT2.tif` (MIC values are variable indices; the names are in the raster labels as `CLASS_i`; `sdm_core.py:7206-7233`). A plain-text summary is written to the log (`sdm_novelty.py:640-692`). The SSP585_2090 summary of the example run:

Measure	current	SSP585_2090
NT1 < 0 (out of range)	23.4%	55.4%
NT2 > 1 (novel combination)	0.0%	0.3%
Similar according to both tests	76.6%	44.3%
Most influential variable, range novelty	bio_3 (43%), bio_15 (24%), bio_2 (13%)	bio_9 (58%), bio_3 (17%), bio_8 (10%)

When the NT1 proportion exceeds 25%, the warning `More than a quarter of the projection is outside the calibration range. Treat the suitability surface in those areas as an extrapolation, not a prediction.` is added to the log (`sdm_novelty.py:686-690`).

> **Caution:** The reference set consists of the presence and pseudo-absence points; this is why even in the current map 23.4% of the cells come out with NT1 < 0: because the training points are selected from within the accessible area (in the example, 200 km around the records), the distant corners of the study area are outside the range even today. This is not an error but a finding that says that this part of the map is also an extrapolation.
>
> **Note:** If the condition number of the calibration covariance matrix exceeds 10^{12} (variables that are near copies of each other), NT2 is not computed; the reason is written to the log and the report: `NT2 was not computed: the calibration covariance matrix is numerically singular …` (`sdm_novelty.py:62, :257-265`). NT1 is written in this case too.

8.10 Grid alignment, clipping to the study boundary and strip-wise projection

These three operations require no settings from the user, but they directly affect the result and the meaning of the warnings the program gives needs to be understood.

8.10.1 Grid alignment

- **What it does.** Each scenario layer is fitted **by position** onto the training grid (transform, number of rows and columns, coordinate system) (`sdm_grid.py:188-280`, `sdm_core.py:7486-7498`):
- if the grid is the same, the file is read as it is;
- if the cell size is the same and the offset is a whole number of cells, the values are shifted without interpolation (`cell-shift`);
- in every other case the layer is resampled with `rasterio.warp.reproject`: bilinear for continuous layers, nearest neighbour for categorical layers.

Training cells not covered by the layer become NaN and are counted; a cell with NaN is excluded from the projection (cells in which all variables are finite are predicted, `sdm_core.py:7799-7807`). - **When and how to set it.** Downloaded GCM sets may be one cell wider than the training layers or cut differently; alignment brings them together safely. It is best to prepare the future layers at the same resolution and extent as the training layers; then no resampling is done. - **Effect on results.** Shifting and identity do not change the values; resampling (especially from a different resolution) smooths the values. - **Caution.** For every aligned layer, a line is written to the log and to the `Processing Notices` section of the report, for example `WARNING grid alignment: bio_2: 468x222 px grid realigned to 466x220 reference by position (cell-shift), offset -1 col / -1 row, 0 reference cell(s) not covered (0.0%)`. If a layer does not cover more than 5% of the training grid, the additional warning `… % of the reference grid is outside this layer's extent (> 5%); those cells are NaN and are left out of the projection. Check that the layer covers the study area.` appears (`sdm_grid.py:40, :304-328`). The summary is stored in the `grid_alignment` field of the `_projection_options.json` file; in the example run `n_realigned: 0` (all eight layers on the training grid).

8.10.2 Clipping to the study boundary

- **What it does.** If a study boundary was given in ① Data, the future rasters are also clipped to the same boundary and placed on **the grid of the clipped training rasters** (`tabs/tab_projection.py:589-613`, `sdm_mask.py:304-418`). The clipped copies are written under `<output folder>\masked_inputs\<scenario>\` with their original file names, so that name matching is not broken. If all files in the list are already the (clipped) files used in training, clipping is skipped (`tabs/tab_projection.py:590-592`).
- **Effect on results.** The current and future maps cover the same area cell by cell; ⑥ compares like with like.
- **Caution.** During clipping, layers with a different cell size are moved onto the training grid by **nearest neighbour** and `Resampled to the training grid (nearest neighbour, another cell size): …` is written to the log (`tabs/tab_projection.py:610-613`, `sdm_mask.py:273-301`). Without a boundary, in the same situation, continuous layers are resampled bilinearly (§8.10.1). If future layers of a different resolution are to be used, bringing them to the training resolution beforehand removes this difference. If the boundary does not overlap the layer, or a layer has no data at all within the boundary, the job stops with an explicit message (`sdm_mask.py:369-403`).

8.10.3 Strip-wise projection (large rasters)

- **What it does.** If the training grid exceeds 4 million cells (default strip size 2 million cells; the grid is split if it is larger than twice this), the raster is projected in row strips and the strips are merged into the same files (`sdm_core.py:7514-7527`, `:7566-7682`). Because each cell is predicted on its own, the result is the same as a single-piece computation; the ExDet and MESS summaries are obtained by summing the counts from the strips (`sdm_novelty.py:452-637`).
- **When and how to set it.** There is no user setting. The strip size can be changed with the `HABITUS_PROJECTION_TILE_CELLS` environment variable (0 = no splitting). A user who gets an out-of-memory error can reduce this value.
- **Effect on results.** The values are the same; the run time increases slightly and memory use decreases. `Large raster (… cells): projecting in k strips of up to n rows to keep memory in check.` is written to the log.
- **Caution.** Intermediate strip files are written to a temporary folder named `.habitus_strips_*` in the output folder and deleted when the job finishes; the output disk needs as much free space as the final maps. Layers that are not on the training grid are first aligned once and written to this folder.

8.11 Outputs and the Completed Projections table

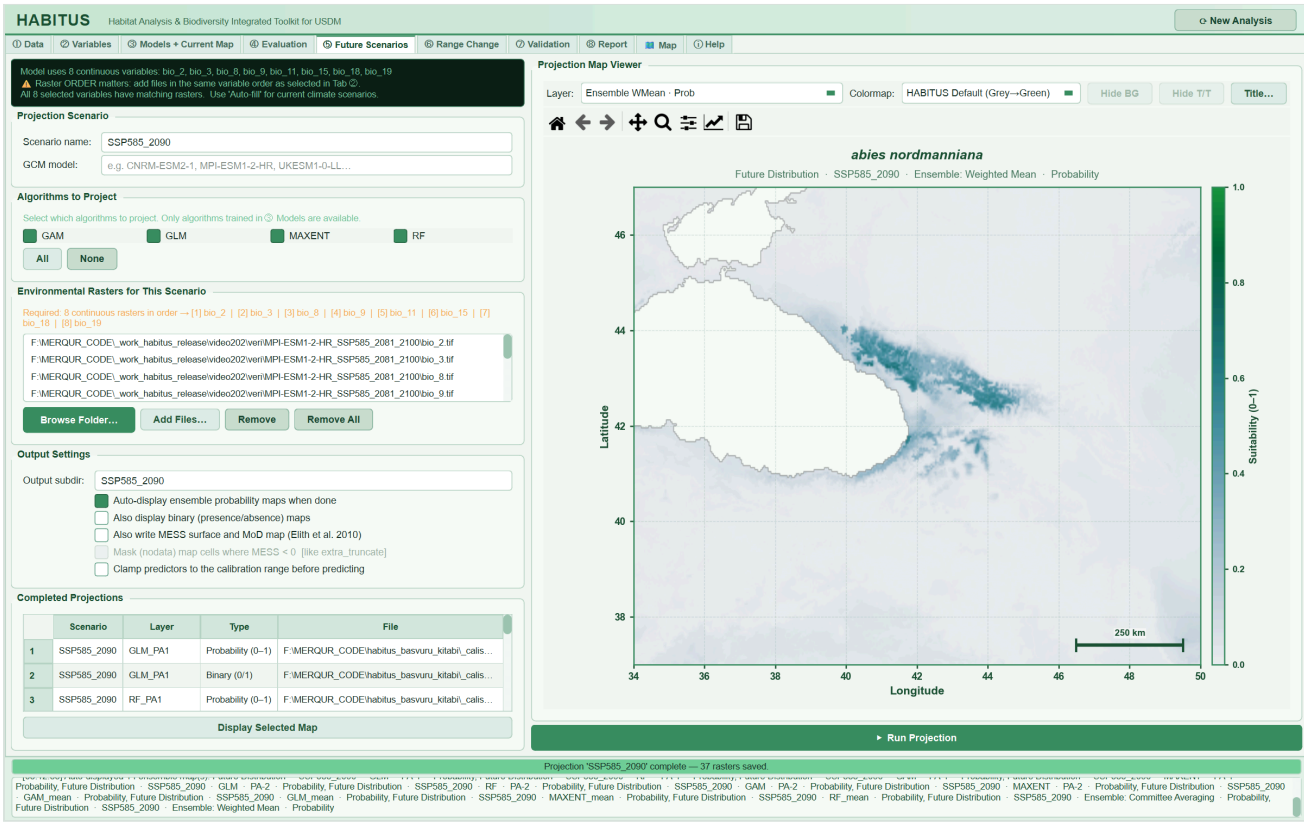


Figure 8.3. After the SSP585_2090 projection has completed: rows of scenario, layer, type and file path in the Completed Projections table; on the right the weighted-mean ensemble probability map (Ensemble WMean · Prob); in the bottom bar the message that 37 rasters were saved.

In the example run, 37 rasters and one JSON record were written to the `SSP585_2090\` folder:

File (`<species>_<scenario>_…`)	Content	Code
`<ALG>_PA<k>_prob.tif`, `_bin.tif`	Probability and binary map of a single model (threshold: the threshold found for that model in ③)	`sdm_core.py:7895-7916`
`<ALG>_mean_prob.tif`, `_mean_bin.tif`, `_sd.tif`	Mean, binary and standard deviation over the replicates of the algorithm (if the two-level ensemble is on in ③)	`sdm_core.py:7964-8023`
`EMwmean_prob.tif`, `EMca_prob.tif` and `_bin`	Weighted-mean and committee-averaging ensemble maps	`sdm_core.py:8055-8103`
`EMsd.tif` (and, if selected, `EMcv`, `EMrange`)	Spread among the ensemble members; not thresholded	`sdm_core.py:8057-8068`
`NT1`, `NT2`, `MIC_NT1`, `MIC_NT2`	ExDet surfaces	`sdm_core.py:7227-7233`
`MESS`, `MoD`	Only if MESS was selected	`sdm_core.py:7279-7281`
`_projection_options.json`	clamp, mess, mess_truncate, number of masked files, MESS < 0 proportion, grid alignment summary	`sdm_core.py:8126-8143`

All probability maps are float32, nodata −9999 and in the 0–1 range; the map is not stretched to its own range, so that thresholds and scenarios remain comparable (`sdm_core.py:7884-7889`). If the source raster has geographic coordinates (or no coordinate system), EPSG:4326 is assigned to the output; for a projected source the original system is kept (`sdm_core.py:7830-7837`). The ensemble binary threshold is 0.5 (majority vote) for committee averaging, and for the weighted mean it is the mean of the members' thresholds with the same weights (`sdm_core.py:6423-6446`).

Name in the UI	What it does	Default	Range / options	Code
`Completed Projections` table	One row for each output file: `Scenario`, `Layer`, `Type`, `File`	empty	row selection	`tabs/tab_projection.py:243-257`, `:644-662`
`Display Selected Map`	Loads the map of the selected rows into the viewer	—	—	`tabs/tab_projection.py:259-262`, `:709-733`

The `Type` column shows `Probability (0–1)` for probability maps and `Binary (0/1)` for binary maps; for novelty and spread files the last part of the key (e.g. `NT1`, `sd`) appears.

> **Caution:** `Display Selected Map` draws every row that does not say `Probability` with the binary colour scale. To inspect NT1, NT2, MESS or `_sd` maps, opening these files in GIS software or in the 🗺 Map tab gives a more accurate display.

8.12 Map viewer (Projection Map Viewer)

The viewer in the right pane (`RasterMapWidget`, `tabs/tab_projection.py:265-272`) is the same as the current map viewer in ③: the `Layer:` list shows the loaded layers by their short names (`Ensemble WMean · Prob`, `GLM · PA1 · Bin`), `Colormap:` selects the colour scale of the probability map, `Hide BG` hides the background country borders, `Title…` edits the title; the disk icon in the toolbar saves the map at journal column width. The title is built from the scenario, GCM, ensemble method and map type (`tabs/tab_projection.py:31-44`). All controls of the viewer and the publication output are described in Chapter 12.

> **Tip:** Save the PNG of the future map into the scenario subfolder (e.g. `SSP585_2090\`) so that it enters the report. The save dialog opens by default in the root of the output folder; a PNG saved to the root enters the "6. Current Distribution" section of the report (§11.5.6).

8.13 Outputs and what passes to the next step

- `state["projections"][<scenario>]`: a key → file path dictionary. The `Current proj. key` / `Future proj. key` lists in ⑥ Range Change are built from it (only layers with `_bin` in their name).
- `state["novelty"][<scenario>]`: ExDet summary; the novelty table in the report (§11.5.8) and the extrapolation share in ⑥ use it and the `NT1.tif` file.
- `state["grid_alignment"]`, `state["projection_settings"]`: processing notes in the report and ODMAP.
- `run_log`: the raster list, algorithms and options of the scenario; the multi-species batch run in ⑧ repeats this record (`tabs/tab_projection.py:580-584`).

8.14 Expert questions

Q1. Are future rasters matched by name or by order? How is an order error prevented? Answer: The basic rule is position: the i-th file is the i-th continuous variable (`sdm_core.py:7335`, `:7407`). However, if the (cleaned) file names form the same set as the model variables, the program corrects the order according to the names (`sdm_core.py:7375-7379`). If the file names differ, no correction is possible. The most robust course is to store the future layers with the same names as in training and to use `Browse Folder…`. In biomod2, `BIOMOD_Projection` matches the new environment by layer names; in HABITUS, name matching takes effect only when the names match exactly.

Q2. What is the reference set of ExDet? Why are MaxEnt's background points not included? Answer: The reference is the union of the presence + absence rows in all pseudo-absence sets (`sdm_core.py:7142-7171`). Mesgaran et al. (2014) recommend choosing the reference according to the modelling approach; in a presence-background model, this is the environment the model has seen. MaxEnt, however, is additionally built with 10,000 background points (`maxent_background`, `sdm_core.py:3916`, `:4596-4598`), and these points are not added to the reference. Points outside the accessible area are absent from both sets; however, because MaxEnt's background may cover a wider environmental range, ExDet may be somewhat pessimistic (showing more novelty) from MaxEnt's point of view. This effect has not been measured.

Q3. Do MESS and ExDet NT1 give different results? Answer: With the same reference, the set of out-of-range cells is the same: MESS < 0 occurs only if a value is smaller than the minimum or larger than the maximum (`sdm_novelty.py:431-435`); NT1 < 0 is exactly this as well (`sdm_novelty.py:221-225`). The values differ (MESS is the smallest similarity on a percentage scale, NT1 the sum of range-normalised distances), the sign is the same. MESS additionally gives cells within the range a similarity between 0 and 100; NT1 is 0 there. The difference appears for a variable that is constant in training (§8.8.1).

Q4. Why is the NT2 threshold 1? Mahalanobis distance or its square? Answer: The code computes D^2 (the square) and divides it by the largest D^2 in the reference set (`sdm_novelty.py:271-290`). A cell whose ratio exceeds 1 is "further away" than even the most extreme point in the reference set; this is the criterion of Mesgaran et al. Taking the square or the root does not change the threshold of 1 (monotonic transformation). The inverse of the covariance matrix is obtained with `pinv`; if the condition number exceeds 10^{12}, NT2 is not computed (`sdm_novelty.py:257-269`).

Q5. Does clamping change the ExDet and MESS results? Answer: No. Clipping is applied only to the prediction matrix (`X_valid`); ExDet and MESS are computed from the raw `X_flat` (`sdm_core.py:7808-7814`,

`:7937-7955`). Thus even with clamping on, the novelty maps show the actual extrapolation. The clipping range for clamping is also the minimum/maximum value of the reference set (`sdm_core.py:4929-4952`).

Q6. Which models are projected; where are the cross-validation models? Answer: By default, for each algorithm and each pseudo-absence set, the "Full" model refitted with all data is projected; this corresponds to the `allData` behaviour of biomod2 (Thuiller et al. 2009) (`sdm_core.py:7846-7855`). If `Also project the cross-validation models` is selected in ③, the fold models are projected as well, and the algorithm mean and spread rest on more replicates; the file names carry the run label, such as `PA1_RUN1` (`sdm_core.py:7874`). This choice is made in ③, not in ⑤, and is kept there for consistency with the current map (`tabs/tab_models.py:1538-1546`).

Q7. Where does the threshold of the future binary map come from? Is it recomputed from the future data? Answer: No. For each model, the threshold found from cross-validation in ③ (`tss_thresholds`, default method max TSS) is used unchanged (`sdm_core.py:7908`); the ensemble threshold is the weighted mean of the members' thresholds for the weighted mean, and 0.5 for committee averaging (`sdm_core.py:6423-6446`). Since there are no observations in the future, re-estimating the threshold would be impossible anyway. Because the probability maps are not stretched, this threshold is meaningful on the same scale in both periods (`sdm_core.py:7884-7889`).

Q8. How are categorical variables (land cover) handled in the future? Answer: Categorical rasters are not added to the future list; the current categorical rasters loaded in ① Data and used in the model are read again (`sdm_core.py:7474-7483`). That is, categorical variables are assumed not to change. If a future land cover scenario is to be used, the program does not support this; the variable must be converted to a continuous layer (e.g. forest proportion) and modelled as continuous.

Q9. What happens if the future layers have a different resolution? Answer: Without a study boundary, continuous layers are resampled to the training grid bilinearly and categorical layers by nearest neighbour (`sdm_grid.py:263-280`); with a boundary, nearest neighbour is used during clipping (`sdm_mask.py:273-301`). In both cases a warning is written to the log and the report. Resampling a coarse GCM layer onto a fine grid produces no information; it only equalises the number of cells. It is recommended that training and projection be prepared at the same resolution (after downscaling if necessary).

Q10. Why does the range change differ when the MESS < 0 mask is on? Answer: The masked cells become nodata in the future map; ⑥ counts only cells that are valid in both maps (`sdm_core.py:8292`). These cells become neither loss nor gain but "excluded from the computation"; both the current and the future area shrink. In the example run this means removing 55.4% of the cells. Reporting the two results, with the mask on and off, together shows how much the result depends on extrapolation. The `Accessible area without extrapolation` option in ⑥ answers the same question without changing the map (§9.6).

Q11. How is an average over several GCMs made instead of a single GCM? Answer: The program does not produce an ensemble across GCMs or a scenario uncertainty surface; the ODMAP record says so too (`odmap.py`, "Uncertainty through scenarios"). Each GCM is projected under a separate scenario name; mean and spread maps are computed outside the program (e.g. R `terra::app`). The difference between GCMs is often larger than the difference between algorithms; in a study done with a single GCM this limitation should be stated.

Q12. Is the projection reproducible? Answer: There is no random component in the projection: the same models and the same rasters give the same map. The models themselves are determined by the seed in ① Data. The options used are written to the `_projection_options.json` file (`sdm_core.py:8126-8143`), and the raster list and algorithms to the `run_log` in the session; the raster paths do not enter the report or the run record, so the source of the GCM files used must be stated separately in the methods text.

8.15 Common mistakes and pitfalls

- **Not changing the default scenario name.** If SSP5-8.5 layers are projected under the name `2050_SSP245`, all files and the report carry the wrong name.
- **Mixing up the order.** When name matching does not take effect (file names differ), an order error passes without a message. Its symptoms are an unusually high NT1 proportion and an unexpected variable in the `Most influential predictor for range novelty` line.
- **Adding a categorical variable to the list.** The message `Raster count mismatch!` appears; categorical layers are reused automatically.
- **Running a second GCM under the same name.** The files are overwritten in the same folder. Use a separate name for each GCM.
- **Assuming that `Remove All` only clears the list.** This button resets all projection and range change results in the session and locks ⑥.
- **Using the MESS mask without saying so.** An area computation made with a masked map leaves out the extrapolated cells; if this is not stated in the methods, the reader misinterprets the loss proportion.
- **Different clamping in the current and future maps.** The clamping choice in ③ and in ⑤ is made separately; if they differ, part of the range change comes from the difference in method.
- **Ignoring NT1.** In the example run, more than half of the future cells are out of range; the suitability in these cells is not a prediction but an extrapolation.

8.16 References

Elith, J., Kearney, M., Phillips, S., 2010. The art of modelling range-shifting species. Methods in Ecology and Evolution 1, 330–342. https://doi.org/10.1111/j.2041-210X.2010.00036.x

Mesgaran, M.B., Cousens, R.D., Webber, B.L., 2014. Here be dragons: a tool for quantifying novelty due to covariate range and correlation change when projecting species distribution models. Diversity and Distributions 20, 1147–1159. https://doi.org/10.1111/ddi.12209

Owens, H.L., Campbell, L.P., Dornak, L.L., Saupe, E.E., Barve, N., Soberón, J., Ingenloff, K., Lira-Noriega, A., Hensz, C.M., Myers, C.E., Peterson, A.T., 2013. Constraints on interpretation of ecological niche models by limited environmental ranges on calibration areas. Ecological Modelling 263, 10–18. https://doi.org/10.1016/j.ecolmodel.2013.04.011

Thuiller, W., Lafourcade, B., Engler, R., Araújo, M.B., 2009. BIOMOD – a platform for ensemble forecasting of species distributions. Ecography 32, 369–373. https://doi.org/10.1111/j.1600-0587.2008.05742.x

Velazco, S.J.E., Rose, M.B., de Andrade, A.F.A., Minoli, I., Franklin, J., 2022. flexsdm: An R package for supporting a comprehensive and flexible species distribution modelling workflow. Methods in Ecology and Evolution 13, 1661–1669. https://doi.org/10.1111/2041-210X.13874

9. Range change (⑥ Range Change)

Tab: ⑥ Range Change · Code: `tabs/tab_range.py` (UI), `sdm_core.py` `RangeChangeAnalyzer` (computation), `sdm_grid.py` (grid alignment)

9.1 What this tab does

⑥ Range Change compares two binary (0/1, absent/present) maps cell by cell and counts how the distribution range of the species has changed. Usually the current ensemble map is compared with the ensemble map of a future scenario; but any pair can be used as long as the two binary maps cover the same area (two scenarios, two algorithms, two periods). The computation is the counterpart of the `BIOMOD_RangeSize` function in biomod2 (Thuiller et al. 2009) (`tabs/tab_range.py:1-5`, `sdm_core.py:8159-8163`).

The tab becomes available once ③ Models is complete (`tabs/tab_models.py:1850`); for the lists to be filled with future maps, at least one projection must have been made in ⑤ Future Scenarios.

- **Input:** the current and future binary maps; optionally the accessible-area raster written by ① Data (`accessible_area.tif`) and the ExDet NT1 surface written by ⑤.
- **Computation:** each cell is given the code −2 (lost), 0 (stable absent), 1 (stable present), 2 (gained); counts, percentages and net change are computed (`sdm_core.py:8196-8347`).
- **Output:** the range change map `<species>_<analysis name>_SRC.tif`, the `Range Change Statistics` table, a coloured map in the viewer and the "8. Range Change Analysis" section in the ⑧ report.

9.2 Screen layout

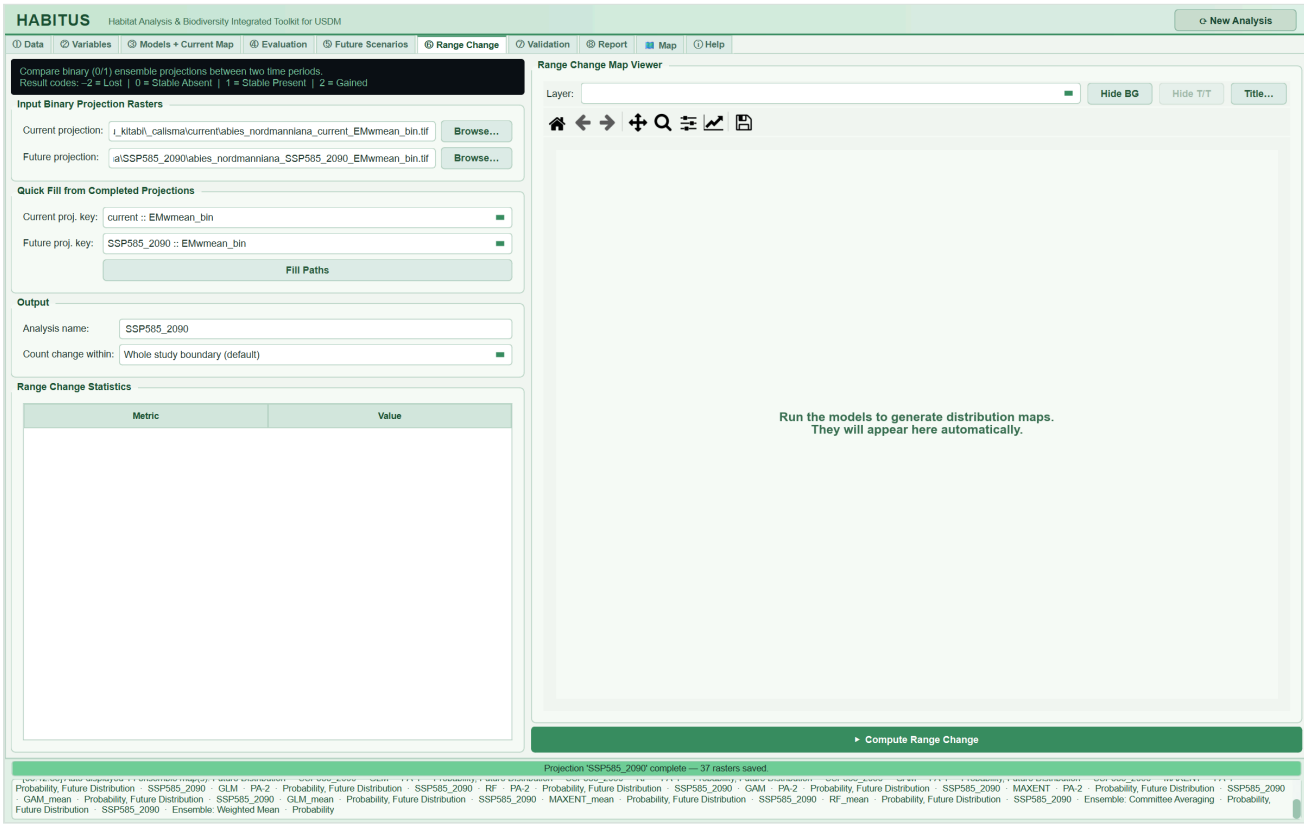


Figure 9.1. Range Change tab before running: at the top the strip explaining the result codes; on the left Input Binary Projection Rasters (two path fields), Quick Fill from Completed Projections (two lists and Fill Paths), Output (Analysis name, Count change within) and the empty Range Change Statistics table; on the right the empty Range Change Map Viewer.

Area	Name in the UI	Described in
Information strip	(untitled)	§9.1, §9.7
File selection	`Input Binary Projection Rasters`	§9.3
Quick fill	`Quick Fill from Completed Projections`	§9.4
Output and extent	`Output`	§9.5, §9.6
Statistics table	`Range Change Statistics`	§9.8
Map	`Range Change Map Viewer` (right pane)	§9.9

9.3 Selecting the binary maps (Input Binary Projection Rasters)

Name in the UI	What it does	Default	Range / options	Code
`Current projection:` + `Browse…`	Path of the baseline (current) binary map	empty	`.tif .tiff .img .asc`	`tabs/tab_range.py:42-46`, `:167-173`
`Future projection:` + `Browse…`	Path of the (future) binary map to be compared	empty	same	`tabs/tab_range.py:48-52`

- **What it does.** These two fields are the only input to the computation; the quick-fill lists only fill these fields. In the computation, 1 counts as "present" and 0 as "absent"; other values (e.g. a probability of 0.37)

fall into no class (`sdm_core.py:8292-8296`).

- **When and how to set it.** The two maps must have been produced **with the same method**: the same ensemble rule (`EMwmean_bin` with `EMwmean_bin`), the same set of algorithms, the same threshold rule and the same clamping/mask choice. The difference between two maps produced with different methods also carries the effect of the methodological difference, not only of climate change.
- **Effect on results.** Which map is selected is the decision that affects the result most (§9.10).
- **Caution.** Whether the file selected with `Browse…` is binary is not checked; if a `_prob.tif` is selected by mistake, most cells fall into no class and the counts become meaningless. If both fields are empty, the message `Please specify both current and future binary rasters.` appears (`tabs/tab_range.py:200-202`).

9.4 Quick fill (Quick Fill from Completed Projections)

Name in the UI	What it does	Default	Range / options	Code
`Current proj. key:`	Selects the current map from the binary maps in the session	`— select —`	`<scenario> :: <layer>_bin`	`tabs/tab_range.py:62, :128-142`
`Future proj. key:`	Selects the future map from the same list	`— select —`	same	`tabs/tab_range.py:63`
`Fill Paths`	Writes the file paths of the selected keys into the fields above	—	—	`tabs/tab_range.py:64-66, :144-165`

- **What it does.** The lists are built from the layers whose names contain `_bin` among all projections produced in the session (`current` in ③ and every scenario in ⑤) (`tabs/tab_range.py:130-131`): single models (`GLM_PA1_bin`), algorithm means (`GLM_mean_bin`) and ensembles (`EMwmean_bin`, `EMca_bin`). The list is refreshed every time the tab is opened; if it has not changed, the user's selection is kept.
- **When and how to set it.** In the example run, `current :: EMwmean_bin` and `SSP585_2090 :: EMwmean_bin` were selected and `Fill Paths` was pressed. Selecting from the list has two advantages over `Browse…`: there are no path errors, and the NT1 surface needed for the extrapolation share is found with certainty (§9.6.3). In addition, the multi-species batch run (⑧) repeats the key selected from the list, whereas a path chosen by hand points only to that species' file (`tabs/tab_range.py:206-211`).
- **Caution.** If `Fill Paths` is pressed while one of the lists is at `— select —`, the message `Fill Paths takes the paths from the lists above. Choose the … binary map in the list first.` appears; if the lists are empty, running ③ and ⑤ or using `Browse` is suggested (`tabs/tab_range.py:154-165`).

9.5 Analysis name (Output · Analysis name)

Name in the UI	What it does	Default	Range / options	Code
`Analysis name:`	Name of the output file and key of the result in the session	`SRC_current_future`	free text	`tabs/tab_range.py:76-77, :216, :267`

The output is written as `<output folder>\<species>_<analysis name>_SRC.tif` (`tabs/tab_range.py:213-216`). A second run with the same name overwrites both the file and the result in the session; the report opens a separate subsection for each name. If several scenarios are being compared, name the analysis after the scenario (`SSP585_2090` in the example run).

9.6 Extent within which change is counted (Count change within)

Name in the UI	What it does	Default	Range / options	Code
`Count change within:`	Extent of the headline counts and of the map	`Whole study boundary (default)`	`Whole study boundary (default)` / `Accessible area` / `Accessible area without extrapolation`	`tabs/tab_range.py:80-94`, `sdm_core.py:8179-8184`

9.6.1 The three extents

- **What it does.** The selected extent determines the numbers in the upper rows of the table and which cells are coded on the map (`sdm_core.py:8318-8322`):
- `Whole study boundary (default)`: every cell that is valid in both maps.
- `Accessible area`: only cells within the accessible area (`M` in ① Data, the area the species can reach).
- `Accessible area without extrapolation`: cells within the accessible area **and** with ExDet NT1 ≥ 0 in the future scenario (Mesgaran et al. 2014), that is, cells where no variable goes outside its training range (`sdm_core.py:8308-8311`).

Whichever is selected, if the inputs are available the net change of the other extents is also added to the table as separate rows (`tabs/tab_range.py:285-288`). - **When and how to set it.** If the study boundary is much wider than the area the species can reach, most of the predicted change can accumulate in cells that are near no record and outside the training range (the tooltip text says this, `tabs/tab_range.py:85-93`). The concept of the accessible area rests on the BAM framework of Barve et al. (2011). For the main number to be reported, `Accessible area without extrapolation` is the most defensible; the whole study boundary is given alongside it for comparison. - **Effect on results.** In the example run (`EMwmean_bin`) the net change is **−46.13** percent over the whole boundary, **−30.25** in the accessible area and **−16.57** in the accessible area without extrapolation. Depending on the extent, the same two maps give results ranging from "shrinking by half" to "shrinking by one sixth". - **Caution.** If there is no accessible area, the second and third options stop with the message `No accessible area is available for this run, so change cannot be counted within it. …`; if the NT1 surface cannot be found, the third option stops with the message `No ExDet NT1 surface was found next to the future map, …` (`tabs/tab_range.py:224-239`).

9.6.2 Where accessible_area.tif comes from

If an accessible-area method (buffer around records, convex hull or user polygon) has been selected in ① Data, this mask is written during data preparation as a 0/1 GeoTIFF on the training grid to the output folder under the name `accessible_area.tif` (`tabs/tab_data.py:1536-1548`, `sdm_spatial.py:1065-1075`). The mask is built from the presence records remaining after thinning (`sdm_core.py:2971-2974`). In the example run a 200 km buffer around the records was used and the mask covered 40.0% of the raster extent. If no accessible area was selected (buffer 0 km), the file is not written.

9.6.3 Where the NT1 surface comes from

⑤ writes `<species>_<scenario>_NT1.tif` for every projection (§8.9). ⑥ looks up the path of the future map in the projection records of the session and takes the NT1 file of the scenario it finds; if it finds none, it uses the file ending in `_NT1.tif` in the same folder as the future map if there is exactly **one** such file (`tabs/tab_range.py:175-193`). If the future map was selected from another folder with `Browse…`, or if the folder contains more than one NT1 file, NT1 cannot be found.

> **Note:** The extrapolation criterion is only the NT1 of the **future** scenario. The current map may also contain extrapolated cells (23.4% of the current map in the example run); these are not additionally excluded in this extent.

9.7 Result codes and writing of the map

For each cell, and only in cells that are not nodata in both maps (`valid`), the following codes are written (`sdm_core.py:8292-8296`, `:8319-8322`):

Code	Name	Condition	Colour (viewer)
−2	Lost	present now, absent in the future	red
0	Stable Absent	absent now, absent in the future	light grey
1	Stable Present	present now, present in the future	light turquoise
2	Gained	absent now, present in the future	green

Cells outside the selected extent and cells that are nodata in either map remain nodata. If the future map is not on the grid of the current map, it is first aligned by position using the nearest-neighbour method (because it is a binary map; `sdm_core.py:8242-8277`); the accessible area and NT1 are brought to the current grid in the same way. If alignment was done, a warning is written to the log and the job ends with the message `Range change complete; the future raster was not on the current grid and was realigned by position (see log).` (`tabs/tab_range.py:311-318`). The computation is done in row strips according to a memory budget (64 MB; `sdm_core.py:8164-8177`); the result is identical to a single-pass computation.

9.8 Statistics table (Range Change Statistics)

Range Change Statistics

	Metric	Value
1	Counted within	whole study boundary
2	Lost pixels (present → absent)	4037
3	Stable present pixels	3065
4	Stable absent pixels	69308
5	Gained pixels (absent → present)	761
6	Total current range (pixels)	7102
7	Total future range (pixels)	3826
8	% Lost (of current range)	56.84
9	% Gained (of current range)	10.72
10	Net change (%)	-46.13
11	Net change, accessible area (%)	-30.25
12	Net change, accessible area without extrapolation …	-16.57
13	Changed cells in extrapolated conditions (%)	58.63

Figure 9.2. Range Change Statistics table of the example run (EMwmean_bin, whole study boundary): numbers of lost, stable present, stable absent and gained cells, current and future range, percentages, net change of the other extents and the share of changed cells in extrapolated conditions.

All numbers are **cell** counts; they are not converted to area units (km²). In the formulas L is the number of lost cells, G gained, S_1 stable present and S_0 stable absent; $C = L + S_1$ is the current range and $F = G + S_1$ the future range (`sdm_core.py:8187-8193`, `:8312-8317`).

Row (name in the UI)	Definition	Example run
`Counted within`	Name of the selected extent	whole study boundary
`Lost pixels (present → absent)`	L	4037
`Stable present pixels`	S_1	3065
`Stable absent pixels`	S_0	69308
`Gained pixels (absent → present)`	G	761
`Total current range (pixels)`	$C = L + S_1$	7102
`Total future range (pixels)`	$F = G + S_1$	3826
`% Lost (of current range)`	$100\,L/C$	56.84
`% Gained (of current range)`	$100\,G/C$	10.72

Row (name in the UI)	Definition	Example run
`Net change (%)`	$100\,(F - C)/C = \%G - \%L$	−46.13
`Net change, accessible area (%)`	same formula, within the accessible area	−30.25
`Net change, accessible area without extrapolation (%)`	same formula, accessible area ∩ NT1 ≥ 0	−16.57
`Changed cells in extrapolated conditions (%)`	$100\times$ (changed cells with NT1 < 0) / (all changed cells), changed = $L + G$	58.63

The denominator of the percentages is always the **current** range (C); if $C = 0$ the denominator is taken as 1 (`max(tc, 1)`). The gain percentage can therefore exceed 100.

9.8.1 Net change of the other extents

- **What it does.** The headline numbers are given for the selected extent; if the accessible-area file exists, the row `Net change, accessible area (%)` is added automatically, and if NT1 also exists, the row `Net change, accessible area without extrapolation (%)` (`sdm_core.py:8330-8340`). The selected extent's own row is not repeated.
- **Effect on results.** From a single table the reader sees how sensitive the change is to the extent. In the example run there is a difference of about 30 percentage points between the three values.

9.8.2 Extrapolated share of change and the 25% warning

- **What it does.** `Changed cells in extrapolated conditions (%)` tells how many of the lost or gained cells are in places where at least one variable is outside its training range in the future scenario (NT1 < 0) (`sdm_core.py:8299-8304`, `:8341-8343`). This share is computed, independently of the selected extent, over all cells that are valid in both maps.
- **Effect on results.** If the share is 25% or above, `WARNING: x% of the cells that change lie in conditions outside the calibration range (ExDet NT1 < 0); treat the change there as extrapolation.` is written to the log (`tabs/tab_range.py:297-305`); a note with the same meaning also appears in the report (§11.5.9). In the example run the share is 58.63%: more than half of the change is in conditions the model has never seen.
- **Caution.** This row appears only when NT1 is found; it does not appear for a future map selected from another folder with `Browse…`.

9.9 Map

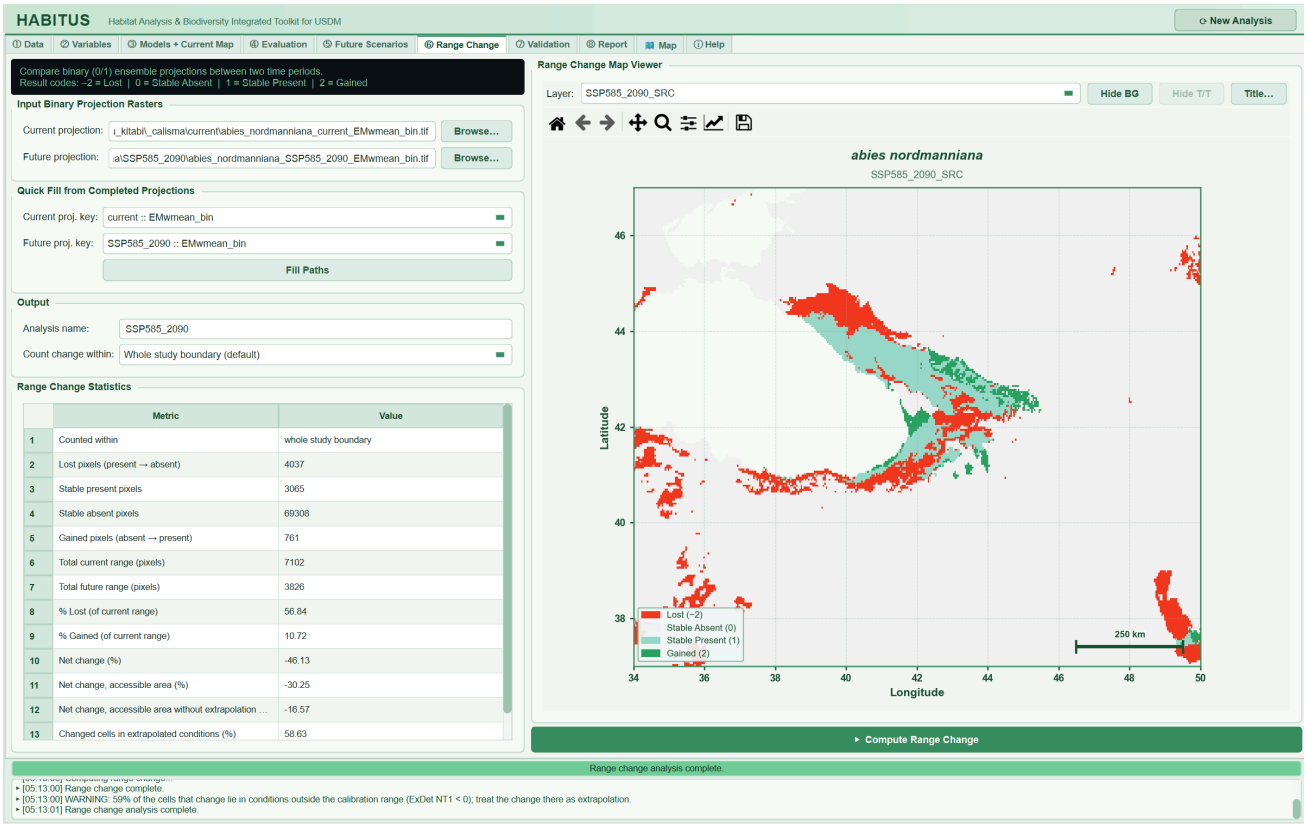


Figure 9.3. Range Change tab after running: on the left the filled paths and the statistics table, on the right the SSP585_2090_SRC map; red lost, light turquoise stable present, green gained cells, legend and 250 km scale bar at the bottom left; in the bottom strip a warning stating that 59% of the changed cells are in extrapolated conditions.

When the job finishes, the `<analysis name>_SRC` layer is loaded into the viewer with the `range_change` style (`tabs/tab_range.py:307-309`). The colours are fixed: red `#f03b20` (−2), light grey `#f0f0f0` (0), light turquoise `#99d8c9` (1), green `#2ca25f` (2) (`map_widget.py:142-146`). The viewer controls and publication output are described in Chapter 12. In the example map, losses are concentrated on the low-lying and western margins of the species' current range and gains at the eastern end; the scattered patches of loss in the south and north-east of the study area lie outside the accessible area.

9.10 Which ensemble map should be chosen: EMwmean_bin or EMca_bin?

The two ensemble binary maps are thresholded with different rules (`sdm_core.py:6423-6446`):

- `EMwmean_bin`: the weighted mean probability map is cut at a threshold that is the mean of the members' own thresholds with the same weights.
- `EMca_bin`: the committee average, that is, the proportion of replicates voting "present"; it is cut at 0.5 (majority vote).

For the same current/future pair of the example run, over the whole study boundary (script: `sekil/_betik/09_alan_degisimi_karsilastirma.py`, using the program's `RangeChangeAnalyzer.compute` function; "mean" is the binary map of the algorithm mean `<ALG>_mean_bin`, ranges are cell counts, "access." accessible area, "extrap." extrapolation):

Binary map	Current	Future	% Lost	% Gained	Net, boundary	Net, access.	Net, no extrap.	Extrap. change
EMwmean	7102	3826	56.84	10.72	−46.13	−30.25	−16.57	58.6%
EMca	7222	3161	65.12	8.89	−56.23	−45.38	−25.05	60.2%
GLM mean	5764	1953	67.52	1.41	−66.12	−62.99	−45.69	43.0%
GAM mean	10077	4194	72.58	14.20	−58.38	−41.24	−28.68	68.4%
MaxEnt mean	9673	3856	68.58	8.45	−60.14	−43.56	−15.74	61.3%
RF mean	7774	8264	58.01	64.32	+6.30	+20.58	−6.41	78.6%

- **Effect on results.** Changing the ensemble rule alone shifted the net change by 10 points (−46 versus −56). The difference between algorithms is much larger: RF predicts expansion (+6), GLM severe contraction (−66). 78.6% of RF's gains are in extrapolated conditions; because tree-based methods stay constant at the last value outside the range, they can leave large areas that move outside the range in the future as "suitable".
- **When and how to set it.** For the main result, the map of the rule selected as the ensemble criterion in ③ is used and its name is stated in the methods. If the results of both ensemble rules and of the algorithm means are given in a supplementary table, the reader sees the source of the uncertainty. For this comparison, each map is run with a separate `Analysis name`.

9.11 Dispersal assumption

- **What it does.** The program counts gained cells wherever they are: the species is assumed to be able to reach every cell that becomes suitable in the future (**unlimited dispersal**). Losses are the same under both assumptions.
- **When and how to set it.** For slowly spreading species such as trees, the **no dispersal** assumption should also be reported (Thuiller et al. 2005). In this case the future range consists only of stable present cells: $F_{extnd} = S_1$ (future range with no dispersal), and the net change is $100\,(S_1 - C)/C = -\%L$. In the example run (`EMwmean_bin`, whole boundary) this means **−56.84** instead of −46.13. The program does not write this number separately; the `% Lost` row of the table with a minus sign is the no-dispersal net change.
- **Caution.** The accessible-area extent behaves partly like a dispersal limit: gains outside the area are not counted. However, because the accessible area is built from current records, cells that lie inside this area but that the species could not reach within a century are still counted. There is no option that applies a specific dispersal rate (km/year).

9.12 Outputs and what passes to the next step

- `<output folder>\<species>_<analysis name>_SRC.tif`: float32, codes −2/0/1/2, nodata that of the current map (otherwise −9999).
- `state["range_results"][<analysis name>] = (statistics, file path)`: the ⑧ report reads this record; grid warnings enter the processing notes of the report (`tabs/tab_report.py:1993-1998`).
- `run_log[<analysis name>]`: keys, paths and extent; the multi-species batch run repeats this record (`tabs/tab_range.py:206-211`, `:240`).

9.13 Expert questions

Q1. Is the computation the same as biomod2's BIOMOD_RangeSize function? Answer: The coding follows the same logic (−2 loss, 0 stable absent, 1 stable present, 2 gain) and the percentages are divided by the current range (`sdm_core.py:8187-8193`). The difference lies in the additional extents: biomod2 counts only over the whole of the maps supplied; HABITUS additionally gives the net change for the accessible area and for the accessible area without extrapolation, and the extrapolated share of the change (`sdm_core.py:8297-8343`). The results are cell counts; for km² they must be multiplied by the cell area, and in geographic coordinates the cell area varies with latitude; the program does not perform this conversion.

Q2. Why is net change equal to "% Gained − % Lost"? Answer: $F - C = (G + S_1) - (L + S_1) = G - L$; since both are divided by C, $\%Net = \%G - \%L$. Example: 10.72 − 56.84 = −46.12 (−46.13 with the rounding difference). This can be used to check the internal consistency of the table.

Q3. Why do cells outside the accessible area not enter the computation at all; shouldn't they be counted as losses? Answer: When `Accessible area` is selected, cells outside the mask are counted in none of the four classes and become nodata on the map (`sdm_core.py:8312-8322`). The logic is this: in a cell the species cannot currently reach, a "present" prediction shows potential suitability, not a realised distribution; a "loss" in these cells is not the loss of an observed range (Barve et al. 2011). The count over the whole boundary is always made; if the main extent is another option, it is added to the table as the row `Net change, whole study boundary (%)` (`sdm_core.py:8330-8340`, `tabs/tab_range.py:285-288`). Only the net change is added; for the loss and gain counts over the whole boundary, the analysis must be run separately with `Whole study boundary`.

Q4. Why is "without extrapolation" defined only by the future NT1? Answer: As NT1 the code uses the surface written by ⑤ for the future scenario (`tabs/tab_range.py:175-193`, `sdm_core.py:8299-8311`). Extrapolated cells in the current map are not additionally excluded. A user who wants to exclude the out-of-range part of the current map as well must combine `current\…_NT1.tif` with the future NT1 outside the program and mask with it; the program has no option for this.

Q5. Is NT2 (novel combinations) used in the change computation? Answer: No; only NT1 < 0 is excluded (`sdm_core.py:8304`, `:8310-8311`). In the example run, cells with NT2 > 1 make up only 0.3% of the future scenario, so the effect is small. In a study where this share is high, the NT2 surface can be used as an additional mask outside the program.

Q6. Is the result reliable if the future map is on a different grid? Answer: The future map is aligned by position to the grid of the current map: with the same cell size, a whole-cell offset is shifted without interpolation, otherwise it is resampled with nearest neighbour (`sdm_core.py:8242-8277`, `sdm_grid.py:244-280`). The alignment is written to the log and to the processing notes of the report. Because the maps produced in ⑤ are already on the training grid, no alignment is needed in normal use.

Q7. Why does the result change with a future map that has the MESS < 0 mask turned on? Answer: Only cells valid in both maps are counted (`sdm_core.py:8292`). Masked future cells drop out of the computation; the area that is "present" today in these cells also drops out of C. The result resembles the "Accessible area without extrapolation" option but is not the same: that option additionally restricts to the accessible area and masks the count, not the map.

Q8. How does the choice of threshold affect range change? Answer: The binary maps are produced with the threshold rule in ③ (default max TSS), and the same threshold is used for both periods (`sdm_core.py:7908`). As the threshold falls, the "present" area grows in both maps; because the percentages are divided by the current range, the relative change can move in different directions depending on the threshold. In the example run, the net change differs by 10 points between `EMwmean_bin` and `EMca_bin`, which

differ only in the thresholding rule (§9.10). Threshold sensitivity is tested by rebuilding with a different threshold rule in ③ and repeating ⑥.

Q9. Does a cell being "present" mean that the species will be there? Answer: No. The binary map shows the cells where the suitability value exceeds the threshold; in a presence–pseudo-absence model this means "climatically suitable". Biotic interactions, land use and dispersal limits are not in the model. Gained cells can be counted as new range only under the unlimited dispersal assumption (§9.11).

Q10. Is the result reproducible? Answer: The computation has no random component; the same two maps give the same count. The selected extent, analysis name and input paths are written to `run_log` and the results to `state["range_results"]`; the report contains the table and the map for each analysis. The comparison table in this chapter can be reproduced with `sekil/_betik/09_alan_degisimi_karsilastirma.py`, which calls the program's own function.

9.14 Common mistakes and pitfalls

- **Comparing two maps produced with different methods.** If `EMwmean_bin` is selected for the present and `EMca_bin` for the future, part of the change comes from the method.
- **Selecting a probability map with `Browse…`.** The program does not warn; cells that fall into no class are not counted.
- **Reporting the whole-boundary result as the only result.** In the example run the whole boundary gives −46 and the accessible area without extrapolation −17 percent; which one is reported should be stated explicitly.
- **Not stating the dispersal assumption.** The net change in the table assumes unlimited dispersal; the no-dispersal value is the `% Lost` row with a minus sign.
- **Ignoring the extrapolated share of changed cells.** A share above 25% says that an important part of the result rests on conditions the model has never seen.
- **Taking the number of cells for area.** The table counts cells; for km² multiply by the cell area (on a geographic grid it varies with latitude).
- **A future map selected from another folder.** NT1 cannot be found; the extrapolation share row and the third extent cannot be used.

9.15 References

Barve, N., Barve, V., Jiménez-Valverde, A., Lira-Noriega, A., Maher, S.P., Peterson, A.T., Soberón, J., Villalobos, F., 2011. The crucial role of the accessible area in ecological niche modeling and species distribution modeling. Ecological Modelling 222, 1810–1819. https://doi.org/10.1016/j.ecolmodel.2011.02.011

Mesgaran, M.B., Cousens, R.D., Webber, B.L., 2014. Here be dragons: a tool for quantifying novelty due to covariate range and correlation change when projecting species distribution models. Diversity and Distributions 20, 1147–1159. https://doi.org/10.1111/ddi.12209

Thuiller, W., Lafourcade, B., Engler, R., Araújo, M.B., 2009. BIOMOD – a platform for ensemble forecasting of species distributions. Ecography 32, 369–373. https://doi.org/10.1111/j.1600-0587.2008.05742.x

Thuiller, W., Lavorel, S., Araújo, M.B., Sykes, M.T., Prentice, I.C., 2005. Climate change threats to plant diversity in Europe. Proceedings of the National Academy of Sciences 102, 8245–8250. https://doi.org/10.1073/pnas.0409902102

10. Independent validation (⑦ Validation)

Tab: ⑦ Validation · Code: `tabs/tab_validation.py` (UI and computation), `sdm_grid.py` `read_on_grid` (grid alignment), `tabs/tab_report.py` `_s9_validation` (section in the report)

10.1 What this tab does

⑦ Validation compares a map produced by the model with a reference source that is independent of the model and measures map accuracy. The reference source is either a raster map (an expert-drawn range map, a published distribution map, a presence/absence layer derived from field data) or field observations whose points and reference values are given in a CSV file. The comparison is made at sample points selected on the map; the result is a confusion matrix and the metrics derived from it.

- **Input:** two GeoTIFFs (the reference and the map to be validated) or one GeoTIFF and one CSV. The map to be validated is usually a map written by ③ Models (for example, the ensemble binary map `EMwmean_bin`).
- **Output:** the result text (metrics, confusion matrix, per-class metrics), an optional report (`.txt`, `.json`, `.html`) and a points table (`.csv`), and two reclassified rasters (`validation/` folder). The latest validation result passes to section 9 of ⑧ Report.
- **Lock order:** the tab opens once ③ Models has finished running; at the same time the rasters in the output folder are loaded automatically into the two drop-down boxes (`tabs/tab_models.py:1848-1856`).

The validation in this tab differs from the cross-validation scores in ④ Evaluation: ④ measures how well the model discriminates on its own data, using folds split from the training records; ⑦ compares the map with an external source. Neither replaces the other.

10.2 Screen layout

From top to bottom the tab consists of four numbered groups.

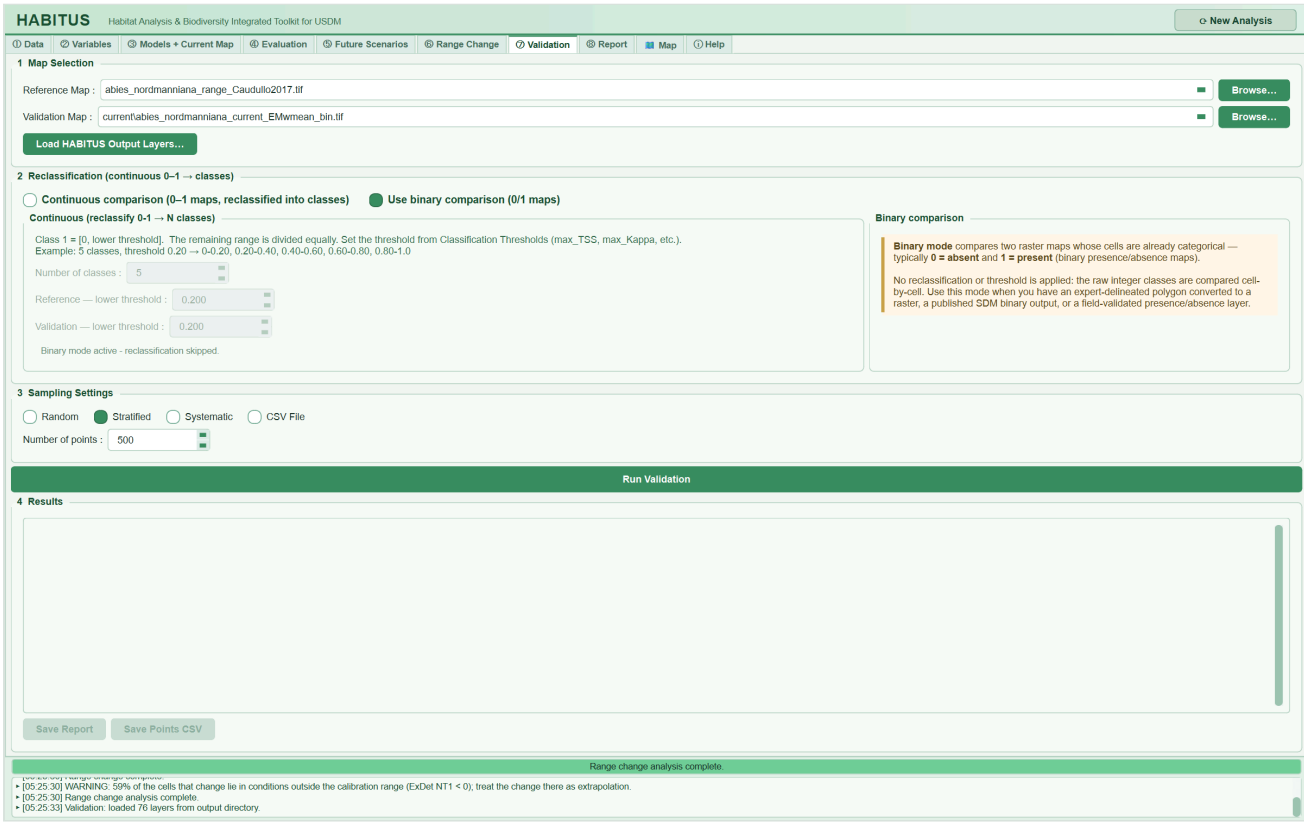


Figure 10.1. The Validation tab before running: 1 Map Selection (reference map Caudullo et al. 2017, validated map the ensemble binary map), 2 Reclassification (binary mode selected, continuous-mode fields greyed out), 3 Sampling Settings (Stratified, 500 points), the Run Validation button and the empty 4 Results box.

Region	What it contains	Subsection
`1 Map Selection`	`Reference Map`, `Validation Map`, two `Browse…` buttons, `Load HABITUS Output Layers…`	§10.3
`2 Reclassification (continuous 0–1 → classes)`	Two mode buttons; the `Continuous (reclassify 0-1 → N classes)` settings on the left, the `Binary comparison` description on the right	§10.4
`3 Sampling Settings`	`Random`, `Stratified`, `Systematic`, `CSV File`; CSV path; `Number of points`	§10.5
`Run Validation`	Runs the comparison	§10.6
`4 Results`	Result text, `Save Report`, `Save Points CSV`	§10.6–10.8

Figures 10.1 and 10.2 are taken from the example run: *Abies nordmanniana*; the reference map is the range map of Caudullo et al. (2017) converted to the model grid (`abies_nordmanniana_range_Caudullo2017.tif`), and the validated map is the ensemble binary map (`current\abies_nordmanniana_current_EMwmean_bin.tif`).

10.3 Map selection (1 Map Selection)

Two maps are selected: the **reference map** (the source taken as correct) and the **map to be validated** (the model's map). In the UI the latter is called `Validation Map`; in this chapter it is referred to as the "validated map".

Name in the UI	What it does	Default	Range / options	Code
`Reference Map :`	Reference raster	first layer in the list	layers in the list	`tabs/tab_validation.py:64-72`
`Validation Map :`	Validated raster	first layer in the list	layers in the list	`tabs/tab_validation.py:74-82`
`Browse…` (two)	Selects a GeoTIFF, adds it to the corresponding list and selects it	—	`*.tif`, `*.tiff`	`tabs/tab_validation.py:341-350`
`Load HABITUS Output Layers…`	Adds all `.tif` files in the output folder to both lists	automatic when the models finish	button	`tabs/tab_validation.py:358-384`

10.3.1 Loading the output layers (`Load HABITUS Output Layers…`)

- **What it does.** Scans the output folder together with its subfolders and adds every `.tif` / `.tiff` file it finds to both lists **in the same order**, with a name relative to the output folder (e.g. `current\abies_nordmanniana_current_EMwmean_bin.tif`). It does not duplicate entries already in the list. It writes "Validation: loaded N layers from output directory." to the log; in the example run 76 layers were loaded.
- **When and how to set it.** When ③ Models finishes, the program presses this button itself; it is pressed again to add maps written later, such as future scenarios or range change maps.
- **Effect on results.** A selection convenience only.
- **Caution.** The list contains all rasters in the output folder: probability and binary maps, individual model maps, uncertainty (`EMsd`), novelty (`NT1`, `NT2`) and the accessible-area raster. Choosing the right file is up to the user. If there is no output folder, the warning "No output directory set yet. Run the analysis first." appears. The reference map is usually not in the output folder; it is added with `Browse…`.

10.3.2 The grids of the two maps

The comparison is made on the grid of the validated map. The reference raster is read onto this grid **by position** (`tabs/tab_validation.py:422-455`, `sdm_grid.py:188-283`):

- If the grids are identical, the raster is read as it is and no note is produced.
- If the cell size and coordinate system are the same and the grid is only shifted by a whole number of cells, it is copied with a shift, without interpolation.
- In every other case it is resampled with `rasterio.warp.reproject`: nearest neighbour in binary mode, bilinear interpolation in continuous mode.

If an alignment was made, a note of the following form is written to the log and to the result text ("NOTE (grid)"): "Reference raster (W×H px) was not on the validation raster's grid (W×H px) and was realigned to it by position (…); N validation cell(s) not covered (x%)." If the cells not covered by the reference exceed 5% of the validated grid, an additional warning appears: "… of the validation grid lies outside the reference raster (> 5%); those cells are not compared. Check that both maps cover the same area." (`tabs/tab_validation.py:447-`

`452`, `sdm_grid.py:40`). These notes are also passed to the "Processing notices" section of the report (`state["validation_notices"]`).

> **Caution:** In continuous mode the reference map is resampled with bilinear interpolation. If the reference is a 0/1 map, intermediate values between 0 and 1 arise along the boundaries and fall into the intermediate classes. Compare a binary reference in binary mode, or prepare it on the same grid so that no resampling to the model is needed at all.

10.4 Comparison mode and reclassification (2 Reclassification)

There are two modes. The mode buttons are active in both cases; when binary mode is selected, the continuous-mode settings are greyed out and the preview line reads "Binary mode active - reclassification skipped." (`tabs/tab_validation.py:191-198`, `316-321`).

Name in the UI	What it does	Default	Range / options	Code
`Continuous comparison (0–1 maps, reclassified into classes)`	Reclassifies both maps into N classes and compares them	selected	mode button	`tabs/tab_validation.py:100-103`
`Use binary comparison (0/1 maps)`	Rounds the cell values and compares them as they are	—	mode button	`tabs/tab_validation.py:102`, `599-603`, `672-677`
`Number of classes :`	Number of classes in continuous mode	5	2–10	`tabs/tab_validation.py:132-137`
`Reference — lower threshold :`	Upper bound of class 1 in the reference map	0.200	0.000–0.990, step 0.01	`tabs/tab_validation.py:139-145`
`Validation — lower threshold :`	Upper bound of class 1 in the validated map	0.200	0.000–0.990, step 0.01	`tabs/tab_validation.py:147-153`
`Ref breaks:` / `Val breaks:` line	Live preview of the class boundaries	—	—	`tabs/tab_validation.py:155-160`, `316-328`

10.4.1 Continuous comparison (`Continuous comparison`)

- **What it does.** Both maps are divided into N classes according to their values in the 0–1 range. Boundaries: $[0,\ t,\ \ldots,\ 1]$; class 1 is $[0, t)$, and the remaining interval between t and 1 is divided into $N - 1$ **equal** parts; the last class also includes 1 (`tabs/tab_validation.py:330-339`, `461-474`). If $t = 0$, the boundaries are equally spaced between 0 and 1. Values below 0 and above 1 fall into no class and are excluded from the comparison. Class labels have the form "C1 [0.000–0.200]".
- **When and how to set it.** When both maps are genuinely continuous (probability or suitability): for example, when comparing the probability maps of two models, or the model's probability map with the suitability map of another study. A threshold from the `Classification Thresholds` table of ④ Evaluation (such as max_TSS) can be used as the lower threshold; class 1 then means "not suitable".
- **Effect on results.** As the number of classes increases, exact agreement becomes harder; overall accuracy and Kappa decrease. Different lower thresholds can be given for the reference and the validated map; if two models have different thresholds, this equalizes the meaning of the classes. In continuous mode, regression metrics are also computed from the raw values at the points (§10.7.3).
- **Caution.** If the reference is a 0/1 map, only the first and last classes are filled and the intermediate classes remain empty in the reference; the diagonal of the confusion matrix is mostly concentrated in class 1 and the metrics come out lower than in binary mode. The two modes do not measure the same thing. Use binary mode for a binary reference.

10.4.2 Binary comparison (`Use binary comparison`)

- **What it does.** The cell values of both maps are rounded to integers and compared cell by cell without applying any threshold (`np.round`; `tabs/tab_validation.py:601`, `674`). The classes are taken from the data and labelled "Class 0", "Class 1". If there are two classes, the counts of true/false positives and negatives, sensitivity, specificity and TSS are added to the result text (§10.7.2).
- **When and how to set it.** When the reference is a presence/absence map and the validated map is the model's binary map (`…_bin.tif`). The description in the UI lists three typical references: a rasterized expert-drawn polygon, a published binary SDM output, a field-verified presence/absence layer.
- **Effect on results.** In the example run (Caudullo et al. 2017 reference, `EMwmean_bin`, 500 stratified points), overall accuracy is 0.866 and Kappa 0.732 (Figure 10.2).
- **Caution.** In this mode values are rounded at 0.5; if a **probability** map (`…_prob.tif`) is selected while binary mode is on, the map is silently split into two at the 0.5 threshold, not at the model's own threshold. A value of exactly 0.5 also goes to 0 because of NumPy's round-half-to-even rule. In binary mode always select the `…_bin.tif` files. If the labels are other than 0/1 (e.g. 1/2), TP/FP/TN/FN are assigned on the assumption that the smaller code is "absence" (`tabs/tab_validation.py:908-911`).

10.5 Sampling design (3 Sampling Settings)

The comparison is made not at all cells but at selected sample points. This is established practice in accuracy assessment (Congalton 1991; Olofsson et al. 2014): the result is an accuracy estimated from a sample.

Name in the UI	What it does	Default	Range / options	Code
`Random`	Random selection without replacement from the valid cells of the classified reference	selected	method button	`tabs/tab_validation.py:485-492`
`Stratified`	An equal number of cells from each class of the reference	—	method button	`tabs/tab_validation.py:494-505`
`Systematic`	A regularly spaced grid over the raster	—	method button	`tabs/tab_validation.py:507-520`
`CSV File`	Reference values from points in a CSV	—	method button	`tabs/tab_validation.py:526-544`, `622-651`
`CSV (id, x, y, reference_value):` + `Browse`	Path of the CSV file (visible only when `CSV File` is selected)	empty	`*.csv`	`tabs/tab_validation.py:218-229`
`Number of points :`	Requested number of points	500	30–100,000, step 50	`tabs/tab_validation.py:231-239`

10.5.1 Random (`Random`)

- **What it does.** `Number of points` cells are selected without replacement from all valid (non-NaN) cells of the classified reference map. The random number generator is a local `numpy.random.default_rng` derived from the run seed (`random_seed`; 42 if unknown); the same seed gives the same points, and the result text reads "Random (seed 42)" (`tabs/tab_validation.py:413-420`, `699-701`).
- **When and how to set it.** It is the default and most easily interpreted design for estimating accuracy over the whole map: every cell has an equal probability of selection, so overall accuracy and Kappa are directly area-weighted estimates.
- **Effect on results.** The rare class (the species usually occupies a small part of the area) is represented by few points; the producer's and user's accuracies of that class are uncertain.
- **Caution.** A selected cell that is NaN in the validated map is discarded; the number of valid points may therefore be smaller than requested. The `Points` line in the result text gives the number actually used.

10.5.2 Stratified (`Stratified`)

- **What it does.** $\max(1, \lfloor n/K \rfloor)$ cells are selected from each class of the reference map (K is the number of classes); if a class does not have that many cells, all of them are taken (`tabs/tab_validation.py:494-505`). The seed is the same as in the random design.
- **When and how to set it.** When the producer's and user's accuracy of each class needs to be estimated with enough points, especially when the presence class is a small part of the area.
- **Effect on results.** Because the classes are represented by equal numbers of points, overall accuracy and Kappa are **not area-weighted**: each class counts with the same weight regardless of its share of the map. In the example run only 5.8% of the valid cells of the reference map are presence (4458 of 77,171 cells); the 500 stratified points, however, are half presence and half absence. From this sample, overall accuracy is 0.866 and Kappa 0.732. Over all common cells of the same two maps (77,171 cells), overall accuracy is 0.938 and Kappa 0.553 (computed for this book from the two reclassified rasters under `_calisma/validation/`). Sensitivity (0.78 and 0.76) and specificity (0.95 and 0.95), on the other hand, are close in the two computations, because they are within-class proportions.
- **Caution.** For an area-weighted estimate of overall accuracy and Kappa under a stratified design, the result of each class must be weighted by its area share in the map (Olofsson et al. 2014). The program does not perform this weighting; use the per-class metrics (producer's/user's accuracy, sensitivity, specificity, TSS) or weight the overall accuracy yourself using the `Save Points CSV` output and the class areas. If n is not exactly divisible by K, the total number of points is smaller than n.

10.5.3 Systematic (`Systematic`)

- **What it does.** With $g = \lfloor \sqrt{n} \rfloor$, a regularly spaced $g \times g$ grid is laid over the full row and column range of the raster; of the grid points, those valid in the reference are taken, stopping when n is reached (`tabs/tab_validation.py:507-520`). There is no randomness; no seed is used.
- **When and how to set it.** When regular coverage of the area is desired. It reduces the spatial autocorrelation of nearby points.
- **Effect on results.** Because the grid is laid over the entire rectangle of the raster, points falling in NoData areas (sea, outside the study extent) are discarded; the number of valid points is often markedly smaller than n. When 500 are requested, the grid is 22 × 22 = 484 points.
- **Caution.** If there is a periodic structure in the raster layout (roads, grid-shaped land use), the systematic sample may coincide with this structure.

10.5.4 CSV points (`CSV File`)

- **What it does.** Reference values are taken not from a map but from points in a CSV file. The file must be UTF-8 and its header row must contain the columns `x`, `y`, `reference_value` (optionally `id`); names are case-sensitive (`tabs/tab_validation.py:526-544`). Each point is converted to a cell with the transform of the validated raster; rows falling outside the raster, non-numeric rows or rows with a missing column are silently skipped. `reference_value` is classified with the **reference class boundaries** (`Reference — lower threshold` and `Number of classes`); the validated class is read from the point's cell in the validated map (`tabs/tab_validation.py:636-648`).
- **When and how to set it.** For validation with field observations, separately collected presence/absence points or point data from another study. Coordinates must be in the coordinate system of the validated raster (degrees in the example data; longitude `x`, latitude `y`).
- **Effect on results.** `Number of points` is not used with this method; all valid points in the file are used. The reclassified rasters are not written in this mode.
- **Caution.** No coordinate transformation is performed; points in a different coordinate system fall into the wrong cells or lie outside the raster and are discarded. The decimal separator must be a point (values such as "41,73" cannot be read as numbers and the row is skipped). Because `reference_value` is classified on the 0–1 scale as in continuous mode, 0/1 field data fall into the first and last classes in continuous mode; for a binary comparison with these data, see the pitfall in §10.10.

10.6 Running (Run Validation)

`Run Validation` runs the job in the UI thread and, while it runs, opens a small window showing the steps (`tabs/tab_validation.py:550-558`). The steps are written to the result box with "▸ ": the number of valid cells of the validated raster, its minimum/maximum/mean value and number of unique values; the class boundaries; the number of cells per class; the same for the reference; the sampling and the number of valid points. This text is then replaced by the result report (`tabs/tab_validation.py:963`). If no validated map is selected, the message "Please select a Validation Map." appears; if no reference is selected, "Please select a Reference Map."; if there are no common valid points, "No valid overlapping points found."

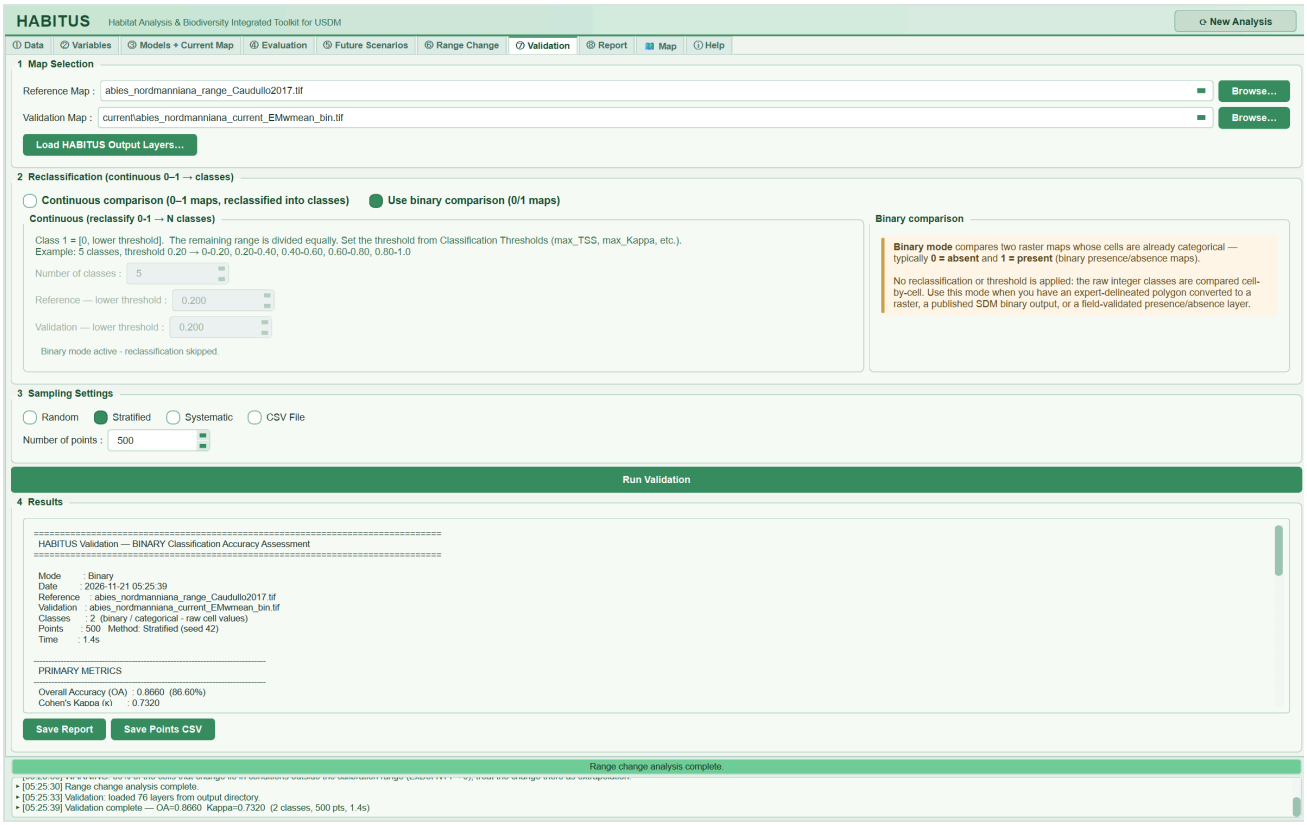


Figure 10.2. The Validation tab after running: binary mode, 500 stratified points (seed 42); in the 4 Results box, Mode, Classes, Points and, under the PRIMARY METRICS heading, overall accuracy 0.8660 and Kappa 0.7320; the Save Report and Save Points CSV buttons are enabled.

10.7 Metrics and the result text

A confusion matrix is built with scikit-learn from the (reference class, validated class) pairs at the sample points; rows are the reference, columns the validated map (`tabs/tab_validation.py:749-779`). The class list is the union of the classes observed on the two sides.

10.7.1 Primary metrics (`PRIMARY METRICS`)

Name in the result text	Definition	Code
`Overall Accuracy (OA)`	Proportion of correctly classified points	`accuracy_score`, `tabs/tab_validation.py:754`
`Cohen's Kappa (κ)`	Agreement corrected for chance agreement (Cohen 1960)	`cohen_kappa_score`, `:755`
`F1-Score (Macro)` / `F1-Score (Weighted)`	Simple / support-weighted mean of the per-class F1	`f1_score`, `:756-757`
`Precision (Macro)` / `Recall (Macro)`	Simple mean of the per-class precision / recall	`:758-759`
`Kappa Interpretation`	Verbal interpretation of Kappa (below)	`:900-904`

For all ratios, if the denominator is zero the result is taken as 0 (`zero_division=0`). Kappa is computed as

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

where p_o is the observed agreement (overall accuracy) and p_e the chance agreement expected from the class proportions of the two maps.

Kappa interpretation scale. The result text verbalizes Kappa with the following scale (`tabs/tab_validation.py:900-904`): $\kappa < 0$ "Poor"; < 0.20 "Slight"; < 0.40 "Fair"; < 0.60 "Moderate"; < 0.80 "Substantial"; otherwise "Almost Perfect". This is the scale proposed by Landis and Koch (1977). The scale is a convention proposed for inter-observer agreement; it does not provide definitive limits for map accuracy.

> **Caution:** Kappa depends on the prevalence of the presence class (Allouche et al. 2006). The same two maps can give very different Kappa values under different sampling designs: in the example run 0.732 with the stratified sample and 0.553 over all cells (§10.5.2). Do not report Kappa alone; report sensitivity, specificity and TSS, together with the sampling design used.

10.7.2 Binary summary (`BINARY SUMMARY`)

In binary mode, and if there are exactly two classes, the following are added to the result text (`tabs/tab_validation.py:906-922`): the confusion matrix is read in the order [0, 1]; `True Positive (TP)`, `False Positive (FP)`, `True Negative (TN)`, `False Negative (FN)`; `Sensitivity` = TP/(TP+FN); `Specificity` = TN/(TN+FP); `True Skill Stat TSS` = sensitivity + specificity − 1 (Allouche et al. 2006). TSS is independent of prevalence; in binary validation it is a more reliable summary than Kappa.

Confusion matrix of the example run (stratified, 500 points; from the report file): of the 250 points that are absence in the reference, 238 are also absence in the model and 12 are presence; of the 250 points that are presence in the reference, 55 are absence in the model and 195 are presence. This gives sensitivity 0.780, specificity 0.952 and TSS 0.732. When equal numbers of presences and absences are sampled, Kappa and TSS take the same value; this is the reason for the equality in the example.

10.7.3 Regression metrics (continuous mode only)

In continuous mode these are computed from the **raw** values at the points (reference and validated, 0–1) (`tabs/tab_validation.py:761-767`, `923-927`): `R^2` (`r2_score`, coefficient of determination; it is not Pearson's r^2 and can be negative), `RMSE`, `MAE`, `Bias` = mean(validated − reference); "(Over)" is appended if it is positive and "(Under)" if negative. If the reference is a 0/1 map, these metrics measure the distance of a probability map from 0/1 and are of limited meaning.

10.7.4 Confusion matrix and per-class metrics

The result text writes three more tables (`tabs/tab_validation.py:929-958`):

- `CONFUSION MATRIX`: rows are the reference classes ("Ref \ Val"), columns the validated classes.
- `PER-CLASS METRICS`: for each class `Precision`, `Recall`, `F1`, `Support` (number of reference points in that class).
- `PRODUCER'S & USER'S ACCURACY`: producer's accuracy (the proportion of a reference class correctly shown on the map) = recall; user's accuracy (the proportion of a map class that is correct on the ground) = precision (Congalton 1991).

The beginning of the result text gives the mode, the date, the names of the reference and validated files, the number of classes, in continuous mode the two thresholds and the class boundaries, the number of points and the method (with the seed), grid notes if any, and the duration.

10.8 Export

Name in the UI	What it does	Default	Code
`Save Report`	Saves the result, depending on the extension, as `.txt` (result text), `.json` (full result dictionary) or `.html` (summary and confusion matrix)	`validation_report_<date_time>`	`tabs/tab_validation.py:970-992`, `1021-1066`
`Save Points CSV`	Saves each point: `id`, `x`, `y`, `ref_value`, `val_value`, `ref_class`, `val_class`, `match`	`validation_points_<date_time>.csv`	`tabs/tab_validation.py:994-1019`
(automatic)	In raster–raster mode, `validation/Reclassify_ref_<name>.tif` and `validation/Reclassify_val_<ad>.tif`	—	`tabs/tab_validation.py:726-730`, `819-839`

Both buttons are disabled until the first successful run. The reclassified rasters are written as float32 with NoData = −9999; if the reference was aligned, it takes the georeferencing of the validated raster. The JSON file contains the class boundaries, the confusion matrix, the class report, the seed and the grid notes; it is the most complete record for later auditing of a validation.

Section 9 of ⑧ Report uses the **latest** validation result in the tab (`tabs/tab_report.py:2269-2320`): mode, reference and validated layer, number of points, overall accuracy, Kappa, weighted F1, R^2, RMSE, MAE (if available), the confusion matrix and the reclassified maps. The sampling method is not included in this section; add it when writing the report.

10.9 Expert questions

Q1. Why is this validation considered "independent"; how does it differ from the scores in ④ Evaluation? ④ Evaluation is computed on cross-validation folds derived from the model's training records; the same record source and the same pseudo-absence design are used. ⑦ Validation, by contrast, compares the map with an external source (`tabs/tab_validation.py:550-817`). Independence depends on the nature of the reference: if the reference is a range map derived from the model's training records (for example, drawn from the same herbarium records), it cannot be considered fully independent. The program does not check the origin of the reference; the user must state it.

Q2. Do overall accuracy and Kappa obtained with stratified sampling reflect the accuracy of the map? Not directly. `Stratified` takes an equal number of points from each reference class (`tabs/tab_validation.py:494-505`) and does not weight the results by area share. In the example run, overall accuracy and Kappa are 0.866 and 0.732 at the 500 stratified points, and 0.938 and 0.553 over all common cells (§10.5.2). For an area-weighted estimate, the stratified estimators of Olofsson et al. (2014) should be used; the program does not compute them. Within-class proportions (sensitivity, specificity, TSS) are unbiased under the stratified design as well, and reporting them is recommended.

Q3. Which metric should be reported instead of Kappa? Kappa depends on prevalence (Allouche et al. 2006) and has been criticized in the map accuracy literature; Pontius and Millones (2011) recommend reporting quantity and allocation disagreement separately. The program does not compute

quantity/allocation disagreement. In binary mode, TSS, sensitivity and specificity are in the result text (`tabs/tab_validation.py:906-922`); if these and the confusion matrix are reported, readers can compute the other metrics themselves. Quantity and allocation disagreement can be computed from the `Save Points CSV` output or from the two reclassified rasters.

Q4. What happens if the reference and the validated map have different grids? The reference is read onto the grid of the validated map by position: an identical grid as it is, a whole-cell shift by shifting, every other case by resampling (nearest neighbour in binary mode, bilinear in continuous mode) (`sdm_grid.py:188-283`). The operation performed and the proportion of cells not covered are written to the result text as "NOTE (grid)"; a coverage gap exceeding 5% triggers an additional warning (`tabs/tab_validation.py:436-452`). If resampling involves a difference in scale (for example, a 1 km reference onto a 4.6 km model grid), nearest-neighbour selection alters the boundary cells of the reference; scale compatibility is the user's responsibility.

Q5. Over which area is sampling done; is it restricted to the accessible area? No. `Random` and `Stratified` select from all cells that are valid in the classified reference; cells that are NaN in the validated map are then discarded (`tabs/tab_validation.py:486-492, 709-722`). The sampling area is thus the entire grid on which both maps have values; it is not restricted to the accessible area of ① Data. Distant, clearly unsuitable cells outside the accessible area are easily classified absences and inflate overall accuracy. To restrict the validation to the accessible area, the reference map must be set to NoData outside this area.

Q6. Is the result reproducible? `Random` and `Stratified` use a local generator built from the run seed (`random_seed`; 42 if unknown) (`tabs/tab_validation.py:413-420, 699`); the same seed, the same maps and the same number of points give the same points. The seed is written to the result text ("Stratified (seed 42)") and to the JSON report. `Systematic` and `CSV File` involve no randomness.

Q7. Are 500 points enough; what is the uncertainty of the result? The program does not give a confidence interval for overall accuracy or Kappa. Because the result is estimated from a single sample, it carries sampling uncertainty. Under simple random sampling the standard error of overall accuracy is approximately $\sqrt{p(1-p)/n}$; for $p = 0.9$ and $n = 500$ this is about 0.013. The accuracy of a rare class requires a stratified design or more points (Olofsson et al. 2014). The variability can be seen by repeating with different seeds or by increasing `Number of points` (up to 100,000).

Q8. How should I compare a probability map with a binary reference? The most correct way is to use, in binary mode, the map obtained by splitting the probability map at the model's selected threshold (`…_bin.tif`). If a probability map is selected in binary mode, the values are rounded, i.e. a 0.5 threshold is applied (`tabs/tab_validation.py:601`); this is not the model's threshold. In continuous mode, on the other hand, a 0/1 reference falls only into the first and last classes and the metrics measure a mixture of two things (§10.4.1).

Q9. How should I use field presence/absence data when validating with CSV points? With the CSV method, `reference_value` is always classified with the reference class boundaries (as in continuous mode) (`tabs/tab_validation.py:642`), whereas the validated map is read in binary mode as rounded 0/1 values. In binary mode these two codings do not match (reference 1 and N, map 0 and 1) and the result is meaningless. To compare field 0/1 data correctly, use continuous mode with `Number of classes` = 2 and both lower thresholds set to the same value (e.g. 0.500): the 0 and 1 of the reference then fall into classes 1 and 2 respectively, and the validated probability map is split into two classes at the same threshold. If the validated map is binary (`…_bin.tif`), the threshold can be any value between 0 and 1.

Q10. What are the limitations of using a range (chorology) map as the reference map? Range maps such as Caudullo et al. (2017) are expert drawings and usually show the outer limit of the species' native range; places within the limit where the species does not occur (unsuitable elevation, land use) are also counted as

"presence". This increases the number of false negatives in cells that the model does not find suitable but that lie within the range. In the example run the model has more presence cells (7102) than the reference (4458); some of the places the model shows "in excess" may be climatically suitable but not reached by the species. This comparison therefore measures map agreement; it is not definitive proof of the model's biological accuracy.

Q11. Is the result passed to the report automatically, and what is missing? Yes; section 9 of ⑧ Report uses the latest validation result (`tabs/tab_report.py:2269-2320`). The report does not state the sampling method or the seed; this information is in the JSON and text reports. The phrase "κ values above 0.60 are generally considered acceptable …" in the interpretive sentence of the report is a generic formula; it cites no source and does not take into account the prevalence dependence of Kappa. Before publishing the report, add the sampling design and adapt the interpretation.

10.10 Common mistakes and pitfalls

- **Selecting a probability map in binary mode.** Values are rounded at 0.5; the model's threshold is not used. In binary mode select `…_bin.tif`.
- **Comparing a binary reference in continuous mode.** The intermediate classes remain empty and Kappa drops artificially (§10.4.1).
- **Reporting the Kappa of a stratified sample as map accuracy.** This is the source of the difference between 0.732 and 0.553 in the example run (§10.5.2). State the design explicitly and report the within-class metrics.
- **Wrong column name or decimal comma in the CSV.** The columns must be `x`, `y`, `reference_value` and numbers must use a decimal point; non-conforming rows are skipped without warning. If no valid row remains, the message "No valid points in CSV." appears.
- **CSV coordinates in a different coordinate system.** No transformation is performed; points fall into the wrong cell or are discarded.
- **0/1 field data from a CSV in binary mode.** The codes do not match (§10.9, Q9); use continuous mode, 2 classes and equal thresholds.
- **Expecting the requested number of points with `Systematic`.** The grid is laid over the whole rectangle; points falling in NoData are discarded.
- **Different extents of the reference and the validated map.** Read the "NOTE (grid)" note and the coverage-gap warning; the comparison is made only in the common cells.
- **Forgetting that of successive validations only the last passes to the report.** Run the comparison you want to see in the report last; save the other results separately with `Save Report`.

10.11 References

Allouche, O., Tsoar, A., Kadmon, R., 2006. Assessing the accuracy of species distribution models: prevalence, kappa and the true skill statistic (TSS). Journal of Applied Ecology 43, 1223–1232. https://doi.org/10.1111/j.1365-2664.2006.01214.x

Caudullo, G., Welk, E., San-Miguel-Ayanz, J., 2017. Chorological maps for the main European woody species. Data in Brief 12, 662–666. https://doi.org/10.1016/j.dib.2017.05.007

Cohen, J., 1960. A coefficient of agreement for nominal scales. Educational and Psychological Measurement 20, 37–46. https://doi.org/10.1177/001316446002000104

Congalton, R.G., 1991. A review of assessing the accuracy of classifications of remotely sensed data. Remote Sensing of Environment 37, 35–46. https://doi.org/10.1016/0034-4257(91)90048-B

Landis, J.R., Koch, G.G., 1977. The measurement of observer agreement for categorical data. Biometrics 33, 159–174. https://doi.org/10.2307/2529310

Olofsson, P., Foody, G.M., Herold, M., Stehman, S.V., Woodcock, C.E., Wulder, M.A., 2014. Good practices for estimating area and assessing accuracy of land change. Remote Sensing of Environment 148, 42–57. https://doi.org/10.1016/j.rse.2014.02.015

Pontius, R.G., Millones, M., 2011. Death to Kappa: birth of quantity disagreement and allocation disagreement for accuracy assessment. International Journal of Remote Sensing 32, 4407–4429. https://doi.org/10.1080/01431161.2011.552923

11. Report, ODMAP and multi-species runs (⑧ Report)

Tab: ⑧ Report · Code: `tabs/tab_report.py` (UI and HTML report), `odmap.py` (ODMAP protocol), `batch_species.py` (multi-species batch run), `map_widget.py` and `study_area_figure.py` (publication maps)

11.1 What this tab does

⑧ Report gathers all results of the session into a single HTML file; it also partly fills in the ODMAP reporting protocol and repeats a completed workflow for the other species in a multi-species occurrence file. The tab is unlocked when ③ Models has finished (`tabs/tab_models.py:1852`).

- **Input:** the session state (models, ensemble, projections, range change, validation results), the `figures\*.png` plots in the output folder, the PNG maps the user has saved from the map viewer, `habitus_run_config.json` and the session log (`.log`).
- **Output:** `<species>_HABITUS_report_<YYYYMMDD_HHMMSS>.html` (all images embedded in the file), optionally `odmap_protocol.md` and `odmap_protocol.csv`, and in a batch run one subfolder per species plus `batch_summary.csv`.
- **Computation:** The report performs no new computation; it reads existing results and produces interpretive sentences and tables. The only maps it redraws are the range-change and validation maps.

11.2 Screen layout

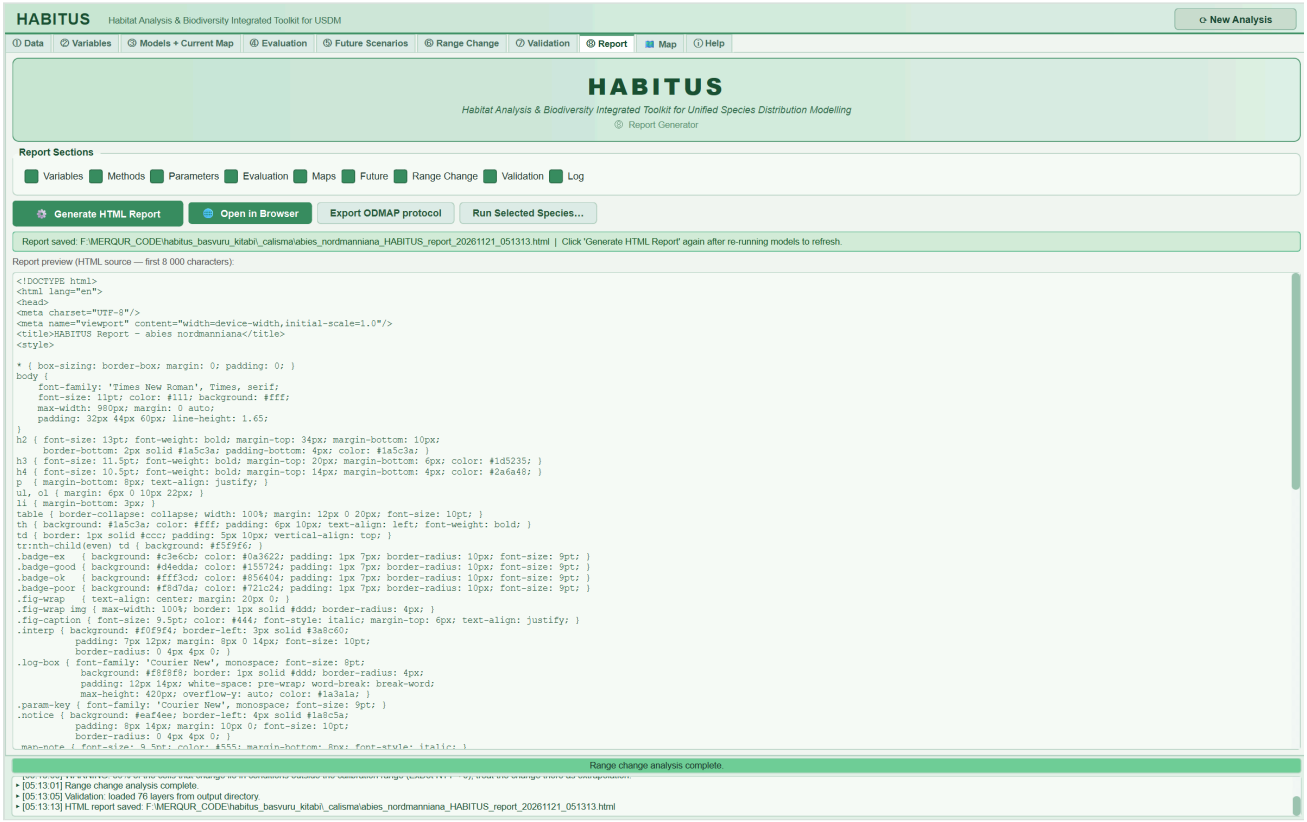


Figure 11.1. The Report tab after a report has been generated: the HABITUS header box at the top, nine section boxes in the Report Sections box, below them the Generate HTML Report, Open in Browser, Export ODMAP protocol and Run Selected Species... buttons, the path of the saved file in the green status line, and at the bottom a preview of the first 8000 characters of the HTML source.

The tab has a single pane (`tabs/tab_report.py:376-465`). From top to bottom: the header box, `Report Sections` (section selection, §11.3), four buttons (§11.4, §11.6, §11.8), the progress bar and status line, and the `Report preview (HTML source – first 8 000 characters):` preview box.

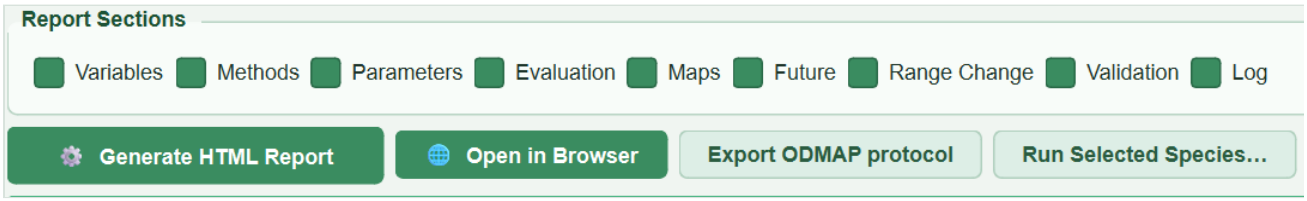


Figure 11.2. Close-up of the Report Sections box and the button row: the Variables, Methods, Parameters, Evaluation, Maps, Future, Range Change, Validation and Log boxes are all ticked by default.

11.3 Report options (Report Sections)

Name in the UI	What it does	Default	Range / options	Code
`Variables`	Section 2: variable screening and selected variables	on	on / off	`tabs/tab_report.py:400`, `:2416-2418`
`Methods`	Section 3: methods text	on	on / off	`:401`, `:2421-2423`
`Parameters`	Section 4: table of general settings and algorithm settings	on	on / off	`:402`, `:2426-2428`
`Evaluation`	Section 5: evaluation tables and plots	on	on / off	`:403`, `:2431-2433`
`Maps`	Section 6: current map PNGs saved by the user	on	on / off	`:404`, `:2436-2439`
`Future`	Section 7: map PNGs in the scenario folders	on	on / off	`:405`, `:2443-2448`
`Range Change`	Section 8: range-change tables and maps	on	on / off	`:406`, `:2464-2467`
`Validation`	Section 9: ⑦ Validation results	on	on / off	`:407`, `:2470-2473`
`Log`	Section 10: session log	on	on / off	`:408`, `:2476-2481`

- **What it does.** Each box adds or removes one section of the report. Section 1 (Study Summary), the environmental novelty section (ExDet) and the processing notices are **always** added, regardless of the boxes (`tabs/tab_report.py:2412-2413`, `:2450-2461`): these three sections cannot be switched off because they determine how the maps are to be read.
- **When and how to set it.** If a complete report is wanted as supplementary material, all are left on. If only the methods and evaluation are to be taken into the main text, the file can be made smaller by switching off `Maps`, `Future` and `Log` (embedded PNGs and a long log increase the size; the example report is 5.5 MB).
- **Effect on results.** It does not change any computation. Even if a box is ticked, the section is not written if its source is missing: for example, if ⑦ Validation was not run, section 9 does not enter the report, and if no map PNG was saved, sections 6 and 7 do not (sections 6, 7 and 9 are absent from the example report).
- **Caution.** The boxes do not select variables, or which scenario or which range-change analysis enters the report; **all** scenarios and range-change analyses in the session are written.

11.4 Generate HTML Report and Open in Browser

Name in the UI	What it does	Default	Range / options	Code
⚙ Generate HTML Report	Generates the report from scratch and saves it to the output folder	—	—	tabs/tab_report.py :417-419, :2365-2539
🌐 Open in Browser	Opens the most recently generated report in the system's default browser	disabled until a report is generated	—	:420-421, :2541-2547

On every click all outputs are re-read and the report is built from scratch; the status line says so: `Each click re-reads all outputs and re-generates from scratch.` (`tabs/tab_report.py:449-451`). The file name is `<species>_HABITUS_report_<timestamp>.html`; every generation writes a new file and older ones are not deleted (`tabs/tab_report.py:2507-2510`). If there is no output folder, a save dialog opens. When the job is finished, the `Report Ready` window shows the path and the first 8000 characters of the HTML source are written to the preview box. In the example run the report was saved as `abies_nordmanniana_HABITUS_report_20261121_051313.html`.

> **Tip:** If a model or scenario has been re-run, the report is not updated automatically; `Generate HTML Report` must be clicked again.

11.5 Sections of the HTML report

The report is in English and is written as a draft of supplementary material to be submitted to a journal; the note at the very top says so explicitly: `Results and interpretive text should be reviewed and adapted before submission to a scientific journal.` (`tabs/tab_report.py:2405-2409`). The sections and their contents are given below.

11.5.1 Header and 1. Study Summary

The header contains the species name (in italics) and the time of generation. Section 1 is a table: target species, number of records after filtering (number of presences in the first pseudo-absence set), number of variables, algorithms, total number of models, output folder and date of generation (`tabs/tab_report.py:703-725`). In the example run: 143 records, 8 variables, 4 algorithms.

11.5.2 2. Selected Variables and Multicollinearity Assessment

The screening method in ② Variables (VIF and |r| thresholds, correlation method) is described in a paragraph; if any of the selected variables exceeds the VIF threshold, they are reported by name. `2.1 Variable Screening Table` lists, for each candidate variable, the VIF, the univariate AUC, and VIF and correlation eligibility; `2.2 Final Selected Variables` lists the selected variables. The correlation heat map in the `figures\` folder (that of the selected variables, if it exists) and the VIF bar chart are embedded (`tabs/tab_report.py:729-878`).

11.5.3 3. Methods

The methods section builds sentences from the settings actually used in the session (`tabs/tab_report.py:879-1205`):

- `3.1 Occurrence data`: filtering steps and the number of records remaining.
- `3.2 Environmental predictors`: the selected variables.
- `3.3 Absence and background data`: absence source (pseudo-absence, user absences or both), number and size of sets; accessible area (buffer/hull method and kilometre value, or user polygon and how many of its units were selected; percentage of the extent); thinning distance, its source and the records remaining; same-cell filtering.
- `3.4 Modelling algorithms`: algorithm descriptions and references; cross-validation design (spatial block, environmental block, other designs or random) and a caution on how the design affects the scores; mean spatial sorting bias (SSB); number of merged folds.
- `3.5 Ensemble modelling`: two-level or single-level combining, number of replicates per algorithm, combining rule.
- `3.6 Model evaluation`: AUC, TSS, Boyce, calibration measures and a caution depending on the absence source.
- `3.7 Reproducibility`: random seed and a reference to the `habitus_run_config.json` file.

11.5.4 4. Model Parameters

The `Global Modelling Settings` table (selected variables, pseudo-absences per set, pseudo-absence strategy, number of sets, number of folds, training/test ratio, permutation-importance repeats) and the `4.1 Algorithm Settings` table (full name, reference, description and hyperparameters of each algorithm) (`tabs/tab_report.py:1209-1267`).

> **Caution:** The `Pseudo-absences per replicate` and `PA strategy` rows in this table do not show the values chosen in ① Data but the defaults 1000 and `random`; the correct values are found in the text of section 3.3 and in the `habitus_run_config.json` file (`settings.background`). The `Train / Test split` row is not used in the spatial block design; the fold arrangement is described in 3.4.

11.5.5 5. Model Evaluation Results

First, the cautions that determine how the scores are to be read (random folding, spatial sorting bias, etc.) are given **before** the table (`tabs/tab_report.py:1271-1392`). Then: `5.0 Performance and its variability` (spread across replicates), `5.1 Individual Model Scores` (AUC, TSS, Boyce, threshold, prevalence, calibration slope and interpretation for each model and fold), `5.1a Ensemble Scores` (scores of the ensemble on the cross-validation held-out folds), `5.1b Threshold-based Metrics and Omission Rates` (Kappa, MCC, sensitivity, specificity, Jaccard, Sørensen, FPB, omission rates, ΔAUC), `5.2 Algorithm Summary` (algorithm mean ± SD) and, under the `5.3` subheadings, the evaluation, ROC, omission/precision-recall/calibration, variable importance and response curve plots from the `figures\` folder. These outputs are detailed in Chapter 7.

11.5.6 6. Current Distribution

The report does not draw the current map GeoTIFFs; it embeds the **PNG files the user has saved** from the map viewer. All PNGs in the **root** of the output folder are taken, except those whose names start with `habitus_` and end with `_study_area.png` (`tabs/tab_report.py:1794-1880`). The map type is guessed from the file name (`prob`/`suitability` → suitability, `bin` → binary, `range`/`src` → range change) and a ready-made interpretive paragraph is added accordingly.

> **Caution:** The save dialog of the viewers in ⑤ and ⑥ also opens by default in the root of the output folder (`map_widget.py:612-620`). A future map saved there appears in this section under the heading "Current Distribution". Save future maps in the scenario subfolder (§11.5.7).

11.5.7 7. Future Scenarios

Every subfolder in the output folder (except `figures`, `validation` and `current`) is treated as a period; for every subfolder that contains PNGs, a `7.k <folder name>` heading is opened and the PNGs are embedded (`tabs/tab_report.py:1884-1973`). The folder name is the scenario name (the `Output subdir` in ⑤, or the scenario name if that is empty). If there are no PNGs at all, the section is not written.

11.5.8 7.1 Environmental Novelty (Extrapolation Detection)

For each projection (`current` first, then the scenarios alphabetically), NT1 < 0, NT2 > 1, the percentage of cells that are similar under both tests and the three most influential variables for range/combinatorial novelty are tabulated; if NT2 could not be computed, the reason is given (`tabs/tab_report.py:2024-2123`). If the future section (7) was not written, the heading becomes `7 Environmental Novelty …`. The method is described in Section 8.9.

7 Environmental Novelty (Extrapolation Detection)

Correlative models return a suitability value everywhere, including where the environment differs from anything seen during calibration. Novelty was quantified with the ExDet method (Mesgaran et al. 2014), which separates two kinds that are routinely conflated.

NT1 marks cells lying outside the calibration range of at least one predictor; this is what a MESS surface detects. **NT2** marks cells in which every predictor is individually within range but the *combination* of values never occurred during calibration. A univariate check reports the second group as safe by construction, and they are not rare: in the worked example of Mesgaran et al., 6,617 of the 10,785 points passing the univariate test showed a distorted correlation structure. The **most influential covariate** columns name the predictors responsible.

Projection	NT1 < 0 (outside range)	NT2 > 1 (novel combination)	Analogous	Most influential (range novelty)	Most influential (combination novelty)
current	23.4 %	0.0 %	76.6 %	bio_3 (43%), bio_15 (24%), bio_2 (13%)	—
SSP585_2090	55.4 %	0.3 %	44.3 %	bio_9 (58%), bio_3 (17%), bio_8 (10%)	bio_3 (93%), bio_8 (3%), bio_19 (2%)

Percentages are shares of all projected cells and sum to 100 for each row. Suitability values in cells flagged by either test describe an extrapolation, not a prediction, and should not be interpreted as habitat suitability without further justification. Per-cell surfaces are written beside the projections as `<scenario>_NT1.tif`, `_NT2.tif` and the two `_MIC_` rasters, whose pixel values index the predictor list carried in the raster tags.

7.2 Processing Notices

The following conditions were detected while preparing inputs or producing maps. They do not stop the analysis but affect how the results should be read.

- **Sampling bias**: Presence records are used as collected; no target-group or bias surface was supplied, so sampling bias (uneven survey effort, e.g. near roads or settlements) is not corrected. Suitability then partly describes where the species was recorded, not only where it can live. Supply a target-group CSV or a bias raster in the Data tab if effort is uneven.

Figure 11.3. The novelty part of the example HTML report: the Environmental Novelty table (NT1 < 0, NT2 > 1, share of similar cells and most influential variables for current and SSP585_2090) and the sampling-bias note in the 7.2 Processing Notices list.

11.5.9 7.2 Processing Notices

Situations that change how the maps are read but do not stop the job are itemized here; if there are none, the section is not written (`tabs/tab_report.py:1981-2022`):

- layers aligned to the training grid in projections, and warnings about uncovered cells (§8.10);
- a future map aligned in range change;
- the sampling-bias warning (if no target group or bias surface was given; this is the only note in the example report);
- the selection-bias note, if a tuning search was run;
- the reason, if the ensemble map could not be produced;
- data input and validation notes.

11.5.10 8. Range Change Analysis

For each range-change analysis: a subheading, the same rows as the table in ⑥ (extent, current and future area, percentages, cell counts, net change of the other extents, share of change under extrapolation), a note if 25% or more of the changed cells lie in extrapolated conditions, a three-level interpretation based on net change (below −10 contraction, above +10 expansion, in between stability) and a colour map redrawn from the `_SRC.tif` file (`tabs/tab_report.py:2126-2265`).

8. Range Change Analysis

Species range change was quantified by comparing binary current and future distribution maps pixel by pixel. Each cell was classified into one of four categories: **Lost** (predicted present now, absent in future), **Stable Present**, **Stable Absent**, and **Gained** (absent now, present in future).

8. SSP585_2090

Metric	Value
Counted within	**whole study boundary**
Current range (cells)	**7102**
Future range (cells)	**3826**
Lost (%)	**56.84**
Gained (%)	**10.72**
Net change (%)	**-46.13**
Stable Present (cells)	**3065**
Stable Absent (cells)	**69308**
Lost (cells)	**4037**
Gained (cells)	**761**
Net change, accessible area (%)	**-30.25**
Net change, accessible area without extrapolation (%)	**-16.57**
Changed cells in extrapolated conditions (%)	**58.6**

59% of the cells that change lie in conditions outside the calibration range of at least one predictor (ExDet NT1 < 0). The change there is extrapolation; the change within the accessible area without extrapolation is the better supported estimate.

> Under the **SSP585_2090** scenario, the analysis indicates a projected range contraction of -46.1%. Lost habitat (56.8%) represents areas where current suitable conditions are projected to disappear, posing a climate risk. Gained habitat (10.7%) represents newly suitable areas that may serve as refugia or colonisation targets.

Figure 11.4. Section 8 Range Change Analysis of the example HTML report: the SSP585_2090 table, the note stating that 59% of the changed cells lie in extrapolated conditions, and the interpretation box built from the net change (the map is not shown in this excerpt).

11.5.11 9. Validation

If a validation was carried out in ⑦ Validation: mode, reference and validation layers, number of points, overall accuracy, Cohen's κ, weighted F1 (R^2, RMSE, MAE if available), confusion matrix and at most two maps from the `validation\reclassify*.tif` files (`tabs/tab_report.py:2269-2359`). Details are in Chapter 10.

11.5.12 10. Session Log and citation

Of the `.log` files in the output folder, the one with the greatest name (the newest session) is written as is into a scrollable box (`tabs/tab_report.py:606-614`, `:2476-2481`). The recommended citation for HABITUS is at the end of the report (`tabs/tab_report.py:2484-2495`).

> **Note:** The log is written to a new file in every session. If the program was closed and reopened with the same output folder, only the log of the last session enters the report; the records of the previous session remain in their own `.log` file.

11.6 ODMAP export (Export ODMAP protocol)

Name in the UI	What it does	Default	Range / options	Code
`Export ODMAP protocol`	Writes the `odmap_protocol.md` and `odmap_protocol.csv` files to the output folder	—	—	`tabs/tab_report.py:422-430`, `:537-570`

- **What it does.** ODMAP (Overview, Data, Model, Assessment, Prediction) is the standard protocol proposed for reporting species distribution models (Zurell et al. 2020). The button builds the 79 elements of the protocol's Table 1 row by row; it fills in the elements for which a record exists in the `habitus_run_config.json` file and the session (`auto`), and marks the rest as `user_required` (`odmap.py:42-111`, `:156`, `:637-686`). No value is invented; the tooltip text also states that the report will not count as ODMAP-compliant until the user has filled in these rows.
- **When and how to set it.** Before submitting a paper; the file is written only when the button is clicked, not automatically (`tabs/tab_report.py:538-540`). The models and projections should be completed first, then exported.
- **Effect on results.** It does not affect the computation. In the example run 46 of the 79 elements were filled in automatically and 33 were left to the user. The status line and the log write the following message: `ODMAP protocol written (79 elements: 46 filled automatically, 33 to be completed by you): …`.
- **Caution.** If there is no output folder, the message `No output folder yet. Run the models first.` appears. If `habitus_run_config.json` cannot be found, all elements become `user_required` and a note to that effect is added to the message (`tabs/tab_report.py:552-564`).

Main elements filled in automatically (from the example run): model objective (algorithms, binary map rule), species name, location and grid (rows × columns, cell size, coordinate system), type of extent boundary (accessible-area buffer), response type (presence-pseudo-absence), variables, algorithms and ensemble method, summary of the steps, software and package versions, sample size and filtering rationale, pseudo-absence settings and seed, training/validation partitioning (including block size), variable pre-selection, algorithm settings, setting for prediction beyond the sampled range (clamping), variable importance, ensemble weights, how spatial autocorrelation was addressed, threshold rule, performance statistics, prediction unit, uncertainty maps, transfer scenarios and novelty quantification (ExDet; MESS and its mask if selected).

Elements left to the user: authors, contact, title, DOI; study objective; temporal extent and resolution; observation type; hypotheses and assumptions; rationale for model complexity; code and data availability; taxonomic reference system, ecological level, data sources, sampling design; independent test data; sources, temporal and spatial resolutions and measurement errors of the variables and transfer layers; model coefficients; temporal autocorrelation and nested data; response curves and the outcome of expert judgement; input data and parameter uncertainty. The multicollinearity element is filled in automatically when the thresholds in ② Variables can be read from the session (`odmap.py:487-495`).

11.7 Publication maps

HABITUS maps are not saved as screenshots of the on-screen viewer but are redrawn at journal column width (`map_widget.py:597-668`). This saving is done with the floppy-disk icon in the toolbar of the map viewers in ③, ⑤ and ⑥; the study-area map is saved with the 💾 `Save study-area figure…` button in the 🗺 Map tab (`tabs/tab_map.py:124-130`, `study_area_figure.py:293-313`). Both open the same `Figure for publication` dialog (`map_widget.py:180-238`).

Name in the UI	What it does	Default	Range / options	Code
`Width:`	Printed width of the figure	`Double column – 174 mm`	`Single column – 85 mm`, `1.5 column – 120 mm`, `Double column – 174 mm`	`map_widget.py:112-116`, `:306`
`Resolution:`	Resolution in raster formats	`600 dpi`	`300 dpi`, `600 dpi`, `1200 dpi`	`map_widget.py:117`, `:197-203`
`Title:`	Title text (study-area map only)	species name	free text	`map_widget.py:209-212`
`Include the title above the map`	Writes the title above the map	on (result maps), off (study area)	on / off	`map_widget.py:204-206`, `:306`, `tabs/tab_map.py:139`
`Italic title (scientific name)`	Writes the title in italics (study area)	on if a species name exists	on / off	`tabs/tab_map.py:289`
`Locator map in the upper-left corner`	Locator inset map (study area)	on	on / off	`tabs/tab_map.py:290`
file type	Format in the save dialog	PNG	PNG, TIFF (LZW), PDF, SVG, EPS, JPEG	`map_widget.py:156-177`

The print layout is drawn with Arial 7–9 pt, a white background and thin lines; fonts are embedded in PDF/SVG/EPS (`map_widget.py:126-131`, `:149-153`). Resolution is not used in vector formats (PDF, SVG, EPS). The height of the map is computed from the geographic aspect ratio (with latitude cosine correction) and limited to at most 9 inches (`map_widget.py:652-662`). The study-area map draws the records, the boundary, a scale bar and a north arrow over the Natural Earth country boundaries embedded in the package (`study_area_figure.py:1-10`). The viewer's other controls are described in Chapter 12.

> **Tip:** Sections 6 and 7 of the report embed only PNG files. If PDF/TIFF is to be saved for a journal, save the same map once more as PNG (the current map to the root of the output folder, the future map to the scenario subfolder). The report text states "300 DPI" for these maps; the default of the save dialog is 600 dpi.

11.8 Multi-species batch run (Run Selected Species…)

Name in the UI	What it does	Default	Range / options	Code
`Run Selected Species…`	Repeats the completed workflow with the same settings for the other species ticked in ① Data	—	confirmation dialog	`tabs/tab_report.py:431-438`, `:469-512`, `batch_species.py`

- **What it does.** In an occurrence file containing several species, after the workflow has been completed manually for one species, the **other** species ticked in ① Data are processed in turn. The steps repeated for each species (`batch_species.py:145-159`): 1. ① Data: the active species is changed and data preparation is run with the same settings; 2. ② Variables: the variables selected for the first species (and the categorical selection) are assigned unchanged; a warning on the number of variables relative to this species' number of records is written to the log (`batch_species.py:201-224`); 3. ③ Models: models are

fitted using the tab's settings as they are (including the current map); 4. ⑤ each scenario: with the same raster list, algorithms, subfolder, MESS, mask and clamping choice (`batch_species.py:226-239`); 5. ⑥ each range-change analysis: the keys selected from the list (e.g. `current :: EMwmean_bin`) are translated to this species' files, with the same analysis name and extent (`batch_species.py:241-259`); 6. ⑧ HTML report.

The output folder per species has the form `<output folder>\<species>` (`tabs/tab_data.py:1115-1124`). - **When and how to set it.** First the workflow is carried out completely and carefully for one species; every analysis to be reported in ⑤ and ⑥ must have been run once, because the batch run repeats only what has been run in the session. The confirmation dialog lists the species to be repeated, the variables, and the number of scenarios and range-change analyses (`tabs/tab_report.py:489-499`). - **Effect on results.** Each species gets the result it would have got if it had been done manually; the program does not use a separate pipeline but drives the tabs themselves (`batch_species.py:12-15`). Variable selection, however, is not redone per species: the first species' selection is applied to all species. - **Caution.** All message windows that open during the run are answered automatically (`Yes` to question dialogs) and written to the log with the prefix `[batch, …]` (`batch_species.py:80-111`). If a step fails, the remaining steps for that species are skipped and the run moves on to the next species. When the job is finished, `<output folder>\batch_summary.csv` is written: `species`, `status` (`done` / `failed`), `failed_step`, `error` (the user-facing last line of the error), `output_dir`, `minutes` (`batch_species.py:263-302`). If this file is open in another program it cannot be written, and a note is added to the log.

Not repeated in a batch run: the tuning search step, ⑦ Validation, ODMAP export and the saving of map PNGs. For this reason sections 6, 7 and 9 are absent from the reports of the species in a batch run. In addition, there is no button in the UI to interrupt a batch run; the job continues until all species are finished.

Messages: if no species is ticked, `The batch run is for an occurrence file with several species: tick the species to analyse in ① Data.`; if the workflow has not been completed, `Complete the workflow for one species first (at least ① Data, ② Variables and ③ Models); …`; if the only ticked species is the one already processed, `No other species is ticked in ① Data.` (`tabs/tab_report.py:474-487`).

11.9 Outputs

- `<species>_HABITUS_report_<time>.html`: single-file report with embedded images.
- `odmap_protocol.md`, `odmap_protocol.csv`: partly filled ODMAP (CSV columns `section, subsection, element, value, source`).
- In a batch run: `<species>\` subfolders and `batch_summary.csv`.
- Saved publication maps: at the path chosen by the user.

11.10 Expert questions

Q1. Can the methods text in the report be put directly into a paper? Answer: The report itself does not recommend this: the note at its beginning says that the text should be reviewed and adapted (`tabs/tab_report.py:2405-2409`). The text is built from the actual settings in the session (e.g. accessible area, thinning, fold design and SSB, `tabs/tab_report.py:930-1092`), but the statements of principle and the citations are generic templates; the pseudo-absence rows in section 4 also show default values (§11.5.4). The numbers should be checked against `habitus_run_config.json`.

Q2. Does the ODMAP output count as "ODMAP-compliant"? Answer: No. The module uses only the elements in Table 1 of Zurell et al. (2020); the full question list in Supplementary Table A1 was not used, and

the file states this explicitly (`odmap.py:11-21`, `:661-664`). In the example run 33 elements must be filled in by the user. Completing it by comparison with the form in the ODMAP Shiny app is recommended.

Q3. Is the MESS or clamping choice visible in the report? Answer: Not in the HTML report; the novelty section tabulates only the ExDet results (`tabs/tab_report.py:2024-2123`). The choices are found in ODMAP ("Settings for extrapolation beyond sample range", "Quantification of novel environments", `odmap.py:466-508`) and in the `_projection_options.json` file in each scenario folder. These choices need to be stated in the methods part of the paper.

Q4. What is the denominator of the ExDet percentages in the report? Answer: All cells predicted in the projection (with all variables finite); the NT1 < 0, NT2 > 1 and similar percentages sum to 100 (`sdm_novelty.py:100-125`, report note `tabs/tab_report.py:2109-2118`). The NT2 percentage is also divided by all cells, not by the subset in which NT2 was computed. In the example run, for SSP585_2090: 55.4 + 0.3 + 44.3 = 100.

Q5. In a batch run, is variable selection redone for each species? Answer: No; the variables confirmed in ② Variables for the first species are assigned unchanged (`batch_species.py:201-218`). The program only writes to the log whether this species' number of records is sufficient for that many variables (`note_selected_predictors`, `batch_species.py:219-224`). If the species differ in ecology or their numbers of records differ greatly, variable selection should be done manually per species and the species run separately. In biomod2, batch work is done with a loop written by the user; the HABITUS counterpart is applying the same settings to all species.

Q6. In a batch run, if one species fails, are the others affected? Answer: No. The failed step is recorded, the remaining steps for that species are skipped and the run moves on to the next species (`batch_species.py:161-190`, `:263-274`). `batch_summary.csv` records at which step and why it failed; the user-facing last line of the error text is taken (`batch_species.py:26-51`). The full traceback remains in the log with the prefix `[batch, trace]`.

Q7. Which resolution should be chosen for publication maps? Answer: The dialog tooltip recommends 300 dpi for colour maps in general and 600 dpi or more for figures with fine text and lines (`map_widget.py:201-203`); the default is 600 dpi. In vector formats (PDF, SVG, EPS) dpi is not used; the raster layer is still embedded as an image. The column widths are the common sizes of Elsevier, Wiley and Springer (85, 120, 174 mm; `map_widget.py:111-116`); if the journal's own size is different, the nearest width should be chosen and scaled according to the journal guidelines.

Q8. Where do the map interpretation paragraphs in the report come from? Answer: They are fixed texts chosen according to the file name (`tabs/tab_report.py:1824-1865`, `:1935-1958`): for example, for a probability map, a suggestion to prioritize areas of high suitability for fieldwork; for a future map, a sentence about a northward/upward shift with warming. These sentences are not derived from the data themselves; they should be rewritten according to the findings before being taken into a paper. The range-change interpretation, by contrast, is built from the actual value of the net change (±10% thresholds) (`tabs/tab_report.py:2184-2196`).

Q9. Is the report reproducible; which files are sufficient? Answer: The report is generated from scratch from the session results on every click; after the session is closed the same report cannot be regenerated, because the models and the ExDet summaries are held in memory. For reproducibility, `habitus_run_config.json` (all settings, seed, package versions), the `_projection_options.json` files in the scenario folders and the saved HTML report should be kept together. Section 3.7 states the seed and the configuration file (`tabs/tab_report.py:1138-1151`).

Q10. How many scenarios and how many range-change analyses appear in the report? Answer: All those carried out in the session. The novelty table writes every projection in `state["novelty"]`, and section 8 every analysis in `state["range_results"]` (`tabs/tab_report.py:2044-2073`, `:2140-2207`). If there is a trial that should not enter the report, the results should be reset with `Remove All` in ⑤ and only the desired ones re-run.

11.11 Common mistakes and pitfalls

- **Not saving the maps as PNG.** The report does not draw current or future maps from the GeoTIFFs; if no PNG was saved, sections 6 and 7 do not appear at all.
- **Saving a future map to the root of the output folder.** It appears in the report under "6. Current Distribution".
- **Trusting the pseudo-absence rows in section 4.** The correct values are in section 3.3 and in the `habitus_run_config.json` file.
- **Assuming the ODMAP file is complete.** Until the `user_required` rows are filled in, the file is not an ODMAP report.
- **Starting a batch run with a workflow that has not been completed for one species.** Only the scenarios and range-change analyses run in the session are repeated; if no scenario was run in ⑤, no future map is produced in the batch run either.
- **Overlooking the variable warning in a batch run.** For species with few records, `[batch] ADVISORY:` lines appear in the log; the number of variables may be too high for that species.
- **Forgetting to regenerate the report.** A new scenario or refitted models enter the report only when `Generate HTML Report` is clicked again; the old file is not deleted, and a new time-stamped file is written.

11.12 References

Zurell, D., Franklin, J., König, C., Bouchet, P.J., Dormann, C.F., Elith, J., Fandos, G., Feng, X., Guillera-Arroita, G., Guisan, A., Lahoz-Monfort, J.J., Leitão, P.J., Park, D.S., Peterson, A.T., Rapacciuolo, G., Schmatz, D.R., Schröder, B., Serra-Diaz, J.M., Thuiller, W., Yates, K.L., Zimmermann, N.E., Merow, C., 2020. A standard protocol for reporting species distribution models. Ecography 43, 1261–1277. https://doi.org/10.1111/ecog.04960

12. Map, Help and output files (🗺 Map, ⓘ Help)

Tabs: 🗺 Map, ⓘ Help · Code: `tabs/tab_map.py`, `occurrence_map.py`, `study_area_figure.py`, `map_widget.py` (figure options), `tabs/tab_help.py`, `run_config.py`, `main_dialog.py` (session log), `main.py` (crash log)

12.1 What this chapter covers

This chapter covers the two tabs that stand outside the workflow and the topics common to the whole workflow:

- The `🗺 Map` tab: display of the inputs on an online basemap and a study area figure for publication (§12.3).
- The `ⓘ Help` tab: the built-in English guide and the update check (§12.4).
- The complete file catalogue of the output folder (§12.5), the run record `habitus_run_config.json` (§12.6), the session and crash logs (§12.7), environment variables (§12.8), troubleshooting (§12.9) and system requirements (§12.10).

Neither tab is locked; they can be used from the moment the program opens (`main_dialog.py:194-196`). They come after the numbered steps in the tab bar and provide no input to the workflow.

12.2 The place of the 🗺 Map tab

`🗺 Map` shows on a map what was given to `① Data` before any model is built: the presence records, the records excluded before modelling once `① Data` has been run, the user's own absence records, the study boundary and the calibration polygon. Its purpose is to see a misplaced record, swapped coordinates or a boundary layer that does not overlap the region before modelling (`tabs/tab_map.py:1-11`).

Each time the tab becomes visible, it rebuilds the map if the inputs have changed (`tabs/tab_map.py:141-169`). Model results are not shown here; the current and future maps are in the map viewer of the `③`, `⑤` and `⑥` tabs (Chapters 6, 8, 9).

12.3 The 🗺 Map tab

At the top of the screen are the layer checkboxes and two buttons, below them a status line, and below that the map (`tabs/tab_map.py:100-139`). The map is drawn inside a web view with the Leaflet library; Leaflet ships with the program, while the basemap tiles are downloaded from `tile.openstreetmap.org` (`occurrence_map.py:45-50`, `92-95`).

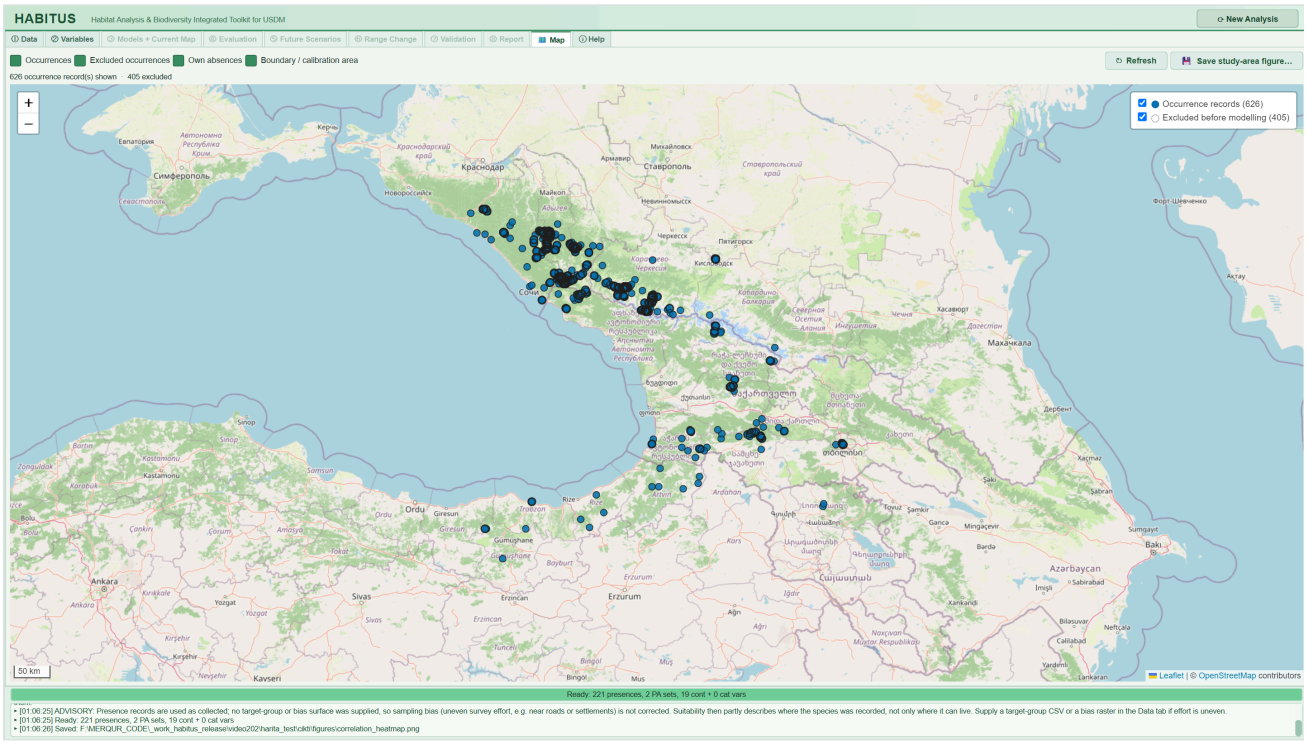


Figure 12.1. The 🗺 Map tab in the sample run after ① Data has been run: four layer checkboxes at the top, ↻ Refresh and 💾 Save study-area figure… on the right; below them the status line "626 occurrence record(s) shown · 405 excluded"; on the OpenStreetMap basemap, blue presence records and excluded records as hollow grey circles; the layer list at top right and the scale bar at bottom left.

Name in the UI	What it does	Default	Range / options	Code
`Occurrences`	Shows the presence records in the CSV; in a multi-species file each ticked species in a separate colour	on	on / off	`tabs/tab_map.py:108`, `179-214`
`Excluded occurrences`	Shows, with their reasons, the records excluded before modelling once `① Data` has been run	on	on / off	`tabs/tab_map.py:109-113`, `216-225`
`Own absences`	Shows the absence CSV given in the `Own Absences` sub-tab as black points	on	on / off	`tabs/tab_map.py:114`, `227-243`
`Boundary / calibration area`	Draws the study boundary (red) and the calibration polygon (purple, dashed)	on	on / off	`tabs/tab_map.py:115`, `245-263`
`↻ Refresh`	Rebuilds the map from the inputs	—	button	`tabs/tab_map.py:120-122`
`💾 Save study-area figure…`	Saves a location map for publication from the ticked layers	—	button; §12.3.2	`tabs/tab_map.py:123-130`, `268-311`

When one of the checkboxes changes, the map is redrawn immediately (`tabs/tab_map.py:116-117`). These checkboxes affect only the display and the saved figure; they do not change the data entering the modelling.

12.3.1 Contents of the layers

- **Presence records.** The columns are read with the names detected by `① Data`, or, if none were detected, as the first three columns. In a single-species file the layer name is "Occurrence records"; in a multi-species file only the species ticked in `① Data` are shown. Colours are taken from a colour-blind-safe list. The popup of each point gives the species name and the coordinates (`tabs/tab_map.py:179-214`, `occurrence_map.py:40-42`).
- **Excluded records.** This layer is empty until `① Data` has been run. Reasons: "duplicate coordinate", "same raster cell as another record" (when `Keep one record per raster cell` is on), "outside raster extent", "falls on NoData cell", "missing coordinate", "spatially thinned" and, for the absence file, "absence record outside extent or on NoData" (`sdm_core.py:1853`, `1875`, `1899`, `1935`, `2779`, `2949`, `2966`). In the sample run, 405 of the 626 records were excluded because another record was in the same cell (Figure 12.1).
- **Coordinate transformation.** If the rasters are in a projected coordinate system, the records are transformed to longitude-latitude for display on the map; a single record that cannot be transformed is dropped and the others are unaffected (`tabs/tab_map.py:35-71`). Before `① Data` has been run, the coordinate system of the rasters is unknown; in that case, for coordinates in metres, the status line gives the warning "coordinates are not longitude/latitude in degrees and the rasters' CRS is not known yet: run ① Data first to place them correctly" (`tabs/tab_map.py:196-199`).
- **Decimation.** If a layer has more than 20,000 points, every k-th point is drawn, for display only; the number in the layer list is the number of points drawn (`tabs/tab_map.py:22-24, 313-326`).
- **Polygons.** The study boundary may be a polygon layer or a rectangle given by corner coordinates. The calibration polygon is drawn only when `Accessible area` `polygon` is selected. A geometry with more than 5000 vertices is simplified for display (`tabs/tab_map.py:74-95, 245-263`).

The status line reports the number of records shown and excluded, the records without valid coordinates and the layers that could not be drawn. For a CSV that cannot be read, the message "The map could not be built: …" appears instead of the map (`tabs/tab_map.py:154-169`).

> **Note:** The basemap tiles require a network connection. Without a connection the points and polygons are still drawn and the basemap remains blank. The program identifies itself to the OpenStreetMap tile server as "HABITUS/<version>" (`occurrence_map.py:7-11, 53-58`). If the web view component (PyQt6-WebEngine) is not installed, the tab shows the message "The map needs PyQt6-WebEngine, which is not installed (pip install PyQt6-WebEngine)." (`occurrence_map.py:138-145`).

12.3.2 `Save study-area figure…` (study area figure)

This button draws the ticked layers of the map, independently of the web view, as a location map that can be placed in a paper (`study_area_figure.py:1-10`). The OpenStreetMap tiles on screen are at screen resolution and carry their own licence terms; the saved figure instead uses the Natural Earth country boundaries (public domain) shipped with the program and needs no network.

Pressing the button first opens the "Figure for publication" options window, then the save dialog (`tabs/tab_map.py:268-311`).

Name in the UI	What it does	Default	Range / options	Code
`Width:`	Width of the figure (journal column widths)	`Double column — 174 mm`	`Single column — 85 mm`, `1.5 column — 120 mm`, `Double column — 174 mm`	`map_widget.py:112-116`, `193-196`; `tabs/tab_map.py:139`
`Resolution:`	Resolution for raster formats (PNG, TIFF, JPEG)	`600 dpi`	300, 600, 1200 dpi; not used for PDF, SVG, EPS	`map_widget.py:117`, `197-203`
`Title:`	Title text	names of the ticked species, otherwise the species name	free text	`tabs/tab_map.py:283-288`, `map_widget.py:209-212`
`Include the title above the map`	Writes the title above the map	off	on / off	`tabs/tab_map.py:139`, `map_widget.py:204-206`
`Italic title (scientific name)`	Writes the title in italics	on if there is a species name	on / off	`tabs/tab_map.py:289`
`Locator map in the upper-left corner`	Adds a small locator map showing the study extent in the upper-left corner	on	on / off	`tabs/tab_map.py:290`, `study_area_figure.py:265-290`
File type in the save dialog	The format is determined from the file extension	PNG	PNG, TIFF (LZW-compressed), PDF, SVG, EPS (vector), JPEG (quality 95)	`map_widget.py:156-177`

The suggested file name is `<species>_study_area.png` in the output folder (`tabs/tab_map.py:294-300`). The window remembers the last choices during the session (`tabs/tab_map.py:293`).

The figure consists of the following elements (`study_area_figure.py:190-262`, `293-313`):

- **Extent.** An 8% margin around everything drawn; if the width or height is less than 0.5 degrees, it is widened to 0.5 degrees.
- **Background.** Natural Earth countries; boundaries at 1:10 million scale if the long side of the extent is less than 20 degrees, otherwise at 1:50 million. Sea is light blue, land light grey.
- **Layers.** Boundary and calibration polygons, presence records, excluded records (hollow) and own absences; in the legend the name of each layer and "(n = <number of records>)".
- **Map elements.** Axes labelled in degrees, dashed graticule, scale bar, north arrow; aspect-ratio correction by the cosine of the mid-latitude.
- **Locator map.** An area 6 times the study extent (3 times if the extent exceeds 30 degrees); the extent is shown with a red frame.
- **180° meridian.** If the data lie on both sides of the 180th meridian and are closer to each other across that line, 360° is added to western longitudes; the region is drawn in one piece.
- **Text.** Arial (otherwise Helvetica, Liberation Sans, DejaVu Sans) 7-9 pt, white background; in PDF, SVG and EPS the font is embedded (`map_widget.py:149-153`, `219`).

- **What it does.** Produces the "study area" figure of a paper without leaving the program, at column width and true resolution.
- **When and how to set it.** If the journal wants a single-column figure, choose 85 mm; for a full page, 174 mm. Because the map contains fine text and lines, 600 dpi is sufficient for most journals; if a vector format (PDF, EPS) is required, the resolution is not used. Because most journals want the title in the figure caption, `Include the title above the map` is off by default.
- **Effect on results.** It does not affect the model result. The figure draws the layers ticked in the Map tab; if a layer's checkbox is off, it is absent from the figure too.
- **Caution.** If there is nothing to draw, the message "Nothing to draw yet: load records or a study boundary in ① Data." appears. If the coordinates are not in degrees, the figure is refused with the message "The coordinates are not longitude/latitude in degrees … Run (1) Data first so that projected coordinates are converted, or check that longitude and latitude are not swapped." (`study_area_figure.py:64-69`, `tabs/tab_map.py:278-282`). For decimated layers (more than 20,000 points) the figure also draws the decimated points. Files whose names end in `_study_area.png` are not taken into the map section of ⑧ `Report` (`tabs/tab_report.py:1802-1803`).

12.4 The ⓘ Help tab

`ⓘ Help` is a user guide embedded in the program, in English only (`tabs/tab_help.py:1000-1017`). The top bar holds the update buttons, with twelve page tabs below it (`tabs/tab_help.py:1027-1101`).

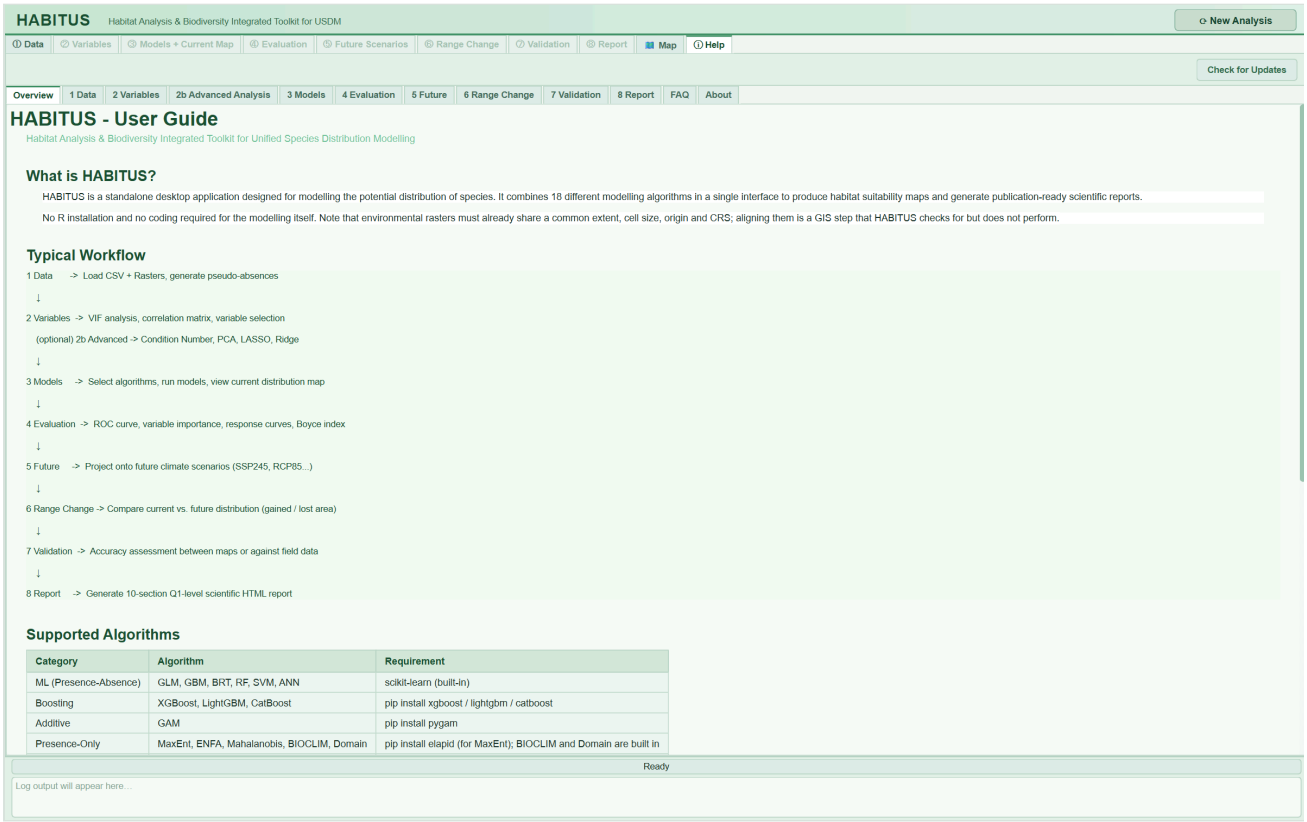


Figure 12.2. The ⓘ Help tab, Overview page: the Check for Updates button at top right, the twelve page tabs below it; on the page, "What is HABITUS?", the eight-step "Typical Workflow" and the "Supported Algorithms" table.

Page	Contents
`Overview`	Description of the program, the eight-step workflow, the algorithm table and the required libraries, quick start
`1 Data`	Presence CSV, rasters, background and pseudo-absence settings, output folder
`2 Variables`	VIF and correlation, colour codes, confirmation
`2b Advanced Analysis`	Condition number, PCA, LASSO, Ridge and how to read them together
`3 Models`	Algorithm selection, minimum-record recommendations, training settings, rationale of the cross-validation design, two-level ensemble, performance metrics, current map
`4 Evaluation`	Sub-tabs, ROC, variable importance, response curves, threshold methods
`5 Future`	Scenario name, algorithm selection, adding future rasters
`6 Range Change`	Result codes (-2, 0, 1, 2), file selection, reading the result table
`7 Validation`	Map selection, reclassification, sampling, metrics, export, interpreting Kappa
`8 Report`	Report sections, generation and export, ODMAP protocol, map quality
`FAQ`	Common errors, performance tips, output files table
`About`	Version, developer, citation, licence (GNU GPL version 3), core dependencies, update status

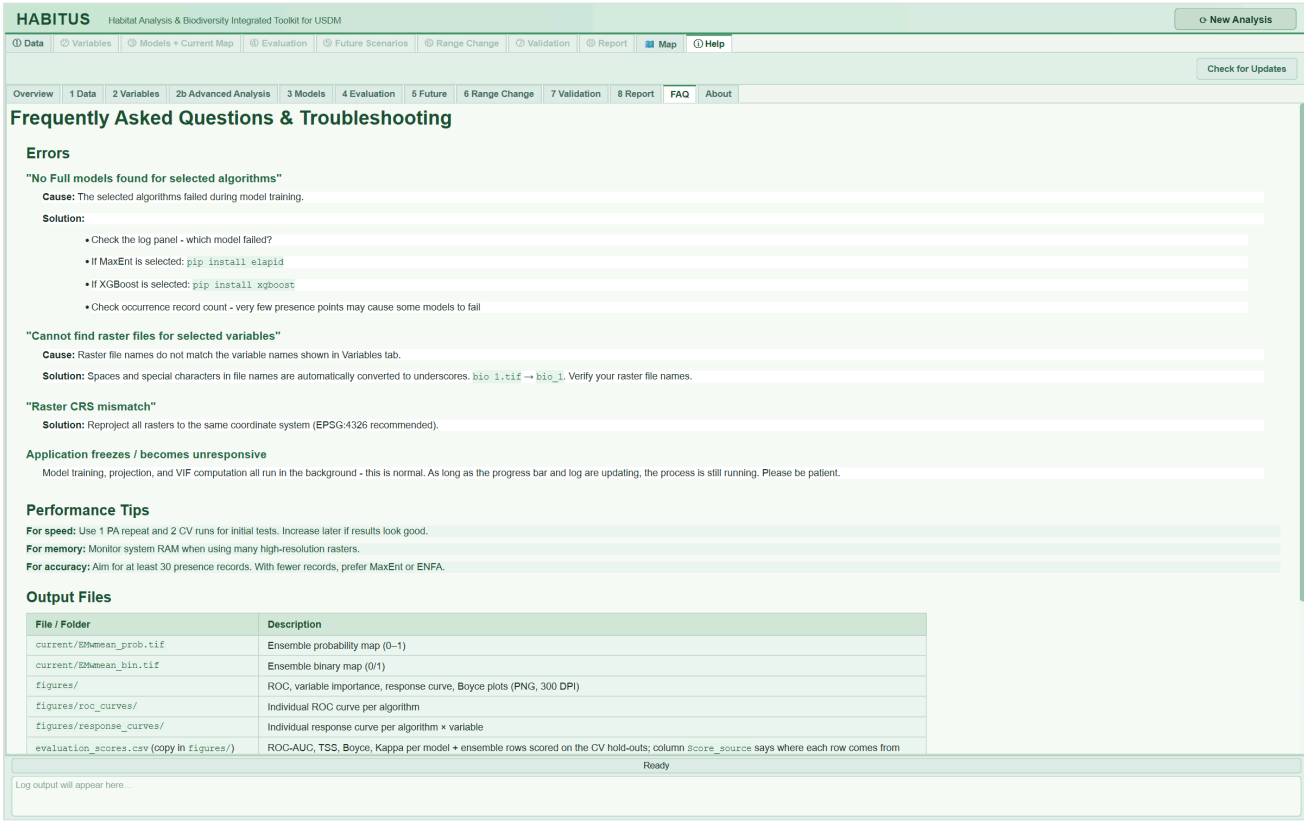


Figure 12.3. ⓘ Help > FAQ page: common error messages and their solutions under the "Errors" heading, "Performance Tips" and the beginning of the "Output Files" table.

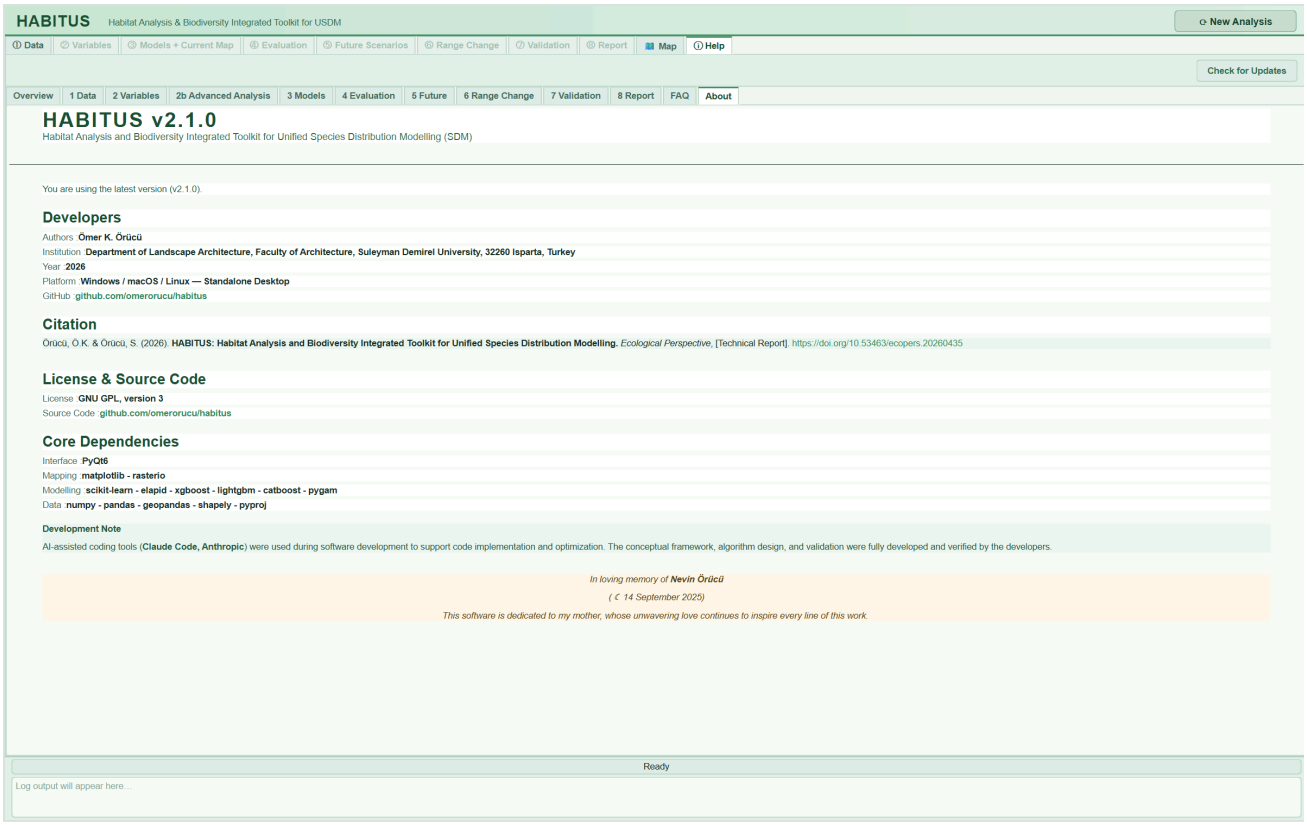


Figure 12.4. ⓘ Help > About page: the heading "HABITUS v2.1.0", the result of the update check ("You are using the latest version"), developer, citation and DOI, licence, core dependencies.

Name in the UI	What it does	Default	Range / options	Code
`Check for Updates`	Manually asks whether there is a new version on the release page; the result appears on the `About` page	—	button	`tabs/tab_help.py:1052-1059`, `1142-1152`
`Download Update`	Appears when a new version is found; downloads the installer for this computer to the Downloads folder	hidden	button	`tabs/tab_help.py:1062-1071`, `1154-1248`

The program performs the update check automatically, in the background, at start-up; it does not do so if the environment variable `HABITUS_NO_UPDATE_CHECK` is defined (`main_dialog.py:130-143`). The query goes to the GitHub release page and is abandoned if no response arrives within 10 seconds (`updater.py:88-94`). The `About` page shows the result of the check in a banner: "Checking for updates...", "You are using the latest version (v...)", "Update Available - v..." or "Update check failed: ..." (`tabs/tab_help.py:1103-1140`). If a new version is found, a line is also written to the log panel (`main_dialog.py:145-148`).

`Download Update` downloads the installer but does not install it: "Close HABITUS and run it to install the update. Your settings and results are not touched." (`tabs/tab_help.py:1212-1218`). If the download fails, the address of the release page is given.

> **Note:** The Help pages are a short reference guide. For exact details (default values, file names) this book should be taken as authoritative; for example, the actual names of the output files are in §12.5.

12.5 File catalogue of the output folder

All outputs go to the `① Data` > `Output Directory` folder (or to the folder entered in `③ Models`) (Section 1.9). In a multi-species file each species is in its own subfolder; that subfolder carries the same tree as below, while the parent folder holds `batch_summary.csv` and (if there is a boundary) `masked_inputs/current/`.

The `<species>` prefix in file names: in projection, report and figure files, spaces in the species name are converted to underscores (`abies_nordmanniana_current_EMwmean_prob.tif`); in the range change file the species name stays as it is (`abies nordmanniana_SSP585_2090_SRC.tif`) (`sdm_core.py:7655-7656`, `tabs/tab_range.py:213-216`). If the species name is unknown, no prefix is added.

The tree below is taken from the folder of the sample run and completed from the code:

```
<output folder>/
  habitus_run_config.json              ③ run record
  evaluation_scores.csv                ③ (+ evaluation_scores.<time>.bak.csv)
  .habitus_state.json                  summary for the CSV backup check
  occurrence_train.csv                 ③ training presences
  occurrence_test.csv                  ③ test presences
  occurrence_all_splits.csv            ③ both together
  occurrences_used.csv                 ① multi-species file only
  accessible_area.tif                  ① accessible area
  background_maxent.csv                ① optional
  <species>_<YYYYMMDD_HHMMSS>.log      session log
  <species>_HABITUS_report_<time>.html ⑧ report
  odmap_protocol.md / .csv             ⑧ via the ODMAP button
  <species>_<analysis name>_SRC.tif    ⑥ range change
  <species>_study_area.png             🗺 optional
  masked_inputs/current/               ① rasters clipped to the boundary
  masked_inputs/<scenario>/            ⑤ scenario rasters clipped to the boundary
  current/                             ③ current-climate maps
  <scenario>/                          ⑤ future maps
  figures/                             ② ④ figures and tables
  advanced_analysis/                   ② Advanced Analysis export
  tuning/tuning_<algorithm>.csv        ③ Tuning
  validation/                          ⑦ reclassified rasters
```

File	Which step	Contents	Code
`accessible_area.tif`	① Data	Accessible area mask from which the background is drawn; not written if `Accessible area` is `none`. Used by ⑥ Range Change	`tabs/tab_data.py:1540-1545`
`background_maxent.csv`	① Data	MaxEnt background points and their environmental values (if requested in the MaxEnt sub-tab)	`tabs/tab_data.py:1550-1559`
`occurrences_used.csv`	① Data	In a multi-species file, the records of this species only	`tabs/tab_data.py:1503`
`masked_inputs/current/`, `masked_inputs/<scenario>/`	① Data, ⑤ Future	Raster copies clipped to the study boundary	`tabs/tab_data.py:1520`, `tabs/tab_projection.py:607`

File	Which step	Contents	Code
`figures/correlation_heatmap.png`, `correlation_heatmap_selected.png`, `vif_barplot.png`	② Variables	Correlation heat map (all and selected variables), VIF plot	`tabs/tab_vif.py:1007`, `1151`, `1278`
`figures/high_correlation_pairs.csv`, `figures/variable_priority_ranking.csv`	② Variables	Highly correlated pairs; VIF and priority ranking	`tabs/tab_vif.py:1056`, `1091`
`advanced_analysis/`	② Advanced Analysis	`pca_variance.png/.csv`, `pca_loadings.csv`, `lasso_coefficient_paths.png`, `lasso_coefficients.csv`, `lasso_path.csv`, `ridge_coefficient_paths.png`, `ridge_coefficients.csv`, `ridge_path.csv`	`tabs/tab_vif_advanced.py:769-882`
`current/<species>_current_<ALGO>_PA<n>_prob.tif`, `_bin.tif`	③ Models	Probability (0-1) and binary (0/1) map for each algorithm and pseudo-absence set	`sdm_core.py:7895`, `7911`
`current/<species>_current_<ALGO>_mean_prob.tif`, `_mean_bin.tif`, `_sd.tif`	③ Models	Mean and standard deviation of the within-algorithm replicates	`sdm_core.py:7981-8005`
`current/<species>_current_EMwmean_prob.tif`, `EMca_prob.tif` (and `_bin`)	③ Models	Ensemble: weighted mean and committee averaging	`sdm_core.py:8083`, `8098`
`current/<species>_current_EMsd.tif`	③ Models	Spread among ensemble members (not thresholded)	`sdm_core.py:8056-8066`
`current/<species>_current_NT1.tif`, `NT2.tif`, `MIC_NT1.tif`, `MIC_NT2.tif`	③ Models	Extrapolation diagnostic (ExDet) and most influential variable	`sdm_core.py:7205-7215`
`current/<species>_current_projection_options.json`	③ Models	Projection options (clamp, MESS) and grid alignment summary	`sdm_core.py:7655-7662`
`evaluation_scores.csv` (a copy in `figures/`)	③ Models, ④ Evaluation	ROC-AUC, TSS, Boyce, Kappa etc. for each model and ensemble; the `Score_source` column states the source of the score	`tabs/tab_evaluation.py:1073-1107`
`occurrence_train.csv`, `occurrence_test.csv`, `occurrence_all_splits.csv`	③ Models	Training and test points with the columns `longitude`, `latitude`, `presence`, `split`	`tabs/tab_models.py:1915-1955`
`habitus_run_config.json`	③ Models	Run record (§12.6)	`tabs/tab_models.py:1640-1657`
`tuning/tuning_<algorithm>.csv`	③ Tuning	Grid search table	`tabs/tab_tuning.py:408-413`

File	Which step	Contents	Code
`figures/evaluation_model_scores.png`, `evaluation_roc_curves.png`, `evaluation_omission_pr_calibration.png/.csv`, `evaluation_variable_importance.png`, `evaluation_response_<ALGO>.png`, `evaluation_boyce_<ALGO>.png`	④ Evaluation	Evaluation figures (300 dpi) and the omission-calibration table	`tabs/tab_evaluation.py:353-847`, `923-937`
`figures/classification_thresholds.csv`	④ Evaluation	Thresholds found by each threshold method and hold-out performance	`tabs/tab_evaluation.py:1323`
`<scenario>/<species>_<scenario>_…tif`	⑤ Future	The same family of files as `current/`, with the scenario name	`sdm_core.py:7895-8138`
`<species>_<analysis name>_SRC.tif`	⑥ Range Change	Change map: -2 loss, 0 stable absence, 1 stable presence, 2 gain	`tabs/tab_range.py:213-216`
`validation/Reclassify_ref_<name>.tif`, `Reclassify_val_<name>.tif`	⑦ Validation	Reclassified reference and model maps	`tabs/tab_validation.py:728-730`, `819-830`
`validation_report_<time>.txt/.json/.html`, `validation_points_<time>.csv`	⑦ Validation	Validation report and sample points, if the user saves them	`tabs/tab_validation.py:970-1005`
`<species>_HABITUS_report_<time>.html`	⑧ Report	Single-file HTML report; the figures are embedded in the file	`tabs/tab_report.py:2507-2510`
`odmap_protocol.md`, `odmap_protocol.csv`	⑧ Report	Partially filled ODMAP protocol	`odmap.py:647-686`
`batch_summary.csv`	⑧ Report (batch)	Per species: status, failed step, error, folder, duration	`batch_species.py:280`
`<species>_<YYYYMMDD_HHMMSS>.log`	all steps	Session log (§12.7)	`main_dialog.py:612-645`

Raster format. Projection rasters are GeoTIFF, `float32`, NoData value -9999 (`sdm_core.py:7205-7215`). Probability maps are in the range 0-1. In the `current/` folder of the sample run, 37 map files were written for four algorithms (Figure 1.3).

> **Caution:** A new run overwrites the files in the output folder; files left from a previous run and not produced in the new run are not deleted (Section 1.6.2). If `evaluation_scores.csv` has been edited by hand, it is backed up as `<name>.<YYYYmmdd_HHMMSS>.bak.csv` before being overwritten; for this, a summary of the file's last write is kept in the `.habitus_state.json` file (`sdm_core.py:6752-6800`).

12.6 Run record: `habitus_run_config.json`

The run record gathers the information needed to reconstruct an analysis in a single machine-readable file (`run_config.py:1-14`). Each time `③ Models` runs, it is written to the root of the output folder after modelling has finished (`tabs/tab_models.py:1640-1657`, `run_config.py:154-163`).

12.6.1 Top-level fields

Field	Contents	Sample run	Code
`habitus_version`	Program version	"2.1.0"	`run_config.py:140`
`random_seed`	Seed of the run	42	`run_config.py:141`
`started`, `finished`	Time the record was created and written (ISO format)	both "2026-11-21T05:11:12"	`run_config.py:97-98`, `142-143`
`settings`	Settings section by section (§12.6.2)	9 sections	`run_config.py:144`
`setting_source`	Source of each setting: `default`, `user` or `recorded`	36 `default`, 2 `user`, 44 `recorded`	`run_config.py:79-92`, `105-119`
`defaults_left_unchanged`	List of settings left at their defaults: `{section, setting, value}`	36 items	`run_config.py:127-134`, `146-149`
`notes`	`ADVISORY:` notes and the cross-validation summary	4 lines	`run_config.py:121-125`
`environment`	Python version, operating system, processor architecture, versions of the scientific packages	Python 3.13.1, Windows 11, AMD64, numpy 2.4.6, scikit-learn 1.8.0, elapid 1.0.4 …	`run_config.py:61-74`

`started` and `finished` are not the start and end of the modelling job; they show the moment the record was created and written. The duration of the job is read from the time stamps in the session log.

Setting source. The code compares each recorded value with the program's default (`run_config.py:105-119`):

- `default`: the value equals the program's default;
- `user`: the setting has a default and the user changed it;
- `recorded`: there is no default to compare with; it is either an input given by the user (presence file, rasters) or a quantity found during the run (number of presences used, block size, prevalence).

The `defaults_left_unchanged` list is also printed explicitly in the report; thus no default enters a publication without the user being aware of it (`run_config.py:87-91`).

12.6.2 `settings` sections

Section	Main fields	Chapter
`data`	presence file, species, numbers of records used and excluded and the reasons, one record per cell, rasters, raster grid (shape, cell size, CRS), prevalence	2
`background`	absence source (`pseudo`, `user`, `user+pseudo`), strategy, numbers of replicates and points, MaxEnt background, sampling bias correction	3
`study_area`	accessible area method, buffer, percentage of the extent, calibration layer	3
`study_boundary`	boundary layer, number of polygons, clipped grid	2
`thinning`	whether thinning was applied, distance, records kept and dropped	3
`spatial_structure`	spatial autocorrelation range and method, study extent	5
`cross_validation`	design, requested number of folds, block size, the design actually used in each pseudo-absence set (`per_pa_set`)	5
`models`	algorithms and all their options, threshold method, Boyce formula, variables, clamp, categorical variables, class weight	5-6
`ensemble`	ensemble method, minimum score, weights, surfaces written, hold-out ensemble scores	6
`tuning_<algorithm>`	If Tuning was done, the grid, the rule and the selected point	6

From the sample run: `data.n_presences_used` 143, `data.n_points_dropped` 483, `data.prevalence` 0.125; in `notes`, "Cross-validation: PA1: spatial_block, 4 fold(s), blocks of 151.5 km [SSB=0.36] | PA2: ...".

> **Note:** The file is written when ③ `Models` finishes. The settings of later steps are not here: the projection options are in the `<species>_<scenario>_projection_options.json` file in each scenario folder; the ⑥ `Range Change` and ⑦ `Validation` settings are in the session log and the report. The program's internal constants (e.g. fixed values other than the ranges of the Tuning grid) are not written to the record; the record contains the values collected from the UI and passed to the computation engine.

12.6.3 The run record and ODMAP

ODMAP (Overview, Data, Model, Assessment, Prediction) is the standard protocol proposed for reporting species distribution models (Zurell et al., 2020). The ODMAP button in ⑧ `Report` fills in the elements of the protocol by reading the run record (`odmap.py:1-22`, `155-200`; Chapter 11):

- Elements that have a counterpart in the record (`auto`) are filled in: record counts, predictors, cross-validation, algorithms, threshold, ensemble etc.
- Elements the program cannot know (authors, hypotheses, data sources, model objective) are marked `user_required` and carry the text "Not recorded by HABITUS: to be completed by the user." (`odmap.py:38`, `153-154`).
- If the run record cannot be found, the protocol is still written, but every element becomes `user_required` (`tabs/tab_report.py:545-565`).

In short, the run record is the source of the protocol; the ODMAP file is a readable, standard-format summary of the record. The protocol does not count as an "ODMAP-compliant report" until the user completes the missing elements.

12.7 Logs

12.7.1 Session log

From the moment the output folder is known, every line of the log panel is also written to the `<species>_<YYYYMMDD_HHMMSS>.log` file in the folder (Section 1.7; `main_dialog.py:612-645`, `790-801`). The file contains:

- a header: "HABITUS Analysis Log", species, start time;
- each message in the form `[HH:MM:ss] message`; every raster read, numbers of excluded records, the autocorrelation range, the duration of each model, every file saved, warnings and `ADVISORY:` notes;
- for errors, the full error text under the heading `--- ERROR ---`;
- at closing, a "Closed" line.

The log of the sample run (`habitus_20261121_050108.log`) begins as follows: "[05:01:08] Reading raster 1 of 19: bio_1.tif" … "[05:01:09] ⚠ 405 of 626 presence records removed (405 in a cell that already has a record) — 221 retained, one per raster cell." … "[05:01:09] Spatial autocorrelation range = 151.5 km (variogram)." `® Report` appends the log file in the output folder to the last section of the report (`tabs/tab_report.py:607-616`).

12.7.2 Crash log

If the program closes before it can show an error window (a native crash or an uncaught error), the cause is written to the crash log in the user data folder (`main.py:253-290`):

Platform	Folder
Windows	`%LOCALAPPDATA%\HABITUS\logs\`
macOS	`~/Library/Application Support/HABITUS/logs/`
Linux	`$XDG_DATA_HOME/HABITUS/logs/`, or `~/.local/share/HABITUS/logs/` if the variable is not set

There is one file per day (`crash_YYYYMMDD.log`); each start-up adds a line with the version number and time. In a native crash, the Python stack of all threads at that moment is written. When reporting a crash, this file should be sent together with the session log.

12.8 Environment variables

Environment variables are defined in the operating system before the program is started. None of them is needed in ordinary use.

Variable	Effect	Default	Code
`HABITUS_UI_SCALE`	Proportionally enlarges or shrinks the UI text and controls	1.0; a value outside 0.5-3.0 or an invalid value is taken as 1.0	`ui_style.py:30-38`
`HABITUS_PROJECTION_TILE_CELLS`	Number of cells in a strip when projecting large rasters; 0 turns striping off. A raster smaller than twice this value is processed in one piece	2000000	`sdm_core.py:7512-7526`
`HABITUS_NO_UPDATE_CHECK`	Any non-empty value turns off the update check at start-up (offline environment)	undefined	`main_dialog.py:131-132`
`HABITUS_SAFE_GRAPHICS`	Linux launch scripts only: if `1`, software rendering is enabled (if windows open blank)	undefined	`scripts/make_linux_archive.sh:71-75`
`QT_QPA_PLATFORM`	Linux scripts: in a Wayland session, `xcb` (XWayland) is selected if it is not defined; the user's value is kept	set by the script	`scripts/make_linux_archive.sh:63-66`
`QTWEBENGINE_DISABLE_SANDBOX`	Linux scripts set it to `1` so that the web view of the Map tab can run from the portable folder	set by the script	`scripts/make_linux_archive.sh:67-70`
`PROJ_DATA`, `PROJ_LIB`	If they do not point to a valid coordinate database, the program sets them to its own copy at start-up	checked by the program	`main.py:110-163`
`PYTHONHASHSEED`	If not defined, set to the run seed when `Data` runs	seed	`run_config.py:49`
`HABITUS_OWNER_PID`	Internal use: the running program writes its own process number; helper processes derived from it do not open a second window. Should not be set by hand	written by the program	`proc_guard.py:46`, `86-122`

12.8.1 `HABITUS_PROJECTION_TILE_CELLS`

- **What it does.** During projection the raster is divided into row strips; each strip is predicted separately and the files are merged (`sdm_core.py:7514-7526`).
- **When and how to set it.** If an out-of-memory error occurs, the value is reduced (e.g. 500000). If memory is plentiful and speed matters, it can be increased.
- **Effect on results.** According to the code comment, striped projection is designed to produce the same files as single-piece projection (`sdm_core.py:7531-7537`); the strip size changes memory use and run time. Striped and single-piece outputs have not been compared separately for this book.
- **Caution.** A non-numeric value falls back to the default (2000000) (`sdm_core.py:7521-7524`).

12.9 Troubleshooting

The full error text is obtained with the `Copy full message` button in the error window and is found in the session log (Section 1.7). The messages below are taken from the code (their beginnings are given).

Message	Cause	Solution	Code
"Please specify an output directory."	`Output Directory` is empty	Select a folder, or first select a CSV and wait for the automatic folder	`tabs/tab_data.py:1370-1371`
"Environmental rasters are not on the same grid."	One of the rasters differs from the first raster in size, cell size, origin or CRS; the message names the differing file	Bring the rasters to a common grid in a GIS program or give a study boundary (Chapter 2)	`sdm_core.py:1735-1742`
"No continuous environmental rasters were given."	Only categorical rasters were added	Add at least one continuous raster	`sdm_core.py:1554`
"Only N valid presence point(s) remain after removing …"	Most records are outside the raster or on NoData; usually the CSV and the rasters have different coordinate systems	Check the coordinate system and the longitude-latitude order; look at the records in `🗺 Map`	`sdm_core.py:2028-2036`
"No presence record can be used: the layer(s) … have no value (NoData) …"	The named layer is NoData at all records	Check the layer or remove it	`sdm_core.py:2021`
"N of M coordinate pair(s) are not valid degrees"	Coordinates in metres or columns reversed	Correct the CSV	`sdm_inputs.py:192`
"Only one predictor is loaded …"	A single continuous raster	Add rasters in `① Data`	`tabs/tab_vif.py:369`
"… is too small for N requested points: the background was drawn from the FULL raster extent instead."	The accessible area is too narrow for the requested number of points	Enlarge the buffer or reduce the number of points	`sdm_core.py:2845`
"Calibration polygon: NONE of the N presence records lies inside the polygon(s)."	The polygon is in another region or another CRS	Check the polygon	`sdm_core.py:2657`
"WARNING: … were merged into others …"	The block design could not provide the requested number of folds	Reduce the number of folds or the block size (Chapter 5)	`sdm_core.py:5131`
"WARNING (design not applied): …"	The requested cross-validation design could not be applied and random folds were used instead; scores may be optimistic	Review the design and the distribution of records (Chapter 5)	`sdm_core.py:5139`
"No model passed the ensemble threshold …"	No model passed the threshold for entering the ensemble	Lower the minimum score or choose another metric (Chapter 6)	`tabs/tab_models.py:1732`
"No Full models found for selected algorithms."	The selected algorithms failed in training	Look at the `FAILED` lines in the log; check that the library is installed	`sdm_core.py:7859`
"Raster count mismatch! …"	The number of future rasters does not match the model's variables	Check the scenario folder and the file names (Chapter 8)	`tabs/tab_projection.py:527`

Message	Cause	Solution	Code
"Cannot find raster files for N selected variable(s)"	Scenario file names do not match the variable names	Match the file names to the names in ② `Variables`	`tabs/tab_models.py:1435`
"Complete the workflow for one species first …"	For a batch run, ①-③ must first be completed for one species	Run one species to the end (Chapter 11)	`tabs/tab_report.py:481`
"ODMAP protocol: No output folder yet. Run the models first."	No output folder	Run ③ `Models` first	`tabs/tab_report.py:554`
"… still running — close anyway?"	The window is being closed while a job is running	`Yes` cancels the job; wait for it to finish	`main_dialog.py:293-301`
"Unable to allocate …" / "MemoryError"	Insufficient memory	Clip the rasters, reduce the number of points, reduce `HABITUS_PROJECTION_TILE_CELLS`	`error_dialog.py:39-41`
"proj.db lacks DATABASE.LAYOUT.VERSION.MAJOR …"	Another installation of the coordinate library interferes	The program selects its own copy at start-up; if it persists, remove the `PROJ_DATA` and `PROJ_LIB` variables	`main.py:82-110`
"Permission denied" / "being used by another process"	The output file is open in another program	Close the file and run again	`error_dialog.py:51-53`
"The map needs PyQt6-WebEngine, which is not installed"	The web component of the Map tab is missing (running from source)	`pip install PyQt6-WebEngine`	`occurrence_map.py:140-141`
Basemap blank in the Map tab	No network connection or the tile server does not respond	Check the connection; the points and `Save study-area figure…` work offline	`occurrence_map.py:92-95`
Windows open blank on Linux	Graphics driver problem	Start with `HABITUS_SAFE_GRAPHICS=1`	`scripts/make_linux_archive.sh:71-75`
Program closed without a message	Native crash	Send the crash log (§12.7.2)	`main.py:253-290`
Window seems unresponsive	A long job is running in the background	If the progress bar and the log are updating, wait	`threads.py`

> **Tip:** To test whether an installation is sound, first run `--selftest` (Section 1.3). The report names missing algorithm libraries and coordinate database problems.

12.10 System requirements

Item	Requirement	Source
Windows	64-bit Windows 10 (version 1809, build 17763) or later; Windows 11	`habitus_setup.iss:33`, `44`
macOS	Separate Apple Silicon and Intel packages; the minimum system version is stated on the package	`habitus.spec:345-373`
Linux	64-bit; distributions with glibc 2.35 or later (the packages take some system libraries, e.g. GL and X11, from the system)	`CHANGELOG.md`, Linux packages
Running from source	Python 3.10 or later and the `requirements.txt` dependencies	`pyproject.toml:11`
Memory	There is no lower limit in the code; the memory needed depends on the size and number of rasters. A measured example: 19 global layers (2.5 minutes) occupy about 2.6 GB in memory	`CHANGELOG.md`
Network	Not required. Only the basemap tiles of the Map tab and the update check connect to the network	`occurrence_map.py:92`, `updater.py:88`
Screen	The window opens at up to 95% of the screen; on a small screen the tabs scroll. Text can be enlarged with `HABITUS_UI_SCALE`	`main_dialog.py:113-121`, `ui_style.py:30-38`

The measure that reduces the memory requirement most is clipping the rasters to the study region with ① `Data` > `Study Boundary` (Chapter 2). The run times of the algorithms are written to the session log; before a large analysis, a small trial (one pseudo-absence replicate, few algorithms) helps to estimate the run time.

12.11 Expert questions

Q1. Can another researcher reconstruct this study from the output folder alone? Answer: Largely. `habitus_run_config.json` carries the seed, the paths of the input files, all UI settings, the cross-validation design actually applied and the package versions (`run_config.py:137-152`; `tabs/tab_models.py:2056-2242`). The input files themselves are not copied into the folder (except the copies clipped with the study boundary, `masked_inputs/`); only their paths are written. No checksum proving that the content of the rasters has not changed is written to the record. The projection settings are in separate `projection_options.json` files. For full reconstruction the input data must be archived separately.

Q2. Why is `defaults_left_unchanged` a separate list? Answer: To separate which numbers in a publication were chosen deliberately and which remained as the program's defaults. The code compares each setting with its default and puts those equal to the default into this list (`run_config.py:105-134`); the report prints this list. In the sample run 36 settings are at their defaults. A reviewer can use this list to ask, for example, whether the number of pseudo-absences (1000) or the number of folds was justified.

Q3. What does the `recorded` source mean, and how does it differ from `user`? Answer: `user` is only for settings that have a default and that the user changed. Values without a default (input files, the block size computed during the run, prevalence, number of records used) are marked `recorded` (`run_config.py:79-92`, `113-118`). Thus a computed number does not look like a user choice. In the sample run 44 values are `recorded` and 2 are `user`.

Q4. Does the run record replace an ODMAP report? Answer: No. ODMAP (Zurell et al., 2020) includes elements the program cannot know, such as authorship, objective, hypotheses and data sources. `odmap.py` fills

in the elements that have a counterpart in the record as `auto`, leaves the others as `user_required` and does not invent any value (`odmap.py:14-22`, `153-154`). According to the code comment, the elements are those of Table 1 of the paper; the detailed question list in the supplementary table was not used as the basis (`odmap.py:11-14`). The user must complete the `user_required` rows.

Q5. Can the output maps be used directly in another GIS program; how are the values interpreted? Answer: Yes; they are written as GeoTIFF, `float32`, NoData -9999 and carry the coordinate system of the source rasters (`sdm_core.py:7205-7215`). The `_prob` files are suitability in the range 0-1; this is not a probability of occurrence, because prevalence is unknown in models built with pseudo-absences (Elith and Leathwick, 2009). The `_bin` files are 0/1 maps obtained with the selected threshold; the threshold method is in the `models.binary_threshold_method` field. The `EMsd` and `_sd` surfaces are not suitability but disagreement between models and are not thresholded (`sdm_core.py:8056-8066`).

Q6. Is there a difference between the map display on screen and the saved GeoTIFF? Answer: Files such as PNG and PDF produced by the viewer and by `Save study-area figure…` are images; they include a colour scale, a scale bar and aspect-ratio correction and should not be used for analysis. Numerical analysis is done with the GeoTIFFs in `current/` and the scenario folders. When the points in the Map tab exceed 20,000 they are decimated for display only (`tabs/tab_map.py:22-24`); modelling is not affected by this decimation.

Q7. When the program writes to the same folder a second time, how do I distinguish the old results? Answer: The program does not delete old files; it overwrites those with the same name. Ways to distinguish them: the `finished` time of the run record, the time-stamped name of the session log and the time-stamped name of the report. `evaluation_scores.csv` is backed up if it was changed by hand (`sdm_core.py:6764-6800`); other files are not backed up. For comparative runs, giving each run a separate folder is the only reliable way (Section 1.9).

Q8. Is the effect of the software environment (package versions) on the result recorded? Answer: Yes, the `environment` field records the Python version, operating system, architecture and the versions of numpy, pandas, scipy, scikit-learn, rasterio, matplotlib, xgboost, lightgbm, catboost, pygam, elapid and PyQt6 (`run_config.py:61-74`). Whether the same result can be expected with the same seed depends on these versions also being the same; a different scikit-learn or elapid version can give different numbers with the same seed. Because the packaged program pins the versions, repeating with the same package version is the most reliable way.

Q9. How accurate are the scale bar and aspect ratio in the study area figure? Answer: The figure does not use a projection; the longitude-latitude axes are drawn with an aspect-ratio correction by the cosine of the mid-latitude (`study_area_figure.py:229-232`). The scale bar uses the method of the map viewer (`study_area_figure.py:241`). This is an adequate approximation for small and medium-sized areas; over wide latitude ranges (e.g. continental scale) the scale bar is correct only for the mid-latitude. If an equal-area projection is required, the figure should be redrawn in a GIS program.

Q10. What information does the update check send out; can the program be used offline? Answer: The check requests only the information on the latest release on GitHub (`updater.py:88-94`); no analysis data are sent. When downloading basemap tiles, the Map tab sends the program's name and version as its identifier (`occurrence_map.py:53-58`). The program works offline; the update check can be turned off with `HABITUS_NO_UPDATE_CHECK` (`main_dialog.py:131-132`), and in the Map tab only the basemap is missing.

12.12 Common mistakes and pitfalls

- **Expecting excluded records in the Map tab.** The `Excluded occurrences` layer is empty until `① Data` has been run.
- **Seeing projected coordinates in the wrong place in the Map tab.** The coordinate system of the rasters becomes known when `① Data` is run; before that, coordinates in metres cannot be placed on the map.
- **Using a screenshot as a publication figure.** The OpenStreetMap tiles are at screen resolution and have their own licence terms; for publication use `Save study-area figure…`.
- **Treating the Help pages as the definitive reference.** Help is a short guide; for file names and defaults the tables in this book are authoritative.
- **Treating the run record as the only evidence.** The record belongs to the moment of `③ Models`; the projection, range change and validation settings are in other files (projection options, session log, report).
- **Sending only the session log after a crash.** A native crash may not be written to the session log; the crash log (§12.7.2) is a separate file.
- **Changing an environment variable while the program is open.** Variables are read at start-up; the program must be restarted for a change to take effect.

12.13 References

Elith, J., Leathwick, J.R., 2009. Species distribution models: ecological explanation and prediction across space and time. Annual Review of Ecology, Evolution, and Systematics 40, 677-697. https://doi.org/10.1146/annurev.ecolsys.110308.120159

Zurell, D., Franklin, J., König, C., Bouchet, P.J., Dormann, C.F., Elith, J., et al., 2020. A standard protocol for reporting species distribution models. Ecography 43, 1261-1277. https://doi.org/10.1111/ecog.04960