Marine Sensitivity

Project Documentation

Benjamin D. Best Timothy P. White

2026-10-14

Table of contents

Preface 3
- Interactive Apps . . . 4
- Documentation . . . 5
- How to Cite . . . 5
- Acknowledgments . . . 5

1 Introduction 6
- 1.1 Overview . . . 7

I Releases 9

2 Releases 10
- 2.1 Every release at a glance . . . 10
- 2.2 History . . . 12
- 2.3 Reading an older release . . . 19

3 v1 — Planning Areas 21
- 3.1 Conceptual framework . . . 21
  - 3.1.1 Where exposure and adaptive capacity sit . . . 22
- 3.2 Extinction-risk weighting, and why these numbers . . . 22
- 3.3 Scoring . . . 23
  - 3.3.1 Worked example . . . 23
  - 3.3.2 What ecoregional rescaling does and does not preserve . . . 24
- 3.4 Grid and projection . . . 24
- 3.5 Marine relevance in v1 . . . 25
- 3.6 Sources in v1 . . . 25
- 3.7 Primary productivity . . . 26
- 3.8 Successor releases . . . 26

4 v7c — sea turtles and the North Atlantic right whale where the record places them 27

II Science 28

5 Science 29
5.1 Conceptual Framework . . . 29
5.2 Cell Vulnerability . . . 29
5.3 Sensitivity Components . . . 30
5.4 Ecoregional Rescaling . . . 30
5.4.1 Geographic Scope . . . 31

6 Data Sources 33
6.1 Dataset overview . . . 33
6.2 The sources . . . 34
6.2.1 AquaMaps SDM (`am_0.05`) . . . 34
6.2.2 NMFS Core Area (`ca_nmfs`) . . . 35
6.2.3 NMFS Critical Habitat (`ch_nmfs`) . . . 35
6.2.4 FWS Critical Habitat (`ch_fws`) . . . 35
6.2.5 FWS Range (`rng_fws`) . . . 36
6.2.6 BirdLife Range (`bl`) . . . 36
6.2.7 IUCN Range (`rng_iucn`) . . . 36
6.3 Notes on individual sources . . . 37
6.3.1 AquaMaps vintage . . . 37
6.3.2 Where each representation lives . . . 37
6.3.3 Sea turtles: why a dedicated range dataset . . . 38
6.3.4 Range maps as masks . . . 38
6.4 Standard grid . . . 38
6.5 Primary productivity . . . 39

7 Taxonomy 40
7.1 Two taxonomic namespaces . . . 40
7.2 Taxonomic authorities . . . 40
7.3 Species categories . . . 41
7.4 Marine relevance . . . 42
7.5 From source taxa to valid species . . . 42
7.6 Data quality . . . 43
7.6.1 Duplicate resolution . . . 43
7.6.2 Synonym handling . . . 43
7.7 Implementation . . . 43

8 Extinction Risk 44
8.1 Regulatory framework . . . 44
8.1.1 Endangered Species Act (ESA) . . . 44
8.1.2 Marine Mammal Protection Act (MMPA) — assigned by taxonomy . . . 44

8.1.3 Migratory Bird Treaty Act (MBTA) — an explicit species list, not all birds . . . 45
8.1.4 IUCN Red List . . . 45
8.2 Scoring . . . 45
8.2.1 US-listed species . . . 45
8.2.2 Species with no US listing . . . 46
8.2.3 The governing score, and where it is applied . . . 47
8.2.4 When the listing is not the whole species: spatial extinction risk . . . 47
8.3 Scale . . . 48
8.4 Implementation . . . 48

9 Model Merging **49**
9.1 The rule, in one place . . . 49
9.2 Pipeline . . . 49
9.2.1 Step 1 — gather the taxon's models . . . 49
9.2.2 Step 2 — the expert range constrains AquaMaps . . . 49
9.2.3 Step 3 — two surfaces, not one . . . 51
9.2.4 Step 4 — sea turtles multiply instead of taking a maximum . . . 51
9.2.5 Step 5 — NMFS DPS-listed species: the model stays the distribution, risk varies by cell . . . 52
9.3 Where extinction risk enters . . . 53
9.4 The statutory floors . . . 53
9.5 Persisted output . . . 53
9.6 Validity after merging . . . 53
9.7 Guarding the rules . . . 54

10 Scoring **55**
10.1 Cell-level scoring . . . 55
10.1.1 Two exceptions: species whose risk varies across the range . . . 55
10.2 Metrics . . . 56
10.3 Ecoregional rescaling . . . 57
10.4 Spatial units . . . 58
10.4.1 Geographic scope . . . 58
10.5 Primary productivity . . . 60
10.6 Zone aggregation . . . 60
10.7 Visualization . . . 60
10.7.1 Flower plot . . . 60
10.7.2 Treemap . . . 60
10.7.3 Score rasters . . . 64
10.7.4 Interactive maps . . . 64
10.8 Implementation . . . 64

11 Scoring Walkthrough **65**

12 Stressors **66**
12.1 Offshore Wind Energy . . . 66
12.2 Oil & Gas . . . 66

III Software **68**

13 Software **69**
13.1 Software Overview . . . 69
13.1.1 Interactive Applications . . . 70
13.1.2 Overcoming Challenges with Large Spatial Data . . . 70
13.1.3 Github Repositories . . . 74
13.1.4 Software Components . . . 77

14 Server **78**
14.1 Setup . . . 78
14.1.1 launch instance . . . 78
14.1.2 ssh to server . . . 79
14.1.3 install docker . . . 80
14.1.4 Backup `/share` with snapshots . . . 81
14.2 Docker compose . . . 82
14.3 DNS . . . 94
14.4 Caddyfile . . . 94
14.5 Services . . . 108
14.5.1 App + data plane . . . 108
14.5.2 Admin . . . 110
14.5.3 Retired . . . 110

15 Database **111**
15.1 Table and column naming conventions . . . 111
15.2 Schema . . . 112
15.2.1 `cell` . . . 113
15.2.2 `taxon` . . . 113
15.2.3 `model` . . . 114
15.2.4 `metric` . . . 114
15.2.5 `cell_metric` . . . 114
15.2.6 `zone` . . . 115
15.2.7 `zone_cell` . . . 115
15.2.8 `zone_metric` . . . 115
15.2.9 `zone_taxon` . . . 115
15.2.10 `dataset` . . . 116
15.2.11 `taxon_model` . . . 117
15.2.12 `listing` . . . 117

15.3 Asset store . . . 117
15.3.1 The key comes from the content . . . 118
15.3.2 The catalog and the pointer table . . . 120
15.3.3 Finding a file . . . 120
15.3.4 Verify before store . . . 120
15.4 How the tables feed the serving tier . . . 121

16 Workflows **122**
16.1 The pipeline is a graph, generated from the notebooks . . . 122
16.2 Stages . . . 123
16.3 Publishing to the asset store . . . 123
16.4 Reproducibility rules . . . 124
16.5 Content-addressed checkpoints . . . 124
16.6 Guarding the science . . . 124

17 APIs **126**
17.1 Raster tiles: COGs, not SQL per request . . . 126
17.2 The `/msens` factory (retired, kept) . . . 127
17.2.1 Endpoints . . . 127
17.2.2 Caching (v7 factory only) . . . 128
17.2.3 Derived cell-id COG . . . 129
17.3 Custom plumber API . . . 129
17.4 STAC . . . 129
17.5 Retired APIs . . . 130

18 Libraries **131**
18.1 Map-tile helpers (as of 0.3.4) . . . 131

19 Apps **132**
19.1 Current Applications . . . 132
19.1.1 Atlas (beta) — static, in-browser scores, species and places . . . 132
19.1.2 Scores — mapper of aggregate sensitivity scores . . . 133
19.1.3 Species — mapper of individual species distributions . . . 133
19.2 Legacy Applications . . . 134

20 Apps User Guide **135**
20.1 Scores app . . . 135
20.1.1 1. Study Area . . . 135
20.1.2 2. Spatial Units . . . 135
20.1.3 3. Layer Selection . . . 138
20.1.4 4. Map tab . . . 138
20.1.5 5. Layers control . . . 138
20.1.6 6. Go to location . . . 141

20.1.7 7. Full screen . . . 141
20.1.8 8. Zoom in / out . . . 141
20.1.9 9. Plot of Scores tab . . . 144
20.1.10 10. Table of Species tab . . . 144
20.1.11 11. Report tab . . . 144
20.2 Species app . . . 146
20.2.1 1. Select a Species . . . 146
20.2.2 2. Mask Selection . . . 146
20.2.3 3. Layer Selector . . . 149
20.2.4 4. Species Map . . . 149
20.2.5 5. Species Info . . . 149

21 Docs **152**

22 Summary **153**
22.1 What this release delivers . . . 153
22.2 Delivered since the first release . . . 154
22.3 Future directions . . . 154

References **155**

Appendices **158**

Glossary **158**

IV Applications (Current) **160**

A Atlas (beta) — the static, in-browser scores, species and places app **161**

B Atlas **162**
B.1 Overview . . . 162
B.2 Quick start . . . 162
B.3 Getting around . . . 165
B.4 Scores lens . . . 165
B.5 Species lens . . . 166
B.6 Places . . . 169
B.7 How places are stored in the link . . . 172
B.8 Flower plot and Table . . . 173
B.9 Report . . . 175
B.10 Accessibility and performance notes . . . 180
B.11 When a service is down . . . 180
B.12 Feedback and versions . . . 181

B.13 Known limitations (as of 0.10.59) . . . 181
B.14 Generating Screenshots . . . 183
B.15 Source Code . . . 183

C Scores — mapper of aggregate sensitivity scores **184**

D Scores **185**
D.1 Overview . . . 185
D.2 Features . . . 185
D.2.1 Map Display . . . 185
D.2.2 Sensitivity Metrics . . . 188
D.2.3 Flower Plot . . . 188
D.2.4 Species Table . . . 188
D.2.5 Data Export . . . 188
D.3 Generating Screenshots . . . 189
D.4 Source Code . . . 189

E Species — mapper of individual species distributions **190**

F Species **191**
F.1 Overview . . . 191
F.2 Features . . . 191
F.2.1 Species Selection . . . 191
F.2.2 Model Display . . . 191
F.2.3 Species Info Panel . . . 193
F.2.4 Deep-Linking . . . 193
F.3 Generating Screenshots . . . 193
F.4 Source Code . . . 193

V Applications (Legacy) **194**

Areas of Interest **195**

Bird Hotspots **196**

Regional Map **197**

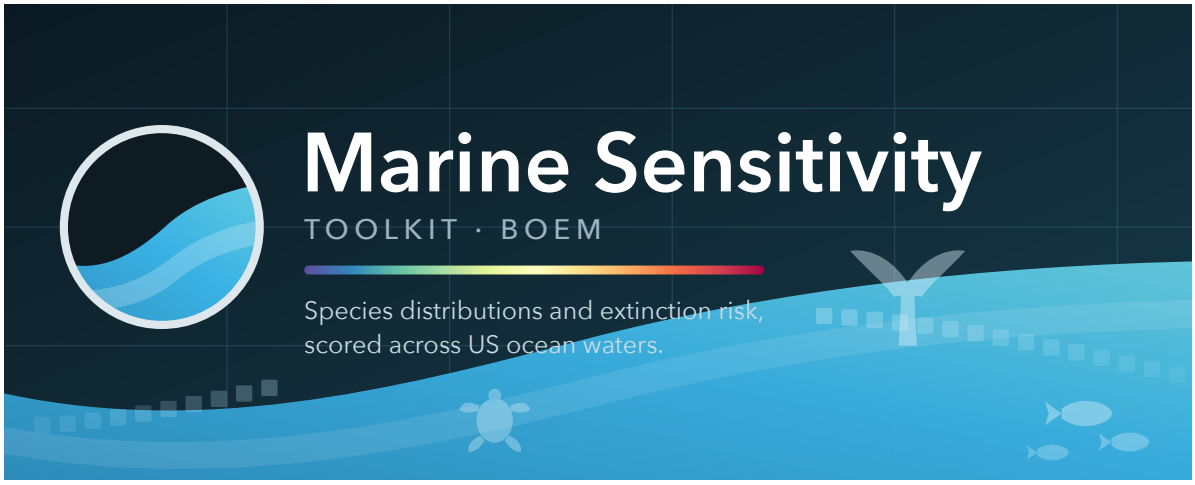
Scores Explorer **198**

Distributions, Vector **199**

Distributions, Raster **200**

Vulnerability Mapper **201**

Preface



The **Marine Sensitivity Toolkit** (MST) is a cloud-native system developed for the Bureau of Ocean Energy Management (BOEM) to assess the sensitivity of marine ecosystems to offshore energy development.

It integrates 9,795 species distribution models with extinction risk assessments across US waters (9,621 of which fall within the 20 BOEM Program Areas of the current program cycle).

> **i** What changed in v3, relative to v2
>
> *Merged models + extinction risk*
> **Datasets** — **NMFS core areas** added — designated regions of concentrated use, a targeted high-confidence habitat indicator rather than a broad-coverage model.
> **Methods** — Extinction risk is reworked into a single **governing `er_score`** per taxon, taken as the most protective across its datasets with a US national listing overriding the international IUCN category, plus statutory floors for the **MMPA** (marine mammals) and **MBTA** (migratory birds). Merging now runs for **every** taxon, masked to the IUCN range where one exists.
> **Reported units** — Scores are reported on the **20 Program Areas** only. The Planning Area geometry is still carried in the database, and a couple of non-composite metrics with it, but no composite score is computed for it — from here on, Planning Areas are history rather than a reporting unit. *(measured: 20 programarea scored on the composite*

metric.)
Scope — Taxonomy is tightened: every source name resolves to a canonical WoRMS or BirdLife identity, and non-marine and extinct taxa are dropped. Sea turtles become their own category, and non-turtle reptiles stop being scored. This roughly halves the valid-species count — a change in **definition**, not a loss of data.
Measured against v2: datasets added: `ca_nmfs`*; valid species 17,353 → 9,795.*

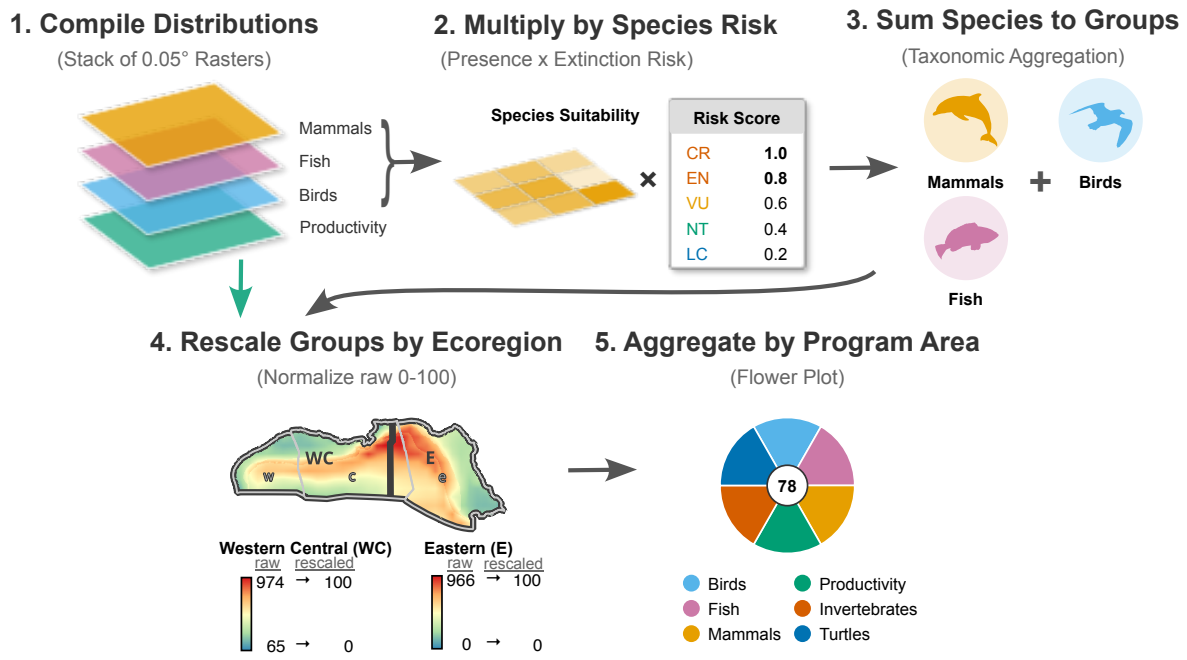


Figure 1: Overview of the MST methodology: species distribution models from the source datasets are merged, weighted by extinction risk scores (incorporating ESA, MMPA, MBTA, and IUCN protections), rescaled by ecoregion, and aggregated to management zones for visualization.

Interactive Apps

- Composite **scores** — sensitivity scores by zone, with flower-plot summaries
- Individual **species** — individual species distribution models and habitat suitability

Both render **any published release**: substitute the version in the URL path (e.g. `/v3/scores/`) to view that one; an older `?ver=v3` link still resolves, 301'd to the equivalent path.

- Scoring **walkthrough** — step through the scoring pipeline interactively, from input rasters to final scores

Documentation

This book documents the scientific methodology, software infrastructure, and interactive applications that comprise the MST. It is organized in two parts:

- **Chapter 5** — data sources, taxonomic integration, extinction risk scoring, model merging, and sensitivity scoring methodology
- **Chapter 13** — server architecture, databases, workflows, APIs, R libraries, and interactive applications

How to Cite

> Best, B.D., T.P. White (2026). Marine Sensitivity Toolkit: Project Documentation. Bureau of Ocean Energy Management, Environmental Studies Program. https://marinesensitivity.org/docs/

Acknowledgments

This work is funded by the BOEM Environmental Studies Program (BOEM-ESP). We thank the data providers whose open datasets make this work possible: AquaMaps, BirdLife International, IUCN Red List, NOAA Fisheries, and U.S. Fish & Wildlife Service.

1 Introduction

The Marine Sensitivity Toolkit (MST) is a cloud-native system developed for BOEM (Bureau of Ocean Energy Management) to assess the sensitivity of marine species to offshore energy development, whether oil & gas or wind. The Outer Continental Shelf Lands Act (OCSLA), Section 18(a)(2)(G), mandates that BOEM consider "the relative environmental sensitivity and marine productivity of different areas of the OCS" when planning the timing and location of offshore energy activities. The MST operationalizes this mandate by combining the best available species distribution data with extinction risk assessments to map areas of the ocean that are most sensitive to human activities.

The MST marks a significant advancement over prior RESA methodologies (Morandi et al. 2018). Earlier approaches often relied on aggregated data from a limited set of broad species groups and surrogate species, lacking spatially explicit information for individual organisms. As a result, previous assessments were typically coarse and areawide, frequently missing critical ecological variation and fine-scale patterns across the OCS. In contrast, the MST uses a high-resolution 0.05° grid (~4 km per cell) on a species level to capture fine-scale conservation concerns and ecological patterns across US waters.

This book documents release **v3**, which integrates **9,795 valid marine species** across **7 source datasets** on a 0.05° grid (~4 km cells) covering the US Exclusive Economic Zone study area; of these, **9,621** have modeled distributions inside the **20 BOEM Program Areas** of the current program cycle. Every figure on these pages is read from that release's own published tables, so a page never reports another version's numbers — see Releases for how the releases differ.

Species distributions are weighted by extinction risk scores that incorporate protections under the Endangered Species Act (ESA), Marine Mammal Protection Act (MMPA), and Migratory Bird Treaty Act (MBTA), as well as IUCN Red List assessments. The merged distribution of sensitivity per species is masked to expert ranges where available — species whose global range maps fall entirely outside the US EEZ are excluded, ensuring that only species with expert-confirmed presence in US waters contribute to scores.

This is a process, not a product. Information is imperfect, especially given the large expanse of US waters. Distributions and abundance of species change, modified increasingly by climate change and human activities. Knowledge on species sensitivities continues to expand with more research. Methods for both modeling and distributing this information continue to improve. In alignment with Executive Order 14303, we aim to provide a transparent and reproducible process that can be regularly updated as new data and methods become available.

1.1 Overview

The MST analytical pipeline proceeds through the following stages (Figure 1.1):

1. **Data sources** (Chapter 6): 7 datasets of species distributions — continuous suitability models, regulatory habitat designations and expert range maps
2. **Taxonomy matching** (Chapter 7): source names resolved to one canonical identity — BirdLife for birds, WoRMS for everything else — so models of the same species merge instead of double-counting it
3. **Model merging** (Chapter 9): one surface per taxon, with the expert range constraining suitability predictions and the governing extinction risk applied to the range cells
4. **Cell scoring** (Chapter 10): extinction-risk-weighted species presence, summed within each scored species category
5. **Ecoregional rescaling**: scores normalized to 0–100 within each BOEM Ecoregion, so richness alone does not decide the ranking
6. **Zone aggregation**: area-weighted means to the spatial units the release scored
7. **Visualization** (Chapter 19): flower plots, treemaps and the interactive map applications

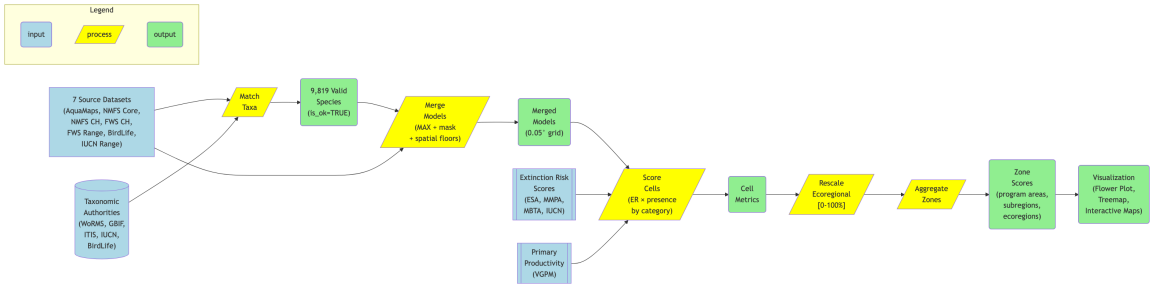


Figure 1.1: Flowchart of the MST analytical pipeline: from source datasets and taxonomic matching through model merging, scoring, and visualization.

Figure 1.2 illustrates the full methodology at a glance. On the left, species distribution data flows in from the source datasets and the taxonomic authorities. In the center, models are merged per species and weighted by extinction risk scores derived from ESA, MMPA, MBTA, and IUCN regulatory frameworks. On the right, cell-level scores are rescaled within BOEM ecoregions and aggregated to program areas for visualization in the interactive mapping applications.

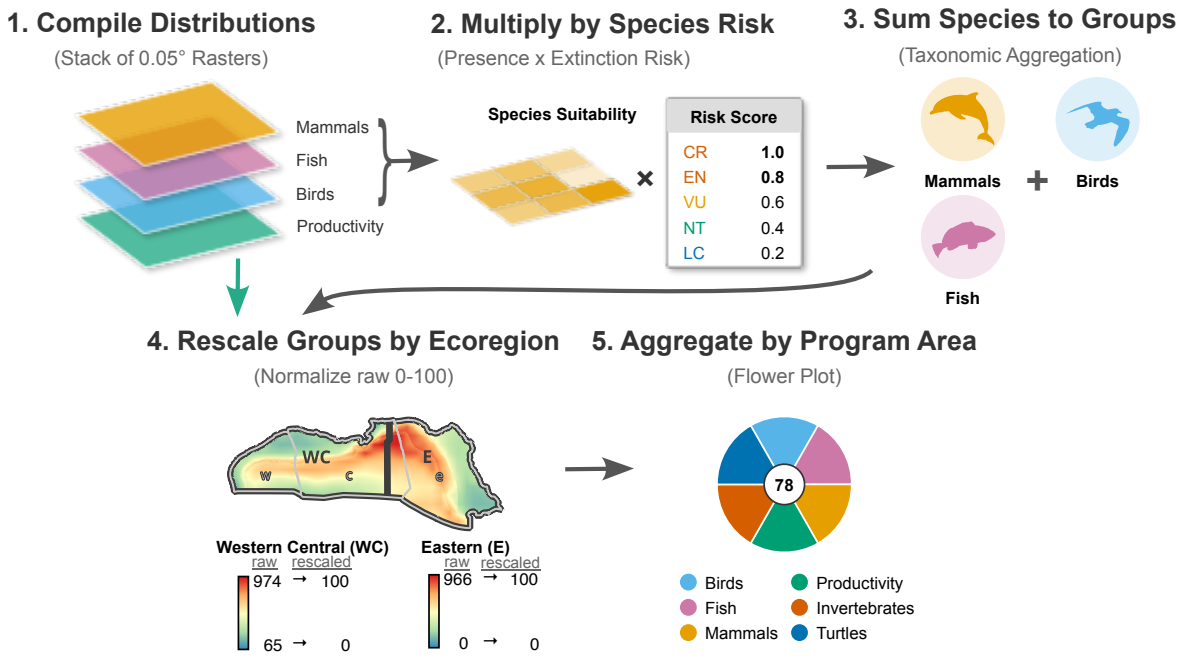


Figure 1.2: Overview of the Marine Sensitivity Toolkit methodology showing the integration of species distribution models, extinction risk scoring, ecoregional rescaling, and spatial aggregation into interactive visualizations.

Part I

Releases

2 Releases

This book is published **once per release**. The page you are reading documents **v3**, and every figure in it comes from that release's own published tables — so a v3 page reports v3's numbers, not the newest ones. Use the version switcher in the sidebar to move between builds.

This page is the exception: it reads *all* releases, because the comparison is exactly what a single-version page cannot show.

> **i** v3 — Merged models + extinction risk
>
> Released 2026-02-01. This is **retired** — superseded, but still published and still readable.

2.1 Every release at a glance

Table 2.1: Published releases. Species counts are read from each release's own `taxon` table and are **not** comparable across releases, because the definition of a valid species changed at v3 and again at v7 (see below).

Version	Released	Status	Title	Datasets	Valid species	In Program Areas	Grid
v7c	2026-10-14	pre-release	v7.2: record-based ranges for sea turtles and the North Atlantic right whale	8	16,153	9,229	`usa05`
v7b	2026-09-20	pre-release	v7.1: turtle core habitat + coverage floor	8	16,153	9,230	`usa05`
v9	2026-08-27	pre-release	AquaX supersedes AquaMaps in US waters	10	17,130	9,576	`global05`
v8	2026-07-28	pre-release	Marine Atlas	8	17,112	9,632	`global05`

Table 2.1: Published releases. Species counts are read from each release's own `taxon` table and are **not** comparable across releases, because the definition of a valid species changed at v3 and again at v7 (see below).

Version	Released	Status	Title	Datasets	Valid species	In Program Areas	Grid
v7	2026-06-12	released	Validity decoupled from Program Areas	8	16,153	9,230	`usa05`
v6	2026-04-09	retired	IUCN range outside US EEZ	8	9,424	9,226	`usa05`
v5	2026-03-24	retired	MMPA spatial floor	8	9,795	9,597	`usa05`
v4b	2026-03-19	retired	Turtle multiplicative merge	8	9,795	9,597	`usa05`
v4	2026-03-01	retired	SWOT sea turtles	8	9,795	9,597	`usa05`
v3	2026-02-01	retired	Merged models + extinction risk	7	9,795	9,621	`usa05`
v2	2026-01-01	retired	Program Areas	6	17,353	9,792	`usa05`
v1	2023-09-01	retired	Planning Areas	5	17,379	—	`usa05`

Why the species count is not a trend line

The count moves for three different reasons, only one of which is "more data":

- **v3** tightened the taxonomy — resolving every source name to a canonical identity, dropping non-marine and extinct taxa — which cut the count roughly in half.
- **v6** removed species whose expert range lies entirely outside US waters, whose apparent US presence was an artifact of environmental-envelope prediction.
- **v7** stopped requiring Program-Area overlap for validity. Nothing was added; the definition changed, and the count went up accordingly.

> Read the column as "what this release considered valid", never as a measurement of biodiversity that changed between releases.

Table 2.2: What each release published, read from its manifest.

Version	Model id	Metrics	Spatial units
v7c	`mdl_seq`	17	Ecoregions (12), Planning Areas (36), Program Areas (20), Subregions (5)
v7b	`mdl_seq`	17	Ecoregions (12), Planning Areas (36), Program Areas (20), Subregions (5)
v9	`mdl_key`	17	Ecoregions (12), Program Areas (20), Subregions (5)
v8	`mdl_key`	17	Ecoregions (12), Program Areas (20), Subregions (5)
v7	`mdl_seq`	17	Ecoregions (12), Planning Areas (36), Program Areas (20), Subregions (5)
v6	`mdl_seq`	17	Ecoregions (12), Planning Areas (36), Program Areas (20), Subregions (4)
v5	`mdl_seq`	17	Ecoregions (12), Planning Areas (36), Program Areas (20), Subregions (4)
v4b	`mdl_seq`	17	Ecoregions (12), Planning Areas (36), Program Areas (20), Subregions (4)
v4	`mdl_seq`	17	Ecoregions (12), Planning Areas (36), Program Areas (20), Subregions (4)
v3	`mdl_seq`	17	Ecoregions (12), Planning Areas (36), Program Areas (20), Subregions (4)
v2	`mdl_seq`	19	Ecoregions (12), Planning Areas (36), Program Areas (20), Subregions (4)
v1	`mdl_seq`	19	Ecoregions (12), Planning Areas (36), Subregions (4)

2.2 History

Each release is summarized under the same four headings, so two versions can be read line for line. The same summary heads that version's own landing page.

- **Datasets** — inputs added or changed, and the role each plays (contributing a value, masking a range, or carrying extinction risk spatially).
- **Methods** — the scientific rules: how models merge, how extinction risk is scored, which statutory floors apply.
- **Scope** — what counts and where: the definition of a valid species, the spatial units, the study area, which taxa are eligible.

- **Technology** — formats, identifiers, serving and applications.

⚠ Scope changes move the count without any data changing

The distinction between **Methods** and **Scope** is the one to watch. v3 tightened the taxonomy, v6 excluded ranges outside the EEZ, and v7 decoupled validity from Program Areas — each moved the headline species count substantially, and none of them added or removed a single input. Reading those jumps as growth or loss is the single easiest mistake to make with this table.

ℹ What changed in v7c, relative to v7

v7.2: record-based ranges for sea turtles and the North Atlantic right whale
Methods — A **second surgical patch of v7**, superseding v7b for review: three method changes, everything else re-executes v7's own scoring verbatim, and turning all three off reproduces v7 exactly. (1) **The turtle record rule replaces v7b's suitability threshold.** A sea-turtle species is dropped from an Alaska ecoregion where it has **never been recorded** in any observational channel (NOAA Alaska sightings, strandings and bycatch since 1963, fishery observers, OBIS, GBIF, iNaturalist, and the Washington and Gulf/Atlantic stranding networks), and kept at its v7 value everywhere else. On v7's data that drops loggerhead, green, leatherback and olive ridley from the East Bering Sea; the Gulf of Alaska, where all four have been recorded, keeps them — v7b's threshold had removed three of them there. AquaMaps suitability is no longer thresholded and the floor-of-1 range fill is back as in v7. (2) **The North Atlantic right whale leaves the Gulf of America**: it is dropped from both Gulf ecoregions, which lie outside its range — a handful of vagrant records against thousands in the Atlantic, and no critical habitat, density model or distribution-model ensemble placing it there. Both drops are rows of one evidence table, and critical habitat is never dropped. (3) **The coverage floor stays** at 5%, as in v7b; here it removes the last sliver of turtle cells from the Aleutian Arc and St. George Basin, so the two rules together leave both Program Areas without a turtle component. A down-weight in place of a drop was worked through and set aside: under ecoregion-relative rescaling a weight common to a species group cancels, and stored values are floored at 1, so only a drop reaches the score. **Productivity is untouched** — every productivity value equals v7's; the year-integration fix still lands with v10.
Reported units — Scores are reported on **20 programarea, 1 subregion** — unchanged from the previous release.
Scope — Which taxa count where, not which taxa count: no species is added or removed, and the definition of a valid species is v7's. Four sea turtles no longer occur in the East Bering Sea, and the North Atlantic right whale no longer occurs in the Gulf of America, so the Aleutian Arc and St. George Basin Program Areas lose their turtle component and the Gulf of America Program Areas' mammal component is rescaled without the

right whale. Every other Program Area scores as in v7.
Measured against v7: valid species 16,153 → 16,153.

i What changed in v7b, relative to v7

v7.1: turtle core habitat + coverage floor
Methods — A **surgical patch of v7**, not a new build: two method changes, everything else re-executes v7's own scoring verbatim, and turning both changes off reproduces v7 exactly. (1) **Sea-turtle core habitat** — AquaMaps suitability now enters the turtle merge only where it is at least 0.5 (AquaMaps' own core-habitat default); a range cell with no suitability at or above that threshold is **dropped** rather than filled at a nominal value of 1. Critical habitat still overrides by taking the higher value, so no designated cell is ever lost to the threshold. (2) **Coverage floor.** A scored component is reportable in a Program Area only where its scored cells cover at least 5% of the area; on v7's data this removes exactly one component (St. George Basin's turtles, at 1.5% coverage) rather than publishing it as a misleadingly precise near-zero. One side effect worth reading carefully: with its turtle component gone, Cook Inlet's average of its remaining components (53.2) edges past Hope Basin's (52.7) for the top Program-Area rank — a reordering caused by removing a below-average component, not by anything improving. The year-integrated productivity fix that an earlier review flagged is **not** part of this patch — productivity is unchanged from v7 here, because under this release's ecoregion-relative rescaling the year-integration barely moves scores (and moves some the wrong way); it lands with v10, whose national reference is where the correction actually counts.
Reported units — Scores are reported on **20 programarea, 1 subregion** — unchanged from the previous release.
Measured against v7: valid species 16,153 → 16,153.

i What changed in v9, relative to v8

AquaX supersedes AquaMaps in US waters
Datasets — **AquaX** (`ax`) joins as a ninth contributing input — an enhanced and revised AquaMaps framework (Reygondeau et al. 2026): 10,536 ensemble habitat-suitability models delivered on the analysis grid itself and **masked to US waters**. For the same species it **supersedes AquaMaps wherever AquaX was modeled** (10,517 species, 10,703 AquaMaps models); AquaMaps continues for the ~6,000 species AquaX did not model and everywhere outside AquaX's mask. A further 2,742 species AquaX modeled but found absent in US waters keep their AquaMaps surface in this pre-release, pending review.
Methods — Extinction risk becomes **spatial for species listed by Distinct Population Segment**: 18 species whose ESA listing applies to some populations only (humpback and Southern Resident killer whales, Cook Inlet beluga, the salmonid ESUs, both

sturgeons, …) take each listed entity's status inside its designated critical habitat, from NOAA Fisheries' authoritative service, and the baseline every unlisted species of its kind gets (the MMPA floor for a marine mammal, the IUCN category otherwise) across the rest of their range. Their merged model stays the habitat suitability; the per-cell risk is carried beside it and multiplied in at scoring, and is offered in the species app as its own layer. Before this the humpback carried the Endangered status of DPSs that breed outside US waters everywhere it occurs, a flat maximum across its entire range. Supersession is a filter on the merge input, not a per-cell blend: inside the AquaX mask a superseded species' AquaMaps cells are dropped before the merge, so an AquaX absence is an absence. Values are AquaX's 0–1000 ensemble mean ÷ 10, on AquaMaps' 0–100 scale; the model's TSS cutoff is recorded but not applied. Every other rule — range constraint, governing extinction risk, turtle multiplication — is unchanged, and a control run with supersession switched off reproduces v8's Program-Area scores exactly.

Reported units — Scores are reported on **20 programarea, 12 ecoregion, 5 subregion** — unchanged from the previous release.

Scope — Scoring extent and species definition are unchanged from v8; what changes is **which model supplies suitability** for two thirds of the scored species inside US waters.

Technology — A same-grid version bump now reuses the previous release's unchanged ingests (copy-on-write clone, verified against their content checkpoints) instead of re-running them, and every reader of the previous release resolves its schema by introspection.

Measured against v8: datasets added: `ax`, `dps_nmfs`*; valid species 17,112 → 17,130.*

i What changed in v8, relative to v7

Marine Atlas

Datasets — The same eight contributing inputs. NOAA SEFSC and NCCOS **density** models are ingested and catalogued but deliberately **excluded from the merge**, because their units (individuals per km²) have no agreed mapping onto the 0–100 suitability scale — they are flagged `is_scored = FALSE` so no published count includes them.

Methods — The expert range **constrains** AquaMaps rather than merely competing with it, and whether a taxon has a range is decided **globally** — so a species whose range lies wholly outside US waters contributes no US cells. Governing extinction risk is applied at merge time, to the range cells. A BirdLife WoRMS crosswalk de-duplicates seabirds that earlier releases split across two taxa. Species categories come from taxonomy rather than keyword matching, removing the catch-all `other` and surfacing `primary_producer`; birds are additionally culled to marine/coastal families with at least 5% of their global range over ocean.

Reported units — Scores are reported on the **20 Program Areas**, and ecoregion and subregion rollups are restored alongside them — so the release covers every spatial unit

it defines, rather than Program Areas alone. *(measured: 20 programarea, 12 ecoregion, 5 subregion scored on the composite metric.)*
Scope — Ingest becomes **global and whole-range** — no land mask, no Program-Area gate — while scoring stays over US waters. A species map now shows the species rather than the analysis extent. Non-turtle reptiles and amphibians remain excluded from scoring.
Technology — Releases are published as **partitioned Parquet on S3** with a STAC catalog and a searchable STAC API, readable directly by DuckDB including in a browser. Score surfaces become **pre-rendered COGs**, retiring the tile factory that ran SQL per request. A stable `mdl_key` replaces the auto-increment `mdl_seq`, which renumbered on every rebuild and broke saved links. One application renders **every** release from a `?ver=` parameter, so the per-version forks are retired.
Measured against v7: valid species 16,153 → 17,112; ***grid changed*** *(`usa05` → `global05`), so `cell_id` denotes a different place; public model id `mdl_seq` → `mdl_key`; spatial units changed.*

i What changed in v7, relative to v6

Validity decoupled from Program Areas
Methods — Scores are recomputed over the full set, with negligible effect: 18 of 20 Program Areas are unchanged.
Reported units — Scores are reported on the **20 Program Areas**. A single subregion also carries a composite score, which is a by-product of the study-area rollup rather than a reporting unit. *(measured: 20 programarea, 1 subregion scored on the composite metric.)*
Scope — **Validity is decoupled from Program Areas.** Earlier releases silently required a species to overlap a current Program Area to count as valid, which hid biodiversity outside the program cycle's footprint. Validity now means a taxonomically valid species with a modeled distribution anywhere in the study area, and Program-Area membership becomes a spatial query answered afterwards. The count rises sharply **without a single species being added**.
Measured against v6: valid species 9,424 → 16,153; spatial units changed.

i What changed in v6, relative to v5

IUCN range outside US EEZ
Datasets — Three missing marine-mammal IUCN ranges ingested from a more complete download.
Reported units — Scores are reported on **20 programarea** — unchanged from the previous release.
Scope — Species whose **IUCN expert range lies entirely outside the US EEZ**

are excluded. Their apparent presence in US waters was an edge-of-range artifact of environmental-envelope prediction, with near-zero suitability, so the effect on scores is negligible even though the species count drops.
Measured against v5: valid species 9,795 → 9,424.

i What changed in v5, relative to v4b

MMPA spatial floor
Methods — The **MMPA floor is corrected** for FWS-managed marine mammals — walrus, polar bear, sea otter, manatee and dugong — which had been missed because the protection flag was derived from the NMFS Species Directory alone.
Reported units — Scores are reported on **20 programarea** — unchanged from the previous release.
Measured against v4b: valid species 9,795 → 9,795.

i What changed in v4b, relative to v4

Turtle multiplicative merge
Methods — The **multiplicative turtle merge**: for sea turtles the DPS extinction-risk surface is multiplied into the suitability surface during merging, with critical habitat overriding by maximum. Risk therefore varies spatially within a species. These taxa pass `er_score = 100` through at scoring so the risk already embedded in their values is not counted twice. From here the toolkit has **two** merge strategies rather than one.
Reported units — Scores are reported on **20 programarea** — unchanged from the previous release.
Measured against v4: valid species 9,795 → 9,795.

i What changed in v4, relative to v3

SWOT sea turtles
Datasets — **SWOT sea turtle ranges** and **NOAA Distinct Population Segment** polygons added, replacing the IUCN range as the global mask for all six sea turtle species. Their role is unusual: the DPS polygons carry **ESA listing status spatially** (Endangered vs Threatened by region), rather than as one scalar per species — the first input to vary extinction risk *within* a species' range.
Reported units — Scores are reported on **20 programarea** — unchanged from the previous release.
Scope — Turtle distributions become substantially more restrictive, since the IUCN and AquaMaps ranges both extended well beyond where the species actually occur.
Measured against v3: datasets added: `rng_turtle_swot_dps`; valid species 9,795 →

9,795.

i What changed in v3, relative to v2

Merged models + extinction risk
Datasets — **NMFS core areas** added — designated regions of concentrated use, a targeted high-confidence habitat indicator rather than a broad-coverage model.
Methods — Extinction risk is reworked into a single **governing** `er_score` per taxon, taken as the most protective across its datasets with a US national listing overriding the international IUCN category, plus statutory floors for the **MMPA** (marine mammals) and **MBTA** (migratory birds). Merging now runs for **every** taxon, masked to the IUCN range where one exists.
Reported units — Scores are reported on the **20 Program Areas** only. The Planning Area geometry is still carried in the database, and a couple of non-composite metrics with it, but no composite score is computed for it — from here on, Planning Areas are history rather than a reporting unit. *(measured: 20 programarea scored on the composite metric.)*
Scope — Taxonomy is tightened: every source name resolves to a canonical WoRMS or BirdLife identity, and non-marine and extinct taxa are dropped. Sea turtles become their own category, and non-turtle reptiles stop being scored. This roughly halves the valid-species count — a change in **definition**, not a loss of data.
Measured against v2: datasets added: `ca_nmfs`*; valid species 17,353 → 9,795.*

i What changed in v2, relative to v1

Program Areas
Datasets — **IUCN Red List range maps** added, and used as the global **mask** — constraining AquaMaps predictions to the expert-assessed range rather than contributing a value of their own.
Methods — A model-merging pipeline is introduced, producing one merged surface per taxon where a taxon has more than one source.
Reported units — The **transitional** release: scores are reported on **both** the 36 Planning Areas and the new 20 Program Areas, so the two can be compared directly on the same underlying data. This is the only release that scores both. *(measured: 36 planarea, 20 programarea scored on the composite metric.)*
Scope — The **BOEM Program Areas** of the 2026 program cycle are introduced. They do not yet *replace* the Planning Areas — this release scores both, and the changeover completes at v3.
Technology — Individual model layers are published, and distributions are stored as cloud-optimized GeoTIFFs.
Measured against v1: datasets added: `rng_iucn`*; valid species 17,379 → 17,353; spatial*

units changed.

i v1 — the basis everything since builds on

Planning Areas
Datasets — **AquaMaps** global environmental-envelope models supply the distribution of nearly every species, with **BirdLife** expert ranges for birds. NMFS/FWS critical habitat and FWS ranges are present only as a small pilot. Continuous suitability is used as published; expert range polygons take a 50% presence value by convention, since an extent-of-occurrence boundary asserts presence *somewhere* within it rather than throughout.
Methods — A cell's score is the sum, over the species present, of presence × extinction-risk weight, computed separately per species category. Weights are the IUCN **equal-steps** scale used by the Red List Index and the STAR metric, rescaled so Least Concern keeps a non-zero weight — so overall biological richness stays part of the signal rather than the score being driven entirely by the few threatened species. Cell scores are then rescaled **within each BOEM Ecoregion** and aggregated by area-weighted mean.
Reported units — Scores are reported on the **36 OCS Planning Areas** and the **12 BOEM Ecoregions**. v1 predates the Program Areas entirely — its database contains none, which is why a Program Area view of v1 is empty rather than merely uninteresting. *(measured: 36 planarea, 12 ecoregion scored on the composite metric.)*
Scope — Marine relevance is decided *spatially* — any model whose distribution reaches the study area is included — with no independent test of whether the species is marine. Models are merged only for the handful of taxa that have more than one source; every other species is scored directly from its single source model.
Technology — A 0.05° geographic (WGS84) grid, chosen because every input arrives on a latitude/longitude grid: analysing in geographic coordinates resamples each input once rather than twice. Area distortion is corrected arithmetically rather than by projection.

2.3 Reading an older release

Every release remains published and readable:

- **Documentation** — this book, at `/{version}/`; the sidebar switcher moves between builds and marks any version whose docs are not built.
- **Applications** — `/v3/scores/` and `/v3/species/`, substituting any published version in the path. The per-version application forks are retired; their old URLs 301 to the equivalent path, so links printed in earlier reports — including an older `?ver=v3` deep link — still resolve to the release they always showed.

- **Data** — `marine-atlas/{version}/tables/*.parquet` on S3, readable directly by DuckDB (including in a browser) without downloading a database.

Promotion is deliberate and separate from publishing. Writing data does not make a release current: the pointer that applications follow by default is only rewritten under an explicit flag, and may only name a released — never a pre-release — version. It currently names **v7**.

3 v1 — Planning Areas

v1 was the first delivered version of the Marine Sensitivity Toolkit, reported to BOEM's Environmental Studies Program under contract 140M0123P0018 as the *Marine Sensitivity Toolkit* final report (2025).

It is documented here in its own right because several of its choices — the conceptual framing, the extinction-risk weighting and its justification, and the honest account of what ecoregional rescaling does and does not preserve — remain the foundation of every later release, and because reports and decisions cite it.

> **i** What this page is, and is not
>
> This is the **methodology** of v1. Numbers are deliberately omitted: the counts in the 2025 report are snapshots of a build that has since been superseded, and reprinting them here would set them beside computed figures elsewhere in this book as if the two were equally current. For v1's counts as published, see the Releases table, which reads them from the v1 release itself.
> The report also retains, as a matter of record, figures it identifies as incorrect. Those are not carried forward.

3.1 Conceptual framework

Vulnerability (V) is a function of exposure (E), sensitivity (S) and adaptive capacity (A):

$$V = f(E, S, A) \tag{3.1}$$

This decomposition is the standard framing in ecological risk and climate-vulnerability assessment (Intergovernmental Panel on Climate Change 2014; Halpern et al. 2007; Hare et al. 2016). The more exposed and sensitive an area is — and the less able it is to recover — the more vulnerable it is to impacts from offshore activities.

For spatial implementation, the vulnerability of a cell v_c is the sum across all species in a taxonomic group S_g of the product of the species' presence in the cell p_{sc} and a species weight w_s, the risk of that species going extinct:

$$v_c = \sum_{s=1}^{S_g} p_{sc} \cdot w_s \tag{3.2}$$

where p_{sc} is presence probability or suitability (0–1) and w_s is how at-risk that species is of going extinct (0–1, ranging from Least Concern at 0.2 to Critically Endangered at 1).

In plain terms: if a cell has many species that are both likely to be present and at high risk of extinction, it gets a higher sensitivity score. This is how places where rare or threatened species concentrate are found.

3.1.1 Where exposure and adaptive capacity sit

The implemented score resolves the **sensitivity** and **adaptive-capacity** terms jointly, and leaves **exposure** to be supplied by the decision being evaluated.

Extinction risk serves as an integrated proxy for both because the IUCN and ESA listing criteria assess population size, rate of decline, range restriction and fragmentation — the same properties that govern whether a population can absorb an added impact and recover from it.

This is a deliberate simplification, and the principal structural limitation of v1: adaptive capacity cannot be varied independently of sensitivity, and traits that distinguish recovery potential *within* a risk category — generation time, fecundity, dispersal, habitat specialization — are not represented. Separating the terms would mean adding activity-specific exposure layers (spill-trajectory probability, lease-block footprints, modeled noise fields) and letting them interact with trait-based adaptive-capacity attributes drawn from established marine vulnerability protocols (Hare et al. 2016).

3.2 Extinction-risk weighting, and why these numbers

IUCN Red List categories and ESA status codes were standardized to numeric risk weights: CR = 1.0, EN = 0.8, VU and ESA Threatened = 0.6, NT = 0.4, LC = 0.2. Where a species had several assessments the most recent was used; where statuses conflicted across regions the precautionary principle applied, taking the higher risk category.

These weights are a rescaling of the IUCN **equal-steps** weighting used for the Red List Index (Butchart et al. 2004, 2007) and for the Species Threat Abatement and Restoration (STAR) metric (Mair et al. 2021), which maps LC = 0, NT = 1, VU = 2, EN = 3, CR = 4. Dividing by five and shifting Least Concern from 0 to 0.2 yields the multipliers above.

STAR is the closest published analogue to the MST score: an additive, spatially explicit sum of species presence weighted by extinction risk.

Least Concern is deliberately given non-zero weight so that overall biological richness remains part of the sensitivity signal — consistent with the statutory language — rather than the score being driven entirely by the small fraction of species that are threatened. A Critically Endangered species still carries five times the weight of a Least Concern one.

Equal-steps weighting is the most widely used convention but not the only defensible one. Extinction-probability weightings (Butchart et al. 2007; Mooers et al. 2008) are strongly convex and would concentrate nearly all weight on CR and EN species. Because the transformation can affect rank order (Mooers et al. 2008), a sensitivity analysis comparing area rankings under equal-steps, extinction-probability and equal (richness-only) weightings is a recommended next step.

3.3 Scoring

Step 1 — cell-level scoring. For each 0.05° cell the raw score v_c is the sum of presence × extinction-risk weight across the species in a taxonomic group.

Step 2 — ecoregional rescaling. Raw cell scores are rescaled to 0–100 within each BOEM Ecoregion, using the minimum and maximum observed inside that ecoregion for the component being scored.

Step 3 — Planning Area aggregation. Rescaled cell scores are aggregated by **area-weighted** average: each cell contributes in proportion to the fraction of its area falling inside the Planning Area, so larger low-latitude cells do not outweigh smaller high-latitude ones.

3.3.1 Worked example

A simplified Planning Area with three equal-area cells in an ecoregion whose fish scores run from 65 to 974:

Cell	Raw fish score	Rescaled	Cell area (km²)	Weight
A	200	$(200-65)/(974-65)\times 100 = 15$	25	0.33
B	600	$(600-65)/(974-65)\times 100 = 59$	25	0.33
C	400	$(400-65)/(974-65)\times 100 = 37$	25	0.33

The Planning Area fish score is $15\times 0.33 + 59\times 0.33 + 37\times 0.33 = 37$ — meaning the Planning Area sits 37% of the way between the lowest and highest fish scores observed in its ecoregion.

3.3.2 What ecoregional rescaling does and does not preserve

Rescaling within ecoregions is a deliberate analytical choice with a cost that should be explicit.

Because min–max rescaling is **monotonic**, it preserves the complete ordering of cells and Planning Areas *within* an ecoregion: no within-region comparison is affected. What it removes is the difference in **absolute magnitude between** ecoregions. A score of 80 in the Gulf of America and a score of 80 in the Arctic both denote "high relative to that region", not equal absolute sensitivity, and the difference in raw species richness between those regions is by design not visible in the rescaled value.

The rationale is that the raw score is count-like — it increases with the number of species modeled in a cell — and is therefore governed by the latitudinal richness gradient. Without rescaling, nearly every subtropical Planning Area would outrank nearly every Arctic one, and the result would approximate a map of model density rather than of conservation concern. Within-region normalization is also consistent with the prior RESA methodology (Niedoroda 2014).

Two consequences follow for interpretation:

1. **Rescaled scores are not a national ranking.** Comparing areas across ecoregions compares each to its own regional context.
2. **The endpoints are observed extrema**, so a single outlier cell can set the 100 for an entire ecoregion and compress every other score in it. Robust percentile endpoints (1st/99th) are recommended in their place.

Because absolute magnitude is management-relevant information, reporting raw scores alongside the rescaled ones — together with a nationally rescaled variant and the rank correlation between the two rankings — would let the effect of the normalization on rank order be assessed directly rather than assumed.

3.4 Grid and projection

The analysis grid is **geographic (WGS84) rather than projected**, because every input dataset — AquaMaps half-degree cells, Bio-ORACLE reference layers, the VIIRS productivity grid, and IUCN/BirdLife range polygons — is distributed on a regular latitude/longitude grid. Analyzing in geographic coordinates resamples each input once rather than twice, avoiding blurred suitability surfaces and displaced range boundaries; and no single equal-area projection serves both the Gulf of America and the Aleutians well.

Area distortion is therefore corrected **arithmetically rather than by projection**: cell area is computed on the sphere, and every cell-to-area aggregation is weighted by that true area and by the cell's fractional overlap with the Planning Area — making the reported statistics

equal-area in effect. Maps are drawn in region-appropriate equal-area projections **for display only**; no analysis is performed in a display projection.

3.5 Marine relevance in v1

In v1 a **spatial inclusion rule** was applied — every model whose distribution intersected the study area was ingested — without an independent test of whether the species is marine.

This admits a small number of non-marine species with coastal range polygons; the introduced White-winged Parakeet (*Brotogeris versicolurus*) in southern Florida is one. Because such species are almost always Least Concern and occupy few cells, the effect on scores is negligible, but their presence in the species viewer is not defensible.

Subsequent versions apply an explicit marine-relevance filter: birds are scored only if they belong to a marine or coastal family **and** at least 5% of their whole global range falls over ocean, with a curated include/exclude list for edge cases.

> **!** Important
>
> The standard **WoRMS `isMarine` flag cannot be used** for this purpose, because seabirds are largely absent from that register. This is the reason the later filter is built from family membership and a measured percentage rather than from the obvious flag — see Section 7.4.

3.6 Sources in v1

v1's distributions came principally from AquaMaps global species distribution models, downscaled from their native 0.5° half-degree cells to 0.05° by bilinear interpolation, together with BirdLife range maps for birds.

> **i** AquaMaps vintage
>
> The 2025 report cites AquaMaps in two ways — a retrieval date in the Methods and "AquaMaps 2019" in the conclusions — which reads as a contradiction. It is not.
> The **model release is AquaMaps Version 10/2019**, which is what the release registry records and what the data itself carries; the later date is when that release was *retrieved*. No newer AquaMaps release has been ingested, and the suitability surfaces have not changed.

Presence values follow external convention where one exists and are assigned by the study where one does not. Continuous suitability models are used as published, without modification.

Expert range maps take a 50% presence value, following the convention for extent-of-occurrence polygons, which assert presence *somewhere* within the boundary rather than throughout it. Designated Critical Habitat takes a higher value reflecting the stronger occupancy evidence implied by a legal designation — an assumption of the study rather than an external standard, and identified as such.

3.7 Primary productivity

Net primary productivity was computed with the Vertically Generalized Production Model (Behrenfeld and Falkowski 1997) from VIIRS satellite data for the most recently completed decade. Monthly NPP was averaged annually, then a ten-year mean and standard deviation computed, downsampled to the analysis grid by bilinear interpolation, and converted to metric tons C km^{-2} yr^{-1}.

3.8 Successor releases

v1 scored **Planning Areas**; the 2026 program cycle introduced **Program Areas**, which v2 adopted. Later releases changed the taxonomy, the merge and the definition of a valid species — see Releases for what changed and when, and read this page as the record of what v1 did.

4 v7c — sea turtles and the North Atlantic right whale where the record places them

> **A release under review**
>
> v7c (titled v7.2) is a pre-release under review. It patches v7's sea-turtle and North Atlantic right whale distributions where the observational record places them. Its chapter (the rules, the evidence behind them and what moved in the scores) is published with the v7c documentation on the signed-in review host, and appears in every version of this book once v7c is released.

Part II

Science

5 Science

> **Interactive Walkthrough**
>
> New to the MST scoring pipeline? The Scoring Walkthrough provides a step-by-step interactive visualization of how input rasters are merged, rescaled by ecoregion, and averaged into final sensitivity scores.

5.1 Conceptual Framework

The term vulnerability (V) is a function of exposure (E), sensitivity (S) and adaptive capacity (A) (Equation 5.1).

$$V = f(E, S, A) \tag{5.1}$$

The more exposed and sensitive an area is—and the less able it is to recover—the more vulnerable it is to impacts from offshore human activities. The MST currently focuses on the **sensitivity** component, quantifying the intrinsic biological sensitivity of marine ecosystems based on the species present and their conservation status. Exposure and adaptive capacity components are planned for future phases (see Chapter 12).

5.2 Cell Vulnerability

For a given cell c, the vulnerability score for a species group g is the sum across all S_g species of the product of species presence probability p_{sc} and extinction risk weight w_s (Equation 5.2):

$$v_c = \sum_{s=1}^{S_g} p_{sc} \cdot w_s \tag{5.2}$$

where:

- v_c = vulnerability score of cell c for species group g

- p_{sc} = probability of species s presence in cell c (0–1 scale, from merged species distribution models)
- w_s = extinction risk weight for species s (0–1 scale, derived from ESA/MMPA/MBTA/IUCN status; see Chapter 8)
- S_g = total number of species in taxonomic group g (e.g., bird, fish, mammal, coral, invertebrate, reptile/turtle)

In other words, if a cell has many species that are both likely to be present and at high risk of extinction, it gets a higher sensitivity score. This helps identify places where rare or threatened species are concentrated.

5.3 Sensitivity Components

The MST assesses sensitivity through three components:

1. **Species** — by taxonomic category (bird, coral, fish, invertebrate, mammal, reptile/turtle): extinction risk weighted species distribution models across 9,795 valid species (see Chapter 6, Chapter 8)
2. **Habitats** — benthic habitats such as coral reefs, seamounts, and hydrothermal vents (planned for future phases)
3. **Primary Productivity** — satellite-derived net primary productivity from the Vertically Generalized Production Model (VGPM), measured as metric tons C km^{-2} yr^{-1} (see Chapter 6)

5.4 Ecoregional Rescaling

Raw cell scores vary naturally across regions due to differences in species richness and oceanographic conditions. To enable meaningful comparison, scores are rescaled to a [0–100] range within each BOEM Ecoregion using min-max normalization (Equation 5.3):

$$v'_c = \frac{v_c - v_{min}}{v_{max} - v_{min}} \times 100 \tag{5.3}$$

where v_{min} and v_{max} are the minimum and maximum cell scores for a given component within the Ecoregion.

Scores are relative within Ecoregions, not absolute. A score of 100 represents the most sensitive 0.05° cell for a given component (e.g., fish, marine mammals, or primary productivity) within a given BOEM Ecoregion—not globally. The final score is an equally weighted average of these ecoregionally rescaled components. This means that a score of 30 in Alaska reflects a different ecological context than a score of 30 in the Pacific.

Why rescale by Ecoregion? Without this step, areas with naturally higher numbers of species (like upwelling zones in the Pacific) would almost always get the highest scores, simply because they have more species. While more species may indicate greater ecological importance, it could also be argued that the fewer species in an ecosystem, the more important each is to its function and resilience. Rescaling by Ecoregion prevents scores from being overwhelmed by species-rich areas. However, comparing *across* Ecoregions would require developing a mathematical relationship between these competing arguments for sensitivity—one that is still a source of earnest academic debate. For this reason, **scores are directly comparable among Program Areas within the same Ecoregion, but should not be compared across Ecoregions**.

5.4.1 Geographic Scope

The current analysis covers 20 BOEM Program Areas from the 11th National Draft Proposed Program (2025) spanning Alaska, the Pacific, and the Gulf of America (Figure 5.1). The Atlantic OCS region is not included in this program cycle, so ecoregions without Program Areas (e.g., Northeast Continental Shelf, Southeast Continental Shelf, Washington/Oregon) are excluded from the analysis. Species whose distributions do not overlap with any Program Area will not appear in the interactive mapping applications.

The Gulf of America Program Areas—GOA Program Area A (GAA) and GOA Program Area B (GAB)—were derived from the larger Gulf of America Planning Areas and span parts of two Ecoregions: Western and Central Gulf of America (WCGOA) and Eastern Gulf of America (EGOA). Because cell scores are rescaled within each Ecoregion before aggregation, cells in GAA that fall within the WCGOA Ecoregion are rescaled relative to WCGOA min/max values, while cells within EGOA use EGOA min/max values. The aggregated Program Area score is then the area-weighted average across all constituent cells, regardless of which Ecoregion they belong to. This approach ensures that the rescaling reflects local ecological context even when a Program Area spans Ecoregion boundaries. See Chapter 10 for implementation details.

See subsequent chapters for detailed descriptions of: data sources (Chapter 6), taxonomic integration (Chapter 7), extinction risk scoring (Chapter 8), model merging (Chapter 9), and scoring methodology (Chapter 10).

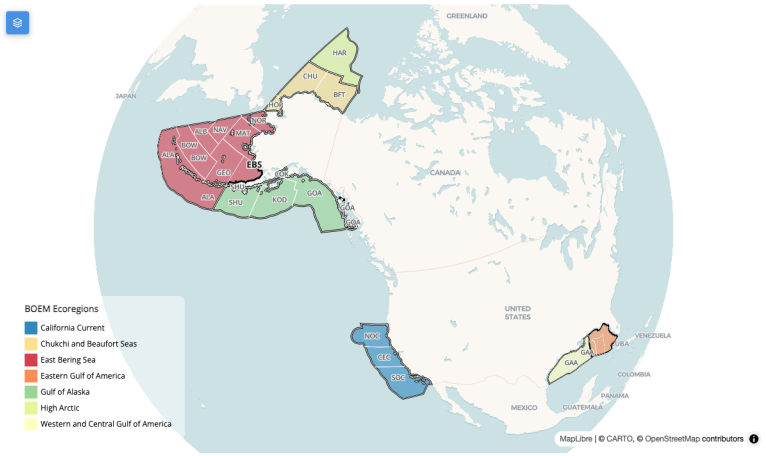


Figure 5.1: BOEM Ecoregions (colored) and Program Area outlines. **Ecoregion keys**: CAC = California Current; CBS = Chukchi and Beaufort Seas; EBS = East Bering Sea; EGOA = Eastern Gulf of America; GOA = Gulf of Alaska; HAR = High Arctic; NECS = Northeast Continental Shelf; PIS = Pacific Island Territories; PUR = Puerto Rico and U.S. Virgin Islands; SECS = Southeast Continental Shelf; WAOR = Washington/Oregon; WCGOA = Western and Central Gulf of America. **Program Area keys**: ALA = Aleutian Arc; ALB = Aleutian Basin; BFT = Beaufort Sea; BOW = Bowers Basin; CEC = Central California; CHU = Chukchi Sea; COK = Cook Inlet; GAA = GOA Program Area A; GAB = GOA Program Area B; GEO = St. George Basin; GOA = Gulf of Alaska; HAR = High Arctic; HOP = Hope Basin; KOD = Kodiak; MAT = St. Matthew-Hall; NAV = Navarin Basin; NOC = Northern California; NOR = Norton Basin; SHU = Shumagin; SOC = Southern California.

6 Data Sources

The MST integrates species distributions from **7 source datasets**, though not in equal measure — `am_0.05` alone supplies the distribution of 98% of this release's valid species. Each source predicts where a species occurs, and they differ in kind: continuous suitability models, habitats tied to a core area, critical habitat or distinct population segment, and expert range maps.

> **Note**
>
> The counts and citations on this page are read from **v3**'s own published `dataset` table, so they describe that release rather than the current one. Earlier releases integrated fewer sources: the table below *is* the answer for this version.

6.1 Dataset overview

Table 6.1: Source datasets in v3, ordered by merge priority. *Mask* datasets constrain a taxon's merged distribution to its known range as well as contributing values. *Taxa* is how many of this release's valid species the dataset actually supplied a model for — not how many models it registered.

Dataset key	Name	Cell values	Mask	Taxa	Share
`am_0.05`	AquaMaps SDM		No	9,584	97.8%
`ca_nmfs`	NMFS Core Area	Core: 100	Yes	1	<0.1%
`ch_nmfs`	NMFS Critical Habitat	EN:100, TN:50	Yes	26	0.3%
`ch_fws`	FWS Critical Habitat	EN:100, TN:50	Yes	17	0.2%
`rng_fws`	FWS Range	EN:100, TN:50, LC:1	Yes	45	0.5%
`bl`	BirdLife Range	CR:50, EN:25, VU:5, NT:2, LC:1, DD:1	Yes	193	2.0%
`rng_iucn`	IUCN Range	CR:50, EN:25, VU:5, NT:2, LC:1, DD:1	Yes	743	7.6%

! Coverage is very uneven — read the *Taxa* column, not the row count

Broad coverage. `am_0.05` (9,584), `bl` (193), `rng_iucn` (743) — between them these account for the distribution of essentially every scored species.
Targeted designations. `ca_nmfs` (1), `ch_nmfs` (26), `ch_fws` (17), `rng_fws` (45) — each under 1% of valid species. These are legal designations and expert ranges for specific taxa (critical habitat, core areas, sea turtle ranges), so they are small by nature rather than incomplete; they raise the value of the cells they cover rather than extending coverage to new species.

The **merged model** (`ms_merge`) is the derived output rather than a source, so it is excluded from the table above: for each taxon it takes the maximum value across that taxon's source datasets in each cell, constrained to the taxon's expert range where one exists (see Chapter 9).

In v3 the merged model covers **9,795** of the 9,795 valid species (100%). Every scored species goes through the merge, so the scored surface is the merged model throughout.

6.2 The sources

6.2.1 AquaMaps SDM (`am_0.05`)

AquaMaps species distribution models downscaled from 0.5° to 0.05° cells

- **Models**: 17,550
- **Response**: suitability
- **Coverage**: Global
- **Taxa**: all taxa, except birds
- **Role**: contributes values

Kaschner, K., K. Kesner-Reyes, C. Garilao, J. Segschneider, J. Rius-Barile, T. Rees, and R. Froese. 2019. AquaMaps: Predicted range maps for aquatic species. World wide web electronic publication, www.aquamaps.org, Version 10/2019. Content from AquaMaps as provided in this R package is licensed under a Creative Commons Attribution-NonCommercial 3.0 UnportedLicense, please see http://creativecommons.org/licenses/by-nc/3.0/

Source · Download · Metadata

6.2.2 NMFS Core Area (`ca_nmfs`)

Core distribution areas designated by NOAA Fisheries for species with restricted ranges, used as high-confidence habitat indicators.

- **Models**: 1
- **Response**: binary
- **Cell values**: Core: 100
- **Coverage**: USA
- **Taxa**: cetaceans
- **Role**: contributes values **and** constrains the merged range

Source · Download

6.2.3 NMFS Critical Habitat (`ch_nmfs`)

The National Marine Fisheries Service (NMFS) developed this geodatabase to standardize its Endangered Species Act (ESA) critical habitat spatial data.

- **Models**: 34
- **Response**: binary
- **Cell values**: EN:100, TN:50
- **Coverage**: USA
- **Taxa**: all taxa
- **Role**: contributes values **and** constrains the merged range

Source · Download · Metadata

6.2.4 FWS Critical Habitat (`ch_fws`)

This page contains the spatial data for active proposed and final critical habitat for FWS only and Joint FWS/NMFS threatened and endangered species. ECOS is a FWS-sponsored platform for FWS data. The ECOS critical habitat on-line mapper includes (some, not all of the) proposed and final critical habitat for species listed as Threatened and Endangered by the FWS, or that are jointly managed by FWS/NMFS. The on-line mapper does not include species which are under the sole jurisdiction of NMFS.

- **Models**: 29
- **Response**: binary
- **Cell values**: EN:100, TN:50
- **Coverage**: USA
- **Taxa**: all taxa
- **Role**: contributes values **and** constrains the merged range

Source · Download · Metadata

6.2.5 FWS Range (`rng_fws`)

This is a dataset representing the polygonal boundaries for species as understood by the United States Fish and Wildlife Service.

- **Models**: 106
- **Response**: binary
- **Cell values**: EN:100, TN:50, LC:1
- **Coverage**: USA
- **Taxa**: all taxa
- **Role**: contributes values **and** constrains the merged range

Source · Download

6.2.6 BirdLife Range (`bl`)

This dataset contains digital distribution information for the world's birds. It is a joint product of BirdLife International and Handbook of the Birds of the World.

- **Models**: 573
- **Response**: binary
- **Cell values**: CR:50, EN:25, VU:5, NT:2, LC:1, DD:1
- **Coverage**: Global
- **Taxa**: birds
- **Role**: contributes values **and** constrains the merged range

Source · Download · Metadata

6.2.7 IUCN Range (`rng_iucn`)

This dataset contains distribution information on species assessed for The IUCN Red List of Threatened Species™. The maps are developed as part of a comprehensive assessment of global biodiversity in order to highlight taxa threatened with extinction, and thereby promote their conservation.

- **Models**: 1,516
- **Response**: binary
- **Cell values**: CR:50, EN:25, VU:5, NT:2, LC:1, DD:1
- **Coverage**: Global
- **Taxa**: all taxa (except birds)

- **Role**: contributes values **and** constrains the merged range

  IUCN Red List of Threatened Species. Version 2025-2. <www.iucnredlist.org>. Downloaded on 21 January 2026.

Source

6.3 Notes on individual sources

6.3.1 AquaMaps vintage

AquaMaps is the foundational dataset for most taxa: environmental-envelope models of habitat suitability as a function of depth, temperature, salinity, primary productivity, ice concentration and distance from land (Kaschner et al. 2019). The **model vintage is AquaMaps Version 10/2019** — the release named in the citation above, and the one the registry records as its environmental end date. Later dates that appear alongside AquaMaps in project records are *retrieval* dates for that same 2019 release, not a newer one; the underlying suitability surfaces have not changed.

Models are native to a 0.5° grid (c-squares) and are interpolated onto the analysis grid **bilinearly**, implemented as a precomputed weight join rather than a per-species raster resample. Source cells that are absent contribute zero, so a distribution fades to zero within about one cell of its edge rather than ending on a hard step.

6.3.2 Where each representation lives

Several inputs are published in two representations: the **Original**, at the source's own resolution (AquaMaps' 0.5° grid, or a vector range as delivered), and the **Interpolated** surface on the 0.05° analysis grid that scoring used. The originals are stored under `native/{ds_key}/` and the gridded surfaces under `cog/{grid_id}/` in the shared, content-addressed store (Section 15.3), and each release's `native_asset` table points at both. Because the file name is derived from the source content, a dataset that is unchanged between releases is stored once and pointed at by each release that uses it, rather than copied into every release. An input delivered directly on the analysis grid has nothing to interpolate, but may still have two representations that differ in content rather than resolution: AquaX (v9) publishes the band **as delivered** beside the surface **as ingested** (the thresholded, integer values the merge consumed), and the Atlas labels that pair *Delivered / As ingested*. An input that is gridded from the start with nothing else to show (the NMFS DPS extinction-risk surfaces, and the v7 ranges whose original could not be matched) has a single representation.

6.3.3 Sea turtles: why a dedicated range dataset

Sea turtle distributions from AquaMaps and from IUCN range maps were both too broad, extending well beyond where the species actually occur. Several species also carry **different ESA protection in different parts of their range**, through NOAA Fisheries Distinct Population Segments (DPS) — which a single global range map cannot express.

The SWOT+DPS dataset replaces the IUCN range as the global mask for all six sea turtle species, combining SWOT Global Distributions (Wallace et al. 2023) for realistic range polygons with the NOAA Fisheries ESA Species Range Geodatabase (NOAA Fisheries 2025) for DPS listing status. Where DPS data exist (Loggerhead, Green, Olive Ridley), Endangered DPS areas take the Endangered value and the remainder of the SWOT range the Threatened value; for species Endangered throughout (Leatherback, Hawksbill, Kemp's Ridley) the entire range takes the Endangered value.

6.3.4 Range maps as masks

Datasets marked *Mask* do double duty: they contribute cell values **and** define the spatial extent a taxon's merged distribution may occupy. This is what stops an environmental-envelope model from asserting presence far outside a species' known range — see Chapter 9.

6.4 Standard grid

All datasets are aligned to the `usa05` grid (US study area, 0.05 deg (0-360 frame)): 3,103 × 2,006 cells of 0.05° at 0–360 longitudes. At mid-latitudes a cell is roughly 4 × 4 km (~16 km^2). Every cell carries a `cell_id`, which is what joins spatial data to every table in the database.

This release carries **661,372** cells.

> ⚠ `cell_id` means a different place on different grids
>
> The grid changed at v8, from a regional grid on 0–360 longitudes to a global grid on −180 to 180. **A given `cell_id` therefore denotes an entirely different location in v7 and in v8.** Cell ids are meaningful only together with the `grid_id` the manifest declares, and the cell-id rasters are lookup images — a grid's geometry cannot be inferred from one.

6.5 Primary productivity

Primary productivity is an explicit statutory consideration: Section 18(a)(2) of the 1978 OCSLA Amendments requires the Department of the Interior to weigh "the relative environmental sensitivity and marine productivity of different areas of the OCS" in the timing and location of OCS activities.

We use satellite-derived net primary productivity (NPP) from the Vertically Generalized Production Model [VGPM; Behrenfeld and Falkowski (1997)] produced by Oregon State's Ocean Productivity Lab, based on VIIRS data for 2014–2023. Values are converted to metric tons C km^{-2} yr^{-1} and averaged over the period. NPP enters scoring as its own metric, without extinction-risk weighting (see Chapter 10).

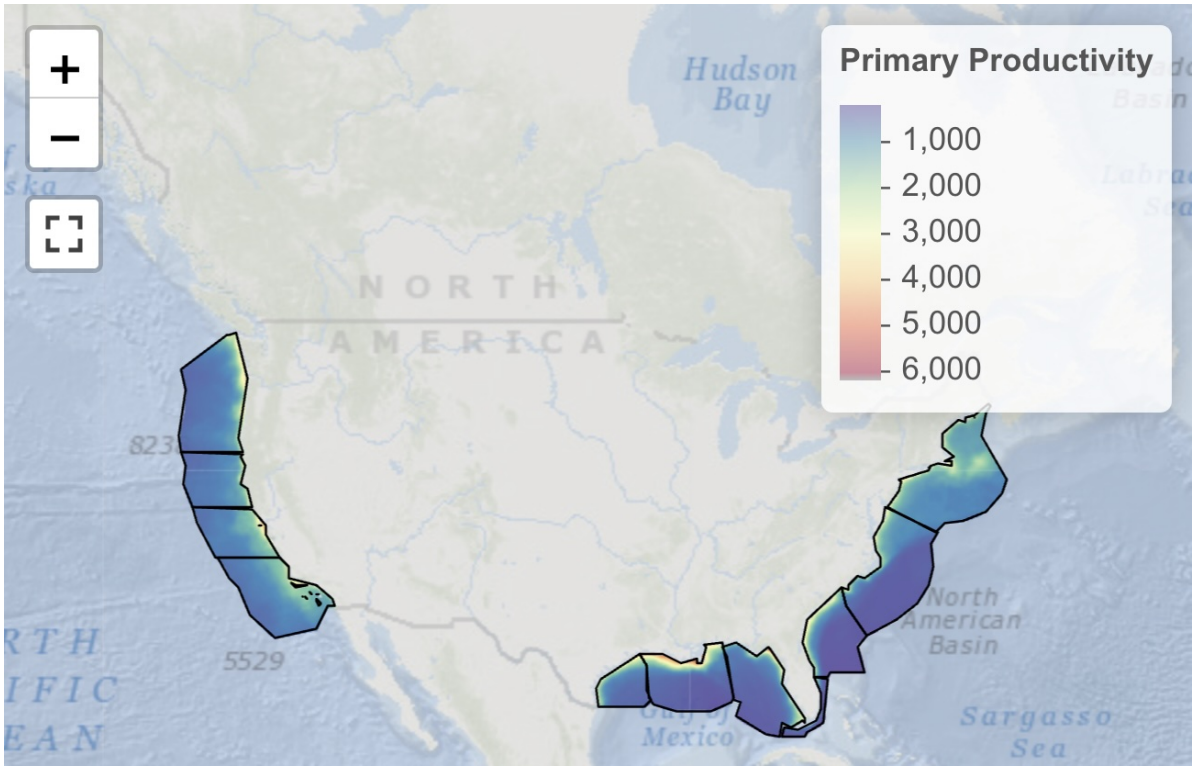


Figure 6.1: Primary productivity throughout the continental United States averaged across months of 2021, as measured by Oregon State's Vertically Generalized Production Model (VGPM).

7 Taxonomy

Integrating distribution data from several providers requires resolving taxonomic identity across naming systems, so that models of the same species merge rather than being counted as different species. The MST uses a multi-authority matching pipeline in which every source name resolves to one canonical identifier.

7.1 Two taxonomic namespaces

Taxa resolve into **one of two** authorities, not one:

- **Birds → BirdLife (`BOTW:{sisid}`)**. WoRMS coverage of birds is poor and its `isMarine` flag is unreliable for seabirds, so birds are *not* resolved through WoRMS.
- **Everything else → WoRMS (`WORMS:{aphia_id}`)**, resolved from the scientific name via the ITIS WoRMS crosswalk, then exact WoRMS matching, then the WoRMS REST API for the remainder.

A merged model is keyed by that identity — `ms_merge|WORMS:137209`, `ms_merge|BOTW:22694927` — which is also why species must be counted on the **(authority, id) pair**: the two namespaces number independently, so counting bare ids collides birds with invertebrates.

7.2 Taxonomic authorities

Table 7.1: Taxonomic authorities used for name resolution.

Authority	Role	Key
WoRMS	World Register of Marine Species — canonical identity for all non-bird marine taxa, and the source of the class/phylum used to assign a species category	`worms_id` (AphiaID)

Authority	Role	Key
BirdLife BOTW	Birds of the World — canonical identity and ranges for birds	`botw_id` (SIS ID)
GBIF	Backbone taxonomy, used as a crosswalk into WoRMS	`gbif_id`
ITIS	US federal standard; the first match attempted where a source supplies a TSN	`itis_tsn`
IUCN Red List	Conservation status, and expert range maps	`iucn_id`

At each stage deprecated names resolve to their accepted synonyms, with the original source name kept alongside the accepted one.

7.3 Species categories

Table 7.2: Species categories in v3. *Scored* is read from the metric registry — a category is scored only where the release published an `extrisk_{cat}` metric for it. *Valid* counts the taxa eligible for scoring.

Category	Taxa	Valid	Scored
`invertebrate`	8,243	5,208	Yes
`fish`	6,679	3,065	Yes
`other`	1,129	910	Yes
`coral`	775	333	Yes
`bird`	614	203	Yes
`mammal`	89	70	Yes
`turtle`	6	6	Yes
`reptile`	26	0	**No**

Categories present in the taxonomy but **not scored** in v3: `reptile`. They are carried in the taxon table — the identity is real — but no metric aggregates them, so they contribute nothing to any score.

How categories are assigned

- **Sea turtles are their own category (`turtle`)**, separated from `reptile` so that the multiplicative turtle merge and the turtle metric have somewhere to live (see Chapter 9). Non-turtle reptiles — sea snakes, marine iguanas — remain in the taxonomy but are not scored.
- `other` collects taxa this release's assignment could not place. It is a symptom rather than a category, and later releases removed the need for it by assigning from the taxonomic hierarchy instead — see Releases.

7.4 Marine relevance

Not every taxon in the source data belongs in a *marine* sensitivity analysis, and the obvious test does not work.

WoRMS `isMarine` fails for birds. Birds are resolved through BirdLife, so most have no WoRMS record at all, and those that do are inconsistently flagged; relying on it would drop nearly every seabird. Earlier releases sidestepped this with a purely **spatial** test — a taxon counted as marine if its distribution overlapped the study area — which admits any coastal or inland bird whose range happens to reach the coast.

This release predates the explicit marine-relevance cull. Marine status comes from WoRMS `isMarine`, with birds admitted through BirdLife instead, so the bird counts on this page are broader than in later releases.

7.5 From source taxa to valid species

Table 7.3: Validity gates applied cumulatively to v3's published `taxon` table. The final row applies the release's own validity flag to the whole table — it is the authoritative count, and not simply the conjunction of the diagnostic gates above it.

Filter step	Removed	Taxa remaining
Source taxa (all datasets)	—	17,561
Resolved to a taxon id	—	17,561
Has a merged distribution	−802	16,759
Not extinct	−21	16,738
Marine (WoRMS, or a BirdLife bird)	−48	16,690
Valid for scoring (the release's `is_ok`)	−6,895	9,795

This yields **9,795 valid species** across the study area.

Of those, **9,621** have modeled distributions inside the **20 BOEM Program Areas** of the current program cycle. That is a **spatial** result, obtained by asking the zone tables which species occur there — not a property of the validity flag.

> ! Validity is not the same question as "occurs in a Program Area"
>
> Through v6 the validity flag silently *required* overlap with a BOEM Program Area, conflating "this is a real, mapped marine species" with "this species happens to fall inside the areas of the current program cycle" — and so hiding biodiversity outside that cycle. From **v7** the two are separate: validity means the taxon has a modeled distribution in the study area, and Program-Area membership is a zone query answered afterwards.
> This is why the headline count is **not comparable across releases**: it rose sharply at v7 without a single species being added, because the definition changed. Per-release counts are on the Releases page.

7.6 Data quality

7.6.1 Duplicate resolution

Several datasets may supply a model for the same species. Resolving every source name to one canonical identifier is what makes those models merge — taking the maximum per cell — instead of appearing as separate species and counting the taxon more than once in every score.

7.6.2 Synonym handling

Names change as taxa are reclassified. The pipeline resolves through `acceptedNameUsageID` in each authority, keeps the original source name alongside the accepted one, and updates deprecated identifiers to their current accepted form.

7.7 Implementation

Name resolution is `msens::match_taxa()`, which runs the identifier cascade and returns the unified taxon table. Category assignment, the marine cull and the validity flags are applied in the `merge_taxon` workflow (see Chapter 16).

8 Extinction Risk

Extinction-risk scoring turns a species' conservation status into a numeric weight, so that a cell holding rare or threatened species scores higher than one holding common ones. The MST combines three US federal frameworks with the international IUCN Red List into a single score from 1 to 100.

8.1 Regulatory framework

8.1.1 Endangered Species Act (ESA)

The Endangered Species Act (1973) protects species listed as **Endangered (EN)** or **Threatened (TN)** by NMFS (marine species) or FWS (terrestrial and coastal species). ESA listing is the strongest signal available and drives the score for US-listed species.

8.1.2 Marine Mammal Protection Act (MMPA) — assigned by taxonomy

The Marine Mammal Protection Act (1972) protects **all** marine mammals in US waters regardless of ESA status, so the MST assigns it **by taxonomy**: every taxon in WoRMS class **Mammalia**, matched on AphiaID.

This is deliberate. Deriving the flag from the NMFS Species Directory instead — as earlier releases did — captures only the managed stocks listed there, roughly 70 against the ~137 marine-mammal taxa the analysis actually carries, and it silently omitted the FWS-managed species entirely (walrus, polar bear, sea otter, manatee, dugong). Taxonomy is the complete test because the statute is written in taxonomic terms.

The floor is **20**, higher than the MBTA's, because the MMPA imposes a stronger statutory standard: it affirmatively requires marine mammal populations to be maintained at Optimum Sustainable Population levels and mandates recovery, whereas the MBTA is primarily a prohibitive take statute. Marine mammals also tend to have lower reproductive rates and longer recovery times than most seabirds.

8.1.3 Migratory Bird Treaty Act (MBTA) — an explicit species list, not all birds

The Migratory Bird Treaty Act (1918) protects **migratory birds native to the United States and its territories** — which is an explicit list, not a taxonomic class.

> **!** The MBTA floor is the FWS **CFR 50 §10.13** list, not all Aves
>
> A bird absent from 10.13 — a non-US or non-migratory species — receives **no** MBTA floor. Treating "bird" as equivalent to "MBTA-protected" would apply a US statutory floor to species the statute does not cover, inflating their weight in US waters.
> Note also that most 10.13 birds are terrestrial. The list sets the *value* of the birds that reach scoring; what keeps terrestrial birds out of the score in the first place is the marine-relevance cull (see Section 7.4).

8.1.4 IUCN Red List

For species with no US federal listing, the score falls back to the IUCN Red List category: Critically Endangered (CR), Endangered (EN), Vulnerable (VU), Near Threatened (NT), Least Concern (LC) and Data Deficient (DD).

> **i** Note
>
> `TN` is an **ESA** status (Threatened), not an IUCN one — IUCN uses `NT` for Near Threatened. `msens::compute_er_score()` **errors** on a code such as `IUCN:TN` rather than scoring it through a default, because a silently-defaulted code is a wrong weight that nothing surfaces.

8.2 Scoring

Scores are computed by `msens::compute_er_score()`, on two tracks.

8.2.1 US-listed species

For a species with an ESA listing, or MMPA or MBTA protection, the score is the **maximum** of the applicable components:

$$\text{er\_score} = \max(\text{ESA base},\ \text{MMPA floor},\ \text{MBTA floor}) \tag{8.1}$$

Table 8.1: Components for a US-listed species; the final score is the maximum across those that apply.

Component	Score
ESA Endangered (EN)	100
ESA Threatened (TN)	50
ESA not listed (LC)	1
MMPA protected (all Mammalia)	20
MBTA protected (CFR 50 §10.13)	10

Worked examples:

- ESA Endangered whale (also MMPA): $\max(100, 20) = \mathbf{100}$
- ESA Threatened seabird on the 10.13 list: $\max(50, 10) = \mathbf{50}$
- Marine mammal with no ESA listing: $\max(1, 20) = \mathbf{20}$
- 10.13 migratory bird with no ESA listing: $\max(1, 10) = \mathbf{10}$

A species carrying an MMPA or MBTA floor is routed onto this track **even when it has no ESA listing**, which is what guarantees the floor is actually applied rather than being lost to the IUCN track.

8.2.2 Species with no US listing

Table 8.2: IUCN-based scores for species not listed by NMFS or FWS.

IUCN category	Score
Critically Endangered (CR)	50
Endangered (EN)	25
Vulnerable (VU)	5
Near Threatened (NT)	2
Least Concern / Data Deficient / other	1

The IUCN scale is deliberately lower than the ESA one: a US statutory listing is a stronger, jurisdiction-specific statement about a species in US waters than an international assessment.

Species with no matching code at all receive the baseline of **1**.

8.2.3 The governing score, and where it is applied

A taxon may carry several conservation codes at once — one per contributing dataset — and they can disagree, or be stale. The pipeline therefore computes one **governing** score per taxon: the most protective across its datasets, with a **US national listing overriding the IUCN category**, plus the statutory floors.

That governing score is applied **at merge time**, to the range cells — the point at which a species' presence in a cell should carry its governing risk rather than whatever code the source dataset happened to record. Scoring then reads values that already reflect it.

8.2.4 When the listing is not the whole species: spatial extinction risk

A single governing score assumes the species is listed as a species. Under the ESA the *listed entity* can instead be a Distinct Population Segment, an ESU or a subspecies, and for some species only certain populations are listed: the humpback whale has six listed DPSs and eight that are not at risk — including the ones that breed in US waters — so its species page, flattened to the highest domestic status, read Endangered everywhere. For every species whose listed entities are scoped below the species (the list is in Chapter 6 under *NMFS Distinct Population Segments*), risk is therefore a **surface**, not a scalar: each listed entity's designated critical habitat, from NOAA Fisheries' authoritative service, carries that entity's status (Endangered 100 / Threatened 50), and the rest of the species' IUCN range carries the baseline every unlisted species of its kind gets — the MMPA floor of 20 for a marine mammal, the IUCN category otherwise. Sea turtles have long had the same treatment through their own SWOT + DPS dataset. How that surface reaches the score differs between the two groups and is described in Chapter 9.

Table 8.3: Distribution of the governing `er_score` across v3's valid species.

er_score	Species	Share
100	41	0.4%
50	46	0.5%
25	37	0.4%
20	41	0.4%
10	81	0.8%
5	64	0.7%
2	73	0.7%
1	9,412	96.1%

Of v3's 9,795 valid species, **59** carry the MMPA floor and **91** the MBTA floor.

8.3 Scale

Scores run from 1 to 100 and represent a percentage-like weight — 100 being maximum risk and 1 the baseline. Integers are used for storage efficiency; conceptually the scale is a 0–1 weight multiplied by 100.

8.4 Implementation

```
msens::compute_er_score(
  extrisk_code,      # "NMFS:EN", "FWS:TN", "IUCN:CR", ...
  is_mmpa = FALSE,   # MMPA protection (all WoRMS Mammalia)
  is_mbta = FALSE)   # MBTA protection (FWS CFR 50 §10.13)
```

`extrisk_code` is `AUTHORITY:STATUS`, where the authority is `NMFS`, `FWS` or `IUCN`, and the status is an ESA code (`EN`, `TN`, `LC`) or an IUCN code (`CR`, `EN`, `VU`, `NT`, `LC`, `DD`). Any other combination is an error, not a default.

9 Model Merging

A taxon may have distribution data in several source datasets. Merging combines them into **one surface per taxon**, applying the taxon's governing extinction risk and constraining the result to where experts say the species actually occurs.

> Interactive walkthrough
>
> The Scoring Walkthrough animates this — maximum merge, range masking and the regulatory floors, step by step from input rasters.

9.1 The rule, in one place

The merge rules are `msens::merge_sql()`, `msens::turtle_sql()` and `msens::dps_sql()` — SQL held in the package rather than inside a notebook. The merge notebook executes them and the unit tests assert them against synthetic fixtures, one per taxon category, so the pipeline and its tests cannot drift apart. Changing a rule in a way that breaks a documented case fails the tests.

9.2 Pipeline

9.2.1 Step 1 — gather the taxon's models

Every model resolving to the same taxon is collected: an AquaMaps suitability surface, any regulatory habitat or range designation (NMFS, FWS), a BirdLife or IUCN range map, and for sea turtles a SWOT+DPS range.

9.2.2 Step 2 — the expert range constrains AquaMaps

This is the heart of the merge, and it is not a plain maximum.

Where a taxon **has an expert range**, the merged surface is the **range footprint**, valued at the maximum of the taxon's governing `er_score` and the AquaMaps value *at those cells*:

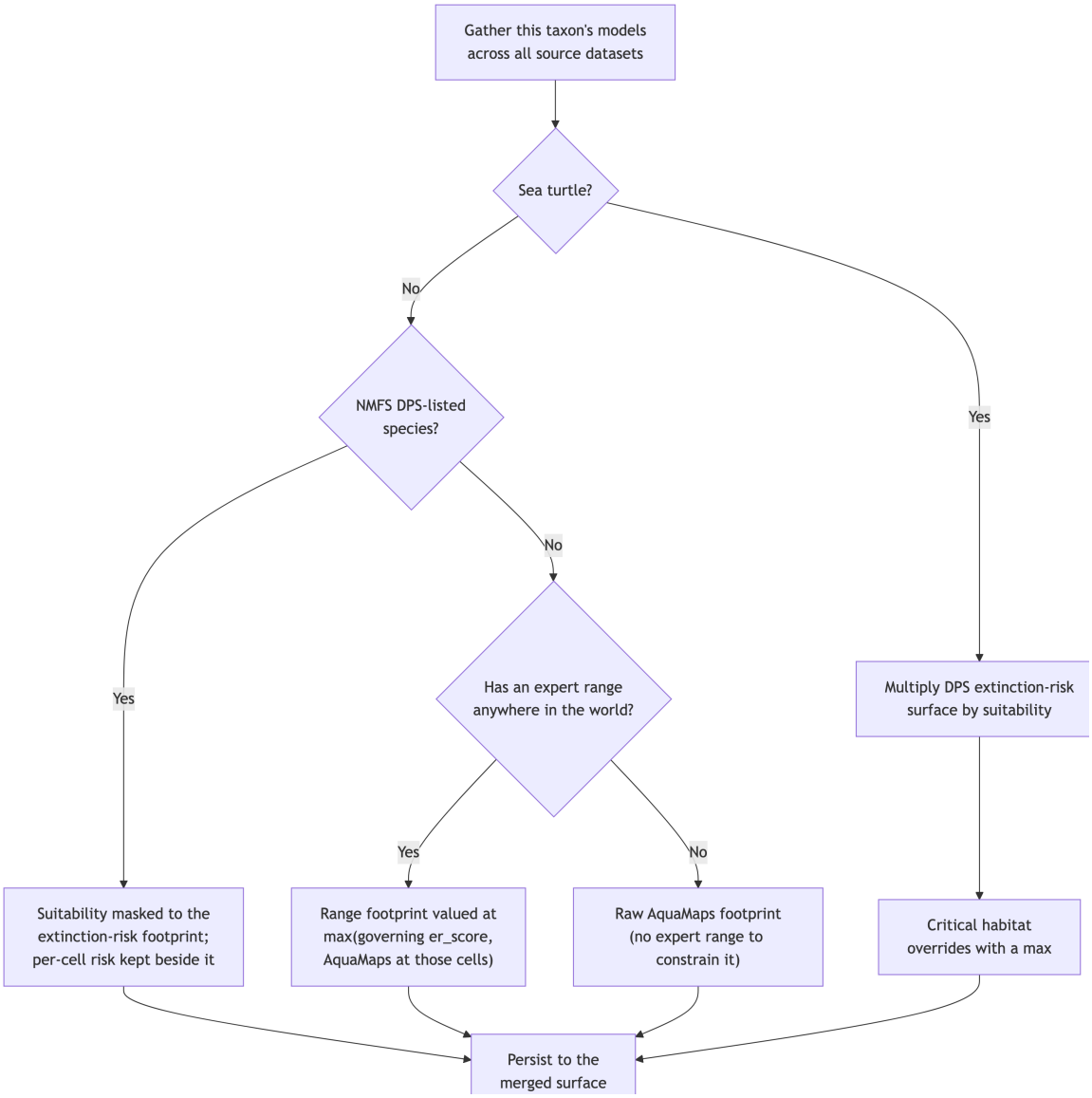


Figure 9.1: Merging one taxon's source models into a single surface.

$$v_c = \max(\text{er\_score},\ v_{am,c}) \qquad c \in \text{range} \tag{9.1}$$

AquaMaps values **beyond** the range are masked away. An environmental-envelope model will happily predict suitable habitat far outside where a species is known to occur; the expert range is what stops that prediction becoming an assertion of presence.

Where a taxon has **no expert range anywhere**, there is nothing to constrain it against, so its raw AquaMaps footprint is kept as-is.

> ! "Has a range" is a **global** question, not a local one
>
> Whether a taxon counts as range-constrained is decided on its range **anywhere in the world**, not on whether that range reaches the study area.
> That distinction is the entire mechanism for excluding species whose expert range lies wholly outside US waters. Such a species has an expert range, so it takes the range-constrained path; its range US is empty, so it contributes **no US cells**. Deciding instead on "has a range *in the US*" would push those same species onto the unconstrained AquaMaps path and re-admit roughly **750** of them — river dolphins such as *Sotalia guianensis* among them — on the strength of edge-of-range model artifacts that expert assessment places nowhere near the US.
> This was a v6 fix, lost in the v8 rewrite and since restored. It is now a named regression test.

9.2.3 Step 3 — two surfaces, not one

9.2.4 Step 4 — sea turtles multiply instead of taking a maximum

For the six sea turtle species, extinction risk **varies across the range**: NOAA Fisheries lists Distinct Population Segments separately, so the same species can be Endangered in one region and Threatened in another. A single scalar weight cannot express that.

So the turtle merge multiplies the DPS extinction-risk surface by the suitability surface:

$$v_c = \max\Big(1,\ \text{round}\Big(\frac{v_{er,c} \times v_{suit,c}}{100}\Big)\Big) \tag{9.2}$$

Critical-habitat designations then override with a maximum, so a designated cell is never scored below its designation.

Because risk is already inside the merged values, turtles receive `er_score = 100` at scoring time as a pass-through multiplier (flagged `is_er_spatial`), which cancels in the division by 100 rather than double-counting.

9.2.5 Step 5 — NMFS DPS-listed species: the model stays the distribution, risk varies by cell

A second group of species carries extinction risk that varies across the range without being sea turtles: those whose ESA *listed entity* is a Distinct Population Segment, an ESU or a subspecies rather than the whole species — the humpback whale, the Southern Resident killer whale, Cook Inlet beluga, the salmon and steelhead ESUs, both sturgeons, and the others listed under *NMFS Distinct Population Segments* in Chapter 6. For them the pipeline builds a per-cell extinction-risk surface (`dps_nmfs`): each listed entity's designated critical habitat at that entity's status, and the species' baseline everywhere else in its IUCN range (the MMPA floor of 20 for a marine mammal; the IUCN category otherwise).

These species do **not** use the turtle multiplication. Their merged value is the **distribution** — the suitability surface (AquaX, or AquaMaps where AquaX did not model the species) masked to the extinction-risk footprint, exactly the range mask every other taxon receives, with a cell that no suitability model covers valued at its risk, as in the plain rule — and the per-cell risk is carried **beside** the value rather than inside it:

$$v_c = \begin{cases} v_{suit,c} & \text{where a suitability model covers } c \\ v_{er,c} & \text{otherwise} \end{cases} \qquad er_c = v_{er,c} \tag{9.3}$$

Scoring then multiplies the two (Chapter 10), so the score is the same extinction-risk-weighted suitability the turtle rule produces — computed one step later. What changes is what the merged model *shows*: under the turtle rule the humpback, whose risk is the MMPA baseline across 99.8 % of its range, collapsed to a near-flat minimum everywhere outside critical habitat — the map drew the *weight*, not the whale. Now the merged model is the whale's habitat suitability, and its risk is a separate layer (*NMFS DPS extinction risk* in the species app).

Critical-habitat datasets are not an input for these species: `dps_nmfs` already carries every designation at its own entity's status, whereas the generic critical-habitat layers carry the species-level status (Endangered for the humpback) and would override the Threatened Mexico DPS habitat back to Endangered.

> **i** Three merge strategies
>
> - **Most taxa** — range-constrained maximum, then a **scalar** extinction-risk weight at scoring time.
> - **Sea turtles** — an extinction-risk **raster** multiplied into the merged model, so risk varies spatially within the species.
> - **NMFS DPS-listed species** — the merged model is the suitability; an extinction-risk **raster** is kept beside it and multiplied in at scoring time.

9.3 Where extinction risk enters

The governing `er_score` (Chapter 8) is applied **at merge time**, valuing the range cells — the point at which a species' presence in a cell should carry its governing risk rather than whatever code the contributing dataset happened to record. Applying it later, at scoring, would lose that: the merged surface would carry source-specific codes that may be stale or out of jurisdiction. The NMFS DPS-listed species are the deliberate exception: their risk is a per-cell surface built from the listings themselves, so it is carried beside the merged value and applied at scoring, where the per-cell weight replaces the scalar one.

9.4 The statutory floors

The MMPA (20) and MBTA (10) floors are part of the governing `er_score`, not a separate pass over the cells. A marine mammal's range cells therefore carry at least 20 wherever it is present, and a CFR 50 §10.13 migratory bird at least 10 — see Chapter 8 for how each flag is assigned, and why the MBTA one is a species list rather than "all birds".

9.5 Persisted output

The merged result is stored as:

- `model` — one row per merged model, keyed by `mdl_seq`.
- `model_cell` — one row per model × cell, holding the merged value in `val`.

9.6 Validity after merging

A taxon is flagged valid once merging has run, on the criteria in Chapter 7 — chiefly that it resolved to a taxon, is marine and not extinct, and has at least one merged cell in the study area. Program-Area overlap is **not** among them from v7 onward; it is a spatial question answered later.

In v3 this yields **9,795 valid species** across the study area, of which **9,621** contribute to scores within the 20 BOEM Program Areas of the current program cycle.

9.7 Guarding the rules

Every merge category has a synthetic fixture in the `msens` test suite asserting its exact expected output: range-only, both-masked, range-outside-the-US excluded, AquaMaps-only single-model, AquaMaps-only multi-model (which must **not** deduplicate), and the multiplicative turtle case.

This is deliberate. Whole-pipeline checks — the version-equivalence gate on Program-Area scores, and the content-hash checkpoints — catch *aggregate* drift, but they hide rule-level breakage: the ~750 wrongly-included species above moved aggregate scores almost not at all while being scientifically wrong. A category with no fixture is a category that can break silently.

10 Scoring

Scoring turns merged species distributions and extinction-risk weights into per-cell sensitivity metrics, which are then rescaled within ecoregions and aggregated to management zones.

> Interactive walkthrough
>
> The Scoring Walkthrough animates this pipeline — input rasters, merge, ecoregional rescaling and component averaging, step by step.

10.1 Cell-level scoring

For each cell, sensitivity is the extinction-risk-weighted sum of species presence within a species category:

$$\text{score}_{c,g} = \sum_{s=1}^{S_g} \frac{\text{er\_score}_s \times v_{s,c}}{100} \tag{10.1}$$

where $\text{score}_{c,g}$ is the score for cell c and category g; er_score_s is species s's extinction-risk weight (1–100, Chapter 8); $v_{s,c}$ is the merged model value for that species in that cell (0–100, Chapter 9); and S_g is the number of valid species in the category. Dividing by 100 returns the product to a 0–100 scale.

In plain terms: for every cell the MST adds up the contributions of the species found there. A cell holding many species that are both likely present and at high risk scores higher — which is how places where rare or threatened species concentrate are identified.

10.1.1 Two exceptions: species whose risk varies across the range

For sea turtles, extinction risk is already inside the merged values via the multiplicative merge (Chapter 9), so the formula collapses to a plain sum:

$$\text{score}_{c,g} = \sum_{s=1}^{S_g} v_{merged,s,c} \tag{10.2}$$

`er_score = 100` acts as a pass-through multiplier that cancels against the division. Cells inside an Endangered DPS therefore contribute more than cells inside a Threatened one, for the same species at the same suitability.

For the NMFS DPS-listed species (the humpback whale, Southern Resident killer whale, salmonid ESUs, …) the merged value is the suitability alone and the risk is a **per-cell** weight carried beside it, so the species-level `er_score` is replaced by the cell's:

$$\text{score}_{c,g} = \frac{1}{100} \sum_{s=1}^{S_g} er_{s,c} \times v_{s,c} \tag{10.3}$$

Numerically this is the turtle product taken one step later; the difference is that the merged model these species publish is their distribution, not their risk-weighted distribution. Each taxon's rule is recorded as `er_mode` in the scoring `taxon` table (`taxon` — the scalar weight; `premultiplied` — sea turtles; `cell` — DPS-listed species).

10.2 Metrics

Table 10.1: Cell-level metrics published by v3. Each has an ecoregionally rescaled counterpart, and the composite is their equally weighted mean.

Metric	Description
`extrisk_bird`	Extinction risk for bird
`extrisk_bird_ecoregion_rescaled`	Extinction risk for bird, rescaled to [0,100] based on Ecoregional min/max values
`extrisk_coral`	Extinction risk for coral
`extrisk_coral_ecoregion_rescaled`	Extinction risk for coral, rescaled to [0,100] based on Ecoregional min/max values
`extrisk_fish`	Extinction risk for fish
`extrisk_fish_ecoregion_rescaled`	Extinction risk for fish, rescaled to [0,100] based on Ecoregional min/max values
`extrisk_invertebrate`	Extinction risk for invertebrate
`extrisk_invertebrate_ecoregion_rescaled`	Extinction risk for invertebrate, rescaled to [0,100] based on Ecoregional min/max values
`extrisk_mammal`	Extinction risk for mammal
`extrisk_mammal_ecoregion_rescaled`	Extinction risk for mammal, rescaled to [0,100] based on Ecoregional min/max values
`extrisk_other`	Extinction risk for other
`extrisk_other_ecoregion_rescaled`	Extinction risk for other, rescaled to [0,100] based on Ecoregional min/max values
`extrisk_turtle`	Extinction risk for turtle

Table 10.1: Cell-level metrics published by v3. Each has an ecoregionally rescaled counterpart, and the composite is their equally weighted mean.

Metric	Description
`extrisk_turtle`	Extinction risk for turtle, rescaled to [0,100] based on Ecoregional min/max values
`primprod`	Primary productivity: Oregon State Vertically Generalized Production Model (VGPM) from Visible Infrared Imaging Radiometer Suite (VIIRS) satellite data (mg C / m^2 / day) from daily averages available as monthly averaged to annual and averaged to overall for the most recently available full years of data 2014 to 2023
`primprod_ecoregion_rescaled`	Primary productivity rescaled to [0,100] based on Ecoregional min/max values
`score_extrisk_primprod_ecoregion_rescaled`	Combined score of extinction risk and primary productivity, equally weighted (and each previously rescaled [0,100] based on Ecoregional min/max values)

One extinction-risk metric exists per **scored** species category — `invertebrate`, `fish`, `other`, `coral`, `bird`, `mammal`, `turtle` — plus primary productivity, which carries no extinction-risk weighting.

10.3 Ecoregional rescaling

Raw cell scores vary naturally between ecoregions, through differences in species richness, oceanography and biogeography. Each metric is therefore rescaled to 0–100 **within each BOEM Ecoregion**:

$$\text{score}'_c = \frac{\text{score}_c - \text{score}_{min}}{\text{score}_{max} - \text{score}_{min}} \times 100 \tag{10.4}$$

with the minimum and maximum taken over that metric's cells within the ecoregion.

> **!** Scores are relative within an ecoregion, never absolute
>
> A score of 100 marks the most sensitive cell for that metric **within that ecoregion** — not globally. A 30 in Alaska and a 30 in the Pacific describe different ecological contexts. Scores are therefore directly comparable **among Program Areas within one ecoregion**, and should **not** be compared across ecoregions.

Why rescale at all? Without it, naturally species-rich areas would almost always score highest simply for holding more species. More species may well indicate greater ecological

importance — but it can equally be argued that the fewer species an ecosystem has, the more each one matters to its function and resilience. Rescaling within ecoregions keeps scores from being dominated by richness. Comparing *across* ecoregions would require settling that argument with a mathematical relationship between the two views, which remains a live academic debate.

10.4 Spatial units

Table 10.2: Spatial units v3 scored, with the geometry vintage each used.

Unit	Field	Count	Vintage
BOEM Ecoregion	`ecoregion_key`	12	`ecoregion_2025-06`
BOEM Planning Area	`planarea_key`	36	`planarea_2025-06`
BOEM Program Area	`programarea_key`	20	`programarea_2026-01`
Subregion	`subregion_key`	4	`subregion_2025-06`

Geometry is versioned by its own **vintage** rather than by release, so several releases legitimately share one boundary set — one Program Area geometry covers v2 through v8, and one ecoregion geometry covers every release. Each release's manifest names the vintage it used, which is what lets this book draw the correct outlines for the version it documents instead of the newest ones.

10.4.1 Geographic scope

The analysis covers **20 BOEM Program Areas** from the 11th National Draft Proposed Program (2025), spanning Alaska, the Pacific and the Gulf of America. The Atlantic OCS is not part of this program cycle, so ecoregions containing no Program Area are outside the scored footprint.

The Gulf of America Program Areas (GAA and GAB) each span two ecoregions — Western and Central Gulf of America, and Eastern Gulf of America. Because rescaling happens *before* aggregation, cells are rescaled against their own ecoregion's range, and the Program Area score is then the area-weighted mean across all its cells regardless of ecoregion. Local ecological context is preserved even where a Program Area crosses a boundary.

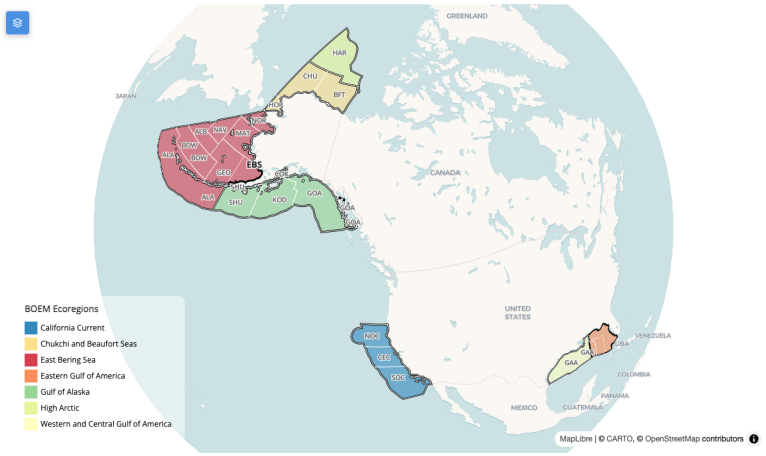


Figure 10.1: BOEM Ecoregions (colored) and Program Area outlines. **Ecoregion keys**: CAC = California Current; CBS = Chukchi and Beaufort Seas; EBS = East Bering Sea; EGOA = Eastern Gulf of America; GOA = Gulf of Alaska; HAR = High Arctic; NECS = Northeast Continental Shelf; PIS = Pacific Island Territories; PUR = Puerto Rico and U.S. Virgin Islands; SECS = Southeast Continental Shelf; WAOR = Washington/Oregon; WCGOA = Western and Central Gulf of America. **Program Area keys**: ALA = Aleutian Arc; ALB = Aleutian Basin; BFT = Beaufort Sea; BOW = Bowers Basin; CEC = Central California; CHU = Chukchi Sea; COK = Cook Inlet; GAA = GOA Program Area A; GAB = GOA Program Area B; GEO = St. George Basin; GOA = Gulf of Alaska; HAR = High Arctic; HOP = Hope Basin; KOD = Kodiak; MAT = St. Matthew-Hall; NAV = Navarin Basin; NOC = Northern California; NOR = Norton Basin; SHU = Shumagin; SOC = Southern California.

10.5 Primary productivity

Net primary productivity from the VGPM satellite model enters as its own metric (`primprod`), without extinction-risk weighting. The raw values (metric tons C km^{-2} yr^{-1}) are rescaled within each ecoregion on the same 0–100 basis as everything else.

10.6 Zone aggregation

Rescaled cell scores are aggregated to each spatial unit by area-weighted mean:

$$\text{score}_{zone} = \frac{\sum_{c \in zone} \text{score}'_c \times A_c}{\sum_{c \in zone} A_c} \tag{10.5}$$

where A_c is the area of cell c — which is latitude-dependent on a 0.05° grid, so the weighting is not cosmetic.

10.7 Visualization

10.7.1 Flower plot

The flower plot summarizes a zone's sensitivity profile by category, after the Ocean Health Index (Halpern et al. 2012):

- **petal length** — the rescaled score (0–100) for that category; longer means more sensitive
- **petal width** — that category's weight in the composite; wider means more influence
- **centre** — the weighted mean across categories

It lets a reader see at a glance which ecological elements drive sensitivity in an area. The scores app implements an interactive version.

10.7.2 Treemap

The treemap shows how individual species contribute to a score, nested first by extinction-risk category and then by taxonomy, with zoom down to single species.

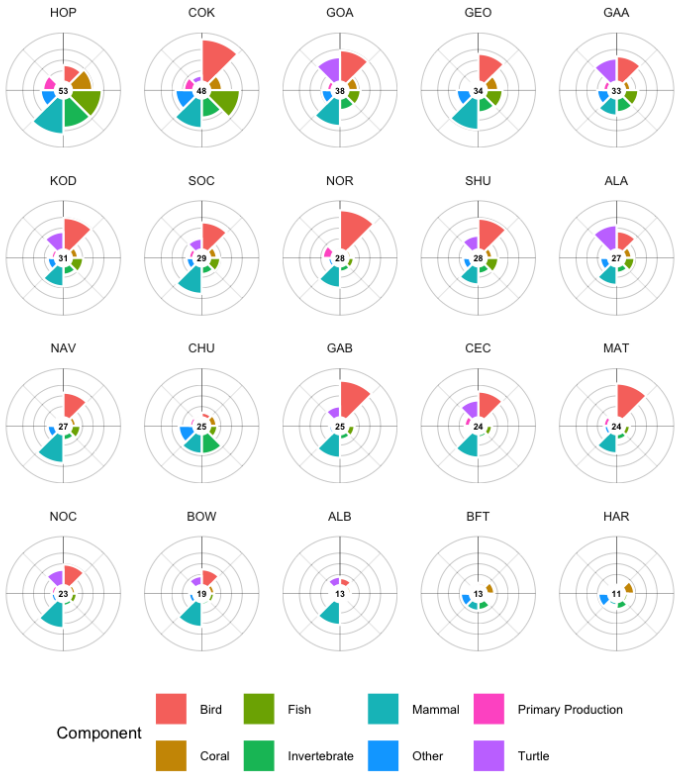


Figure 10.2: Flower plots for the BOEM Program Areas, showing ecoregionally rescaled sensitivity scores by species category.

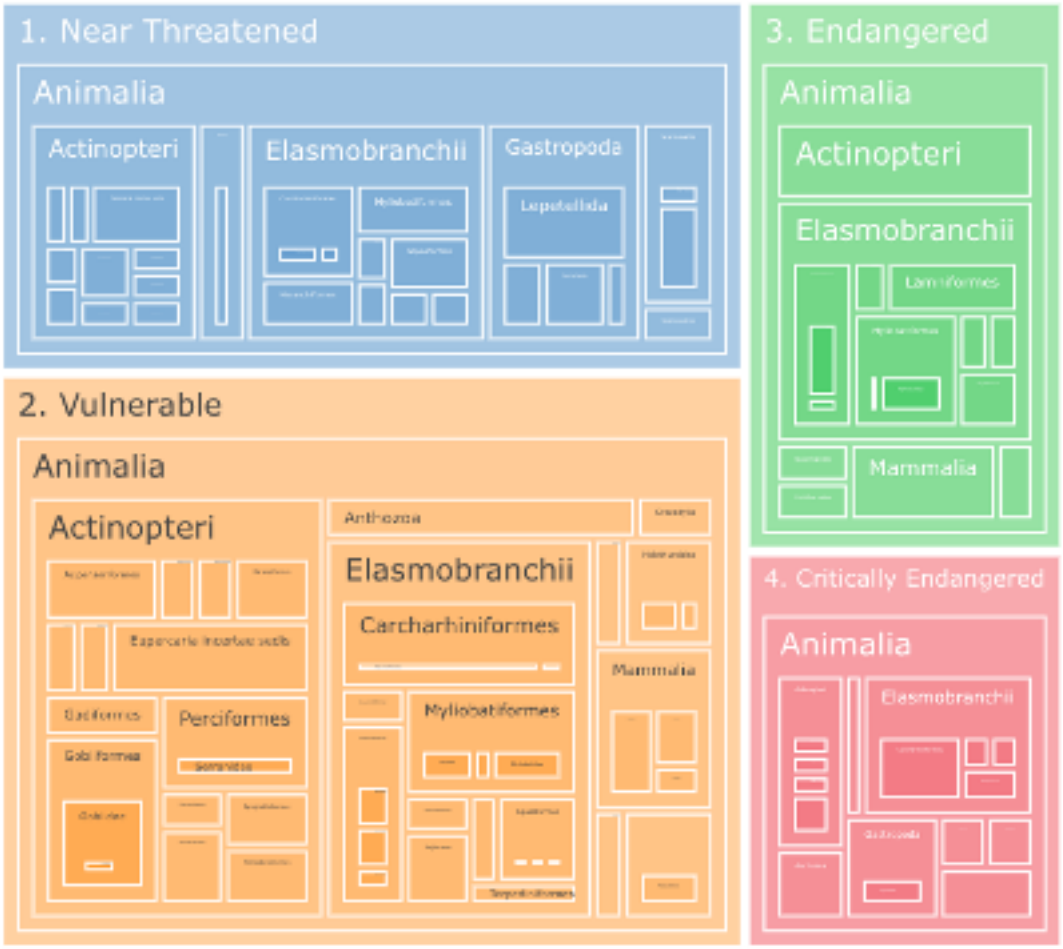


Figure 10.3: Treemap of species contributions to extinction-risk scores, organized by risk category and taxonomy.



Figure 10.4: Zoomed treemap showing individual species within a risk category and taxonomic group.

10.7.3 Score rasters

Cell-level scores are published as **Cloud-Optimized GeoTIFFs**, one per metric x subregion, with the manifest carrying each raster's URL and its build-time rescale range. The map reads those directly.

This replaced a tile factory that ran SQL against the database on every tile request. The rasters were verified byte-identical to the factory's output before the switch. They are stored **without overviews** deliberately: the renderer decimates from full resolution per request, so a pre-built pyramid would disagree with it at low zoom.

10.7.4 Interactive maps

Two Shiny applications present the results, each rendering **any published release** from the URL path rather than being frozen to one:

- **Scores** — composite and per-category sensitivity by zone, with flower plots and treemaps (see Appendix D)
- **Species** — individual and merged distribution models with regulatory status (see Appendix F)

Substitute the version in the path (e.g. `/v3/scores/`) to view that release; an older `?ver=v3` deep link still resolves, 301'd to the equivalent path. See Chapter 19.

10.8 Implementation

Scoring runs in the `score_zones`, `score_cell_metrics` and `score_zone_metrics` workflows (Chapter 16). Whole-pipeline equivalence between releases is checked by `msens::pra_score_delta`, which compares Program-Area scores on the species the two releases share.

11 Scoring Walkthrough

Interactive step-by-step visualization of the sensitivity scoring pipeline

This interactive walkthrough demonstrates how the Marine Sensitivity Toolkit calculates sensitivity scores, from input rasters through model merging, ecoregional rescaling, and final aggregation. Use the arrow keys or navigation buttons to step through the pipeline. Click any step in the top bar to jump directly to it.

For the underlying methodology, see Scoring and Model Merging. For source code, see the workflows index, which lists every notebook in the pipeline with its current dependency graph.

12 Stressors

The stressor component of the vulnerability framework (see Chapter 5) addresses the human activities that may impact marine ecosystems. This section is a placeholder for future phases of the MST, which will incorporate stressor-receptor interaction matrices to complete the full vulnerability assessment.

12.1 Offshore Wind Energy

Evaluation of stressors from the offshore wind industry needs to be evaluated based on human activities given the phase of development, whether pre-construction, construction, operation or decommissioning (Figure 12.1).

12.2 Oil & Gas

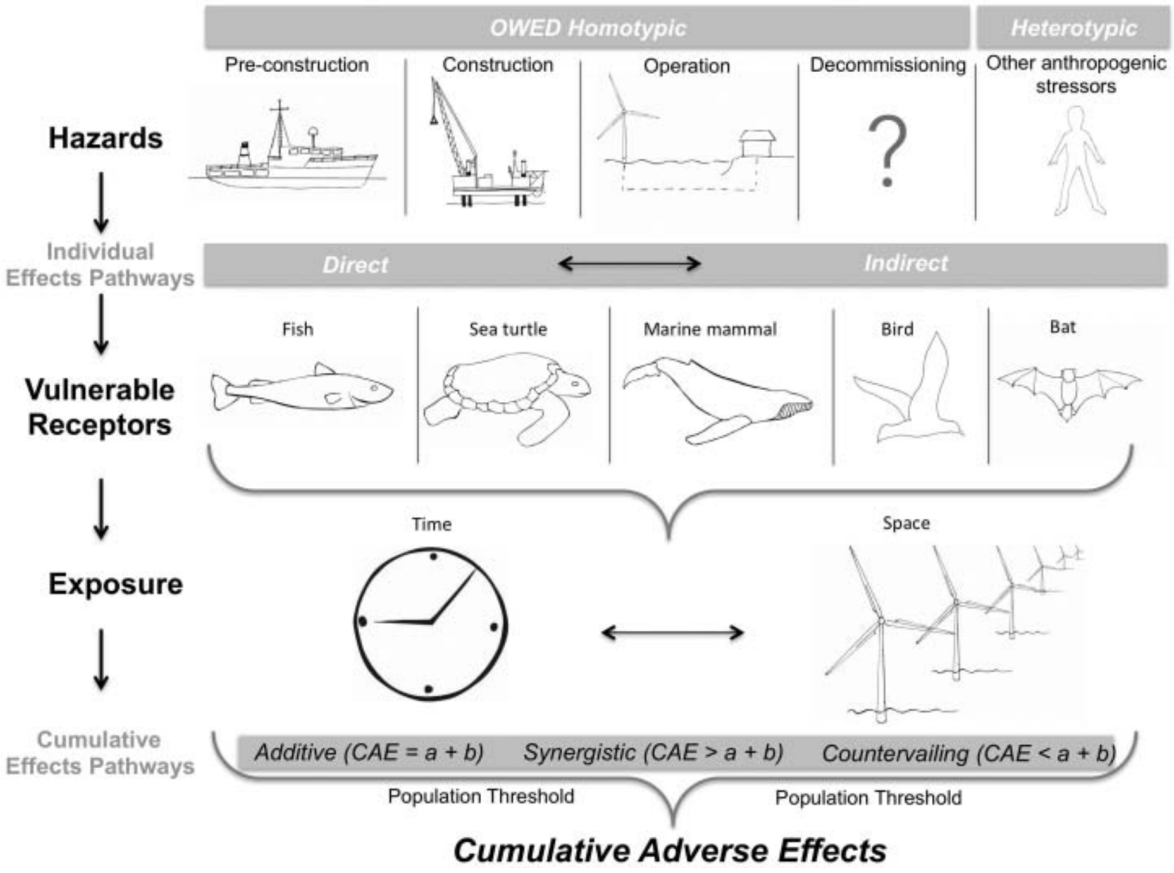


Figure 12.1: Cumulative adverse effects of offshore wind energy development on wildlife (Goodale and Milman 2016).

Part III

Software

13 Software

13.1 Software Overview

The **Marine Sensitivity Toolkit** (MST) is a stack of software components for reproducibly, interactively and hierarchically generating environmental vulnerability maps and scores. Tabular data is collated across studies evaluating sensitivity of species to oil & gas and offshore wind energy development. The best available species and benthic habitat distributions are being mosaicked across the US EEZ. Scores are summarized according to subgroups within Species, Benthic Habitats and Primary Productivity, averaged into an overall score, and visualized as a flower plot, applicable to various levels of BOEM relevancy: regional, ecoregions, protraction diagrams, blocks and aliquots.

The software components for achieving this interactively are:

1. **server** — a Docker Compose configuration orchestrating the reverse proxy, Shiny server, APIs, TiTiler raster serving and file server;
2. **database** — a columnar, embedded analytic database (**DuckDB**) as the single source of truth for species distributions, per-cell scores and derived per-zone metrics;
3. **workflows** — Quarto notebooks to ingest source datasets (AquaMaps, IUCN, NMFS/FWS, BirdLife, WoRMS, BOEM zones), merge taxonomic authorities, and compute scores that are written back to DuckDB;
4. **APIs** — a custom R `plumber` API for data requests, **TiTiler** serving the published COGs, and a searchable STAC API with one Item per model;
5. **libraries** — re-usable documented functions as the `msens` R package, including the version/manifest registry that lets one application render any release, and the tile-URL helpers the applications use;
6. **applications** — two Shiny apps (`scores`, `species`) that combine COG raster layers with static PMTiles vector layers and live-pushed choropleth paint rules, each rendering **any published release** from the URL path;
7. **documentation** — this Quarto book with figures, tables, glossary and references in interactive HTML or static DOCX / PDF output.

This toolbox is intended to primarily serve BOEM needs internally, but by being open-source and fully reproducible the hope is to enlist buy-in and even contributions from external partners, whether from other government agencies, academia, NGOs or industry.

We ascribe to the philosophy of sharing all code for the sake of reproducibility, transparency and efficiency (Maitner et al. 2024; Lowndes et al. 2017); i.e. the FAIR principles of Findability, Accessibility, Interoperability, and Reusability (Wilkinson et al. 2016).

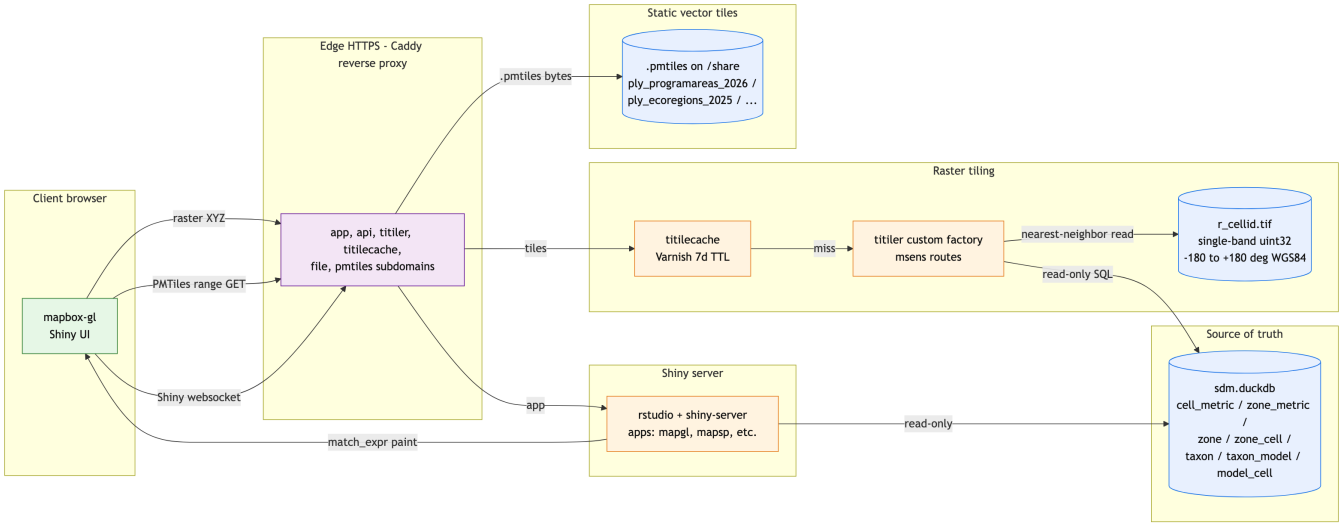


Figure 13.1: System architecture: a release is published as partitioned Parquet plus pre-rendered COGs. TiTiler serves the score rasters; zone polygons are served as static PMTiles through Caddy; the Shiny apps combine both layer types and push per-zone scores live into the polygon paint spec.

13.1.1 Interactive Applications

We have developed a series of interactive applications to explore the data and results of the MST project. These applications allow users to visualize the data, explore the results, and interact with the data in a more intuitive way. The applications are built using the `shiny` package in R (Chang et al. 2024), which allows us to easily create a user interface with complex reactivity for an interactive web application easily accessed through a web browser. The applications are designed to be user-friendly and intuitive, with interactive maps, charts, and tables that allow users to explore the data in a more dynamic way.

13.1.2 Overcoming Challenges with Large Spatial Data

The MST project incorporates many large spatial datasets that are problematic to render in a typical interactive application. For instance, the most common interactive mapping R package `leaflet` has a 4 MB limitation for displaying rasters (see "Large Raster Warning" in Raster Images • leaflet). Vectors (points, lines and polygons) get smoothed when containing many

vertices, but contiguity between polygons is lost and rendering degrades depending on the user's connection speed.

To work around these limitations we use a "cloud native" stack (see also the Cloud-Optimized Geospatial Formats Guide) that transfers only the pixels / features visible in the current viewport:

- **Raster cell values** are published as pre-rendered Cloud-Optimized GeoTIFFs and delivered as small `256×256` PNG tiles by TiTiler's stock `/cog` routes, which range-read only the part of the COG a tile needs.
- **Vector polygons** (Program Areas, Ecoregions, Planning Areas, …) are served as static PMTiles archives by the Caddy file server — a single archive per layer, range-read by the browser.
- **Choropleth values** (per-Program-Area scores) are pushed live from DuckDB into the PMTiles paint spec via a `mapbox-gl match` expression, so geometry stays static but color updates as the user changes metric / subregion.

Let's take a closer look at each.

13.1.2.1 Raster: pre-rendered COGs

Each metric (e.g. `extrisk_mammal`) resolves, in DuckDB's `cell_metric` table, to a list of `(cell_id, val)` rows — one per grid cell. Materializing the full raster and shipping it to the browser is not viable (several GiB), so only the tiles in view are transferred.

Scores are published as **one COG per metric × subregion**, written by a publishing workflow into a content-addressed store. Each release's manifest carries the href and the build-time rescale range, so the application reads them and hands the URL to TiTiler's stock `/cog` endpoints — no custom endpoint, and no database query in the tile path (Figure 13.2).

Two details are load-bearing:

- **No overviews.** The renderer decimates from full resolution on each request, so a prebuilt pyramid would disagree with it at low zoom.
- **The object key includes the encoding, not just the payload.** Rewriting an object at a stable URL once left GDAL's `/vsicurl` serving a cached header describing bytes that no longer existed: fine at z5+, HTTP 500 at z2–z4.

This replaced a **custom TiTiler factory** that rendered each tile by executing a validated SELECT against DuckDB and looking the result up against a pre-baked cell-id COG, with Varnish caching each tile URL for seven days. That path is retired but kept behind a flag (see Chapter 17); the two were verified to produce the same picture before the switch.

Binary overlays — the "cells outside Program Areas" mask — are published the same way, as their own single-color rasters rather than as a query.

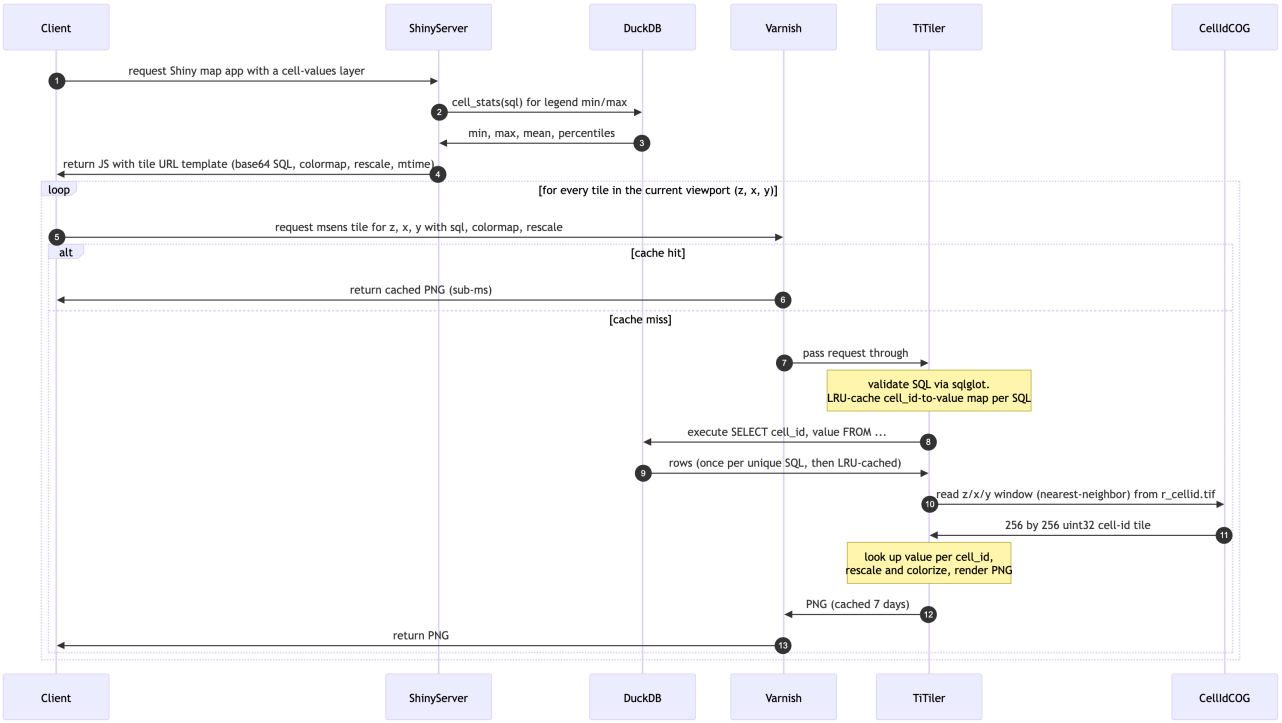


Figure 13.2: Sequence diagram: a cell-values raster layer is rendered on-the-fly by the custom msens TiTiler factory from the DuckDB source of truth. Varnish caches identical tile URLs for 7 days.

13.1.2.2 Vector: static PMTiles served by Caddy

For polygon layers (Program Areas, Ecoregions, Planning Areas, protractions, blocks, aliquots) we build PMTiles archives offline via tippecanoe and publish them as static files on the server. A PMTiles archive is a single file with an internal directory structure that lets any HTTP client issue small range requests for individual tiles; no tile server process is required — Caddy's static file handler is all we need (Figure 13.3).

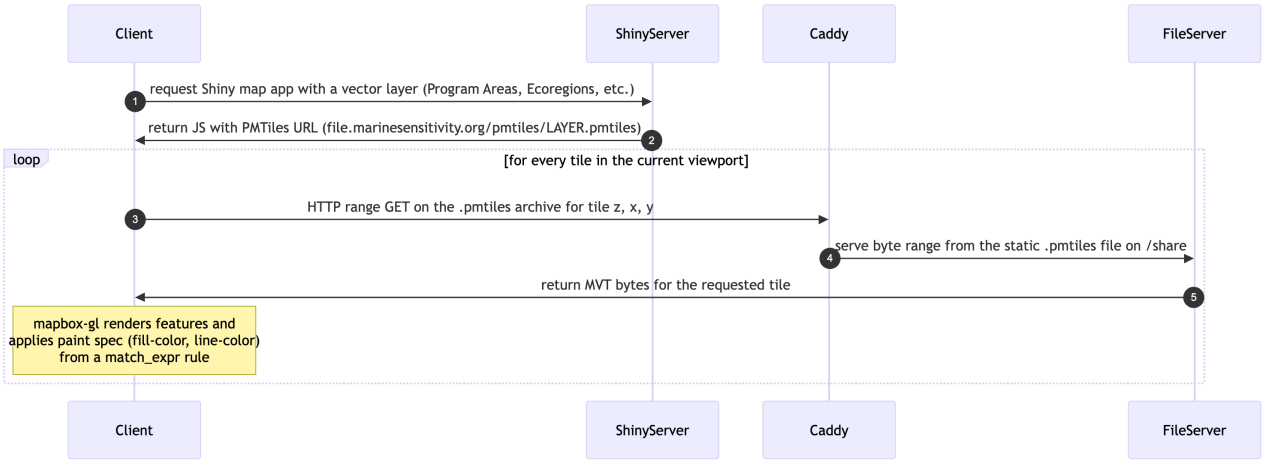


Figure 13.3: Sequence diagram: vector layers are served as static PMTiles archives through Caddy. Each tile is a byte-range read of the archive — no tile server process is in the path.

This is a deliberate simplification from the prior architecture which used PostgreSQL + PostGIS + pg_tileserv. For our workload — polygon layers that change only when the data is rebuilt — a compiled tile server and live database query engine add latency and operational complexity for no benefit over byte-range reads of a pre-baked archive. (PostgreSQL and `pg_tileserv` are now commented out of `docker-compose.yml` entirely and are not running; see Chapter 14.)

13.1.2.3 Choropleth: live DuckDB values + static PMTiles geometry

The Program Area choropleth is the interesting hybrid case. The *geometry* (Program Area polygons keyed by `programarea_key`) is static — it comes from the same PMTiles file as the polygon outlines. But the *fill color* depends on the currently-selected metric + subregion, which means the per-Program-Area score has to come live from DuckDB.

The Shiny server handles this by querying `zone_metric` (pre-aggregated per-zone values), building a `mapbox-gl match` expression that maps `programarea_key → fill_color`, and

pushing that paint rule to the map through `mapgl`'s Shiny proxy. No tiles are re-rendered; only the paint specification changes (Figure 13.4).

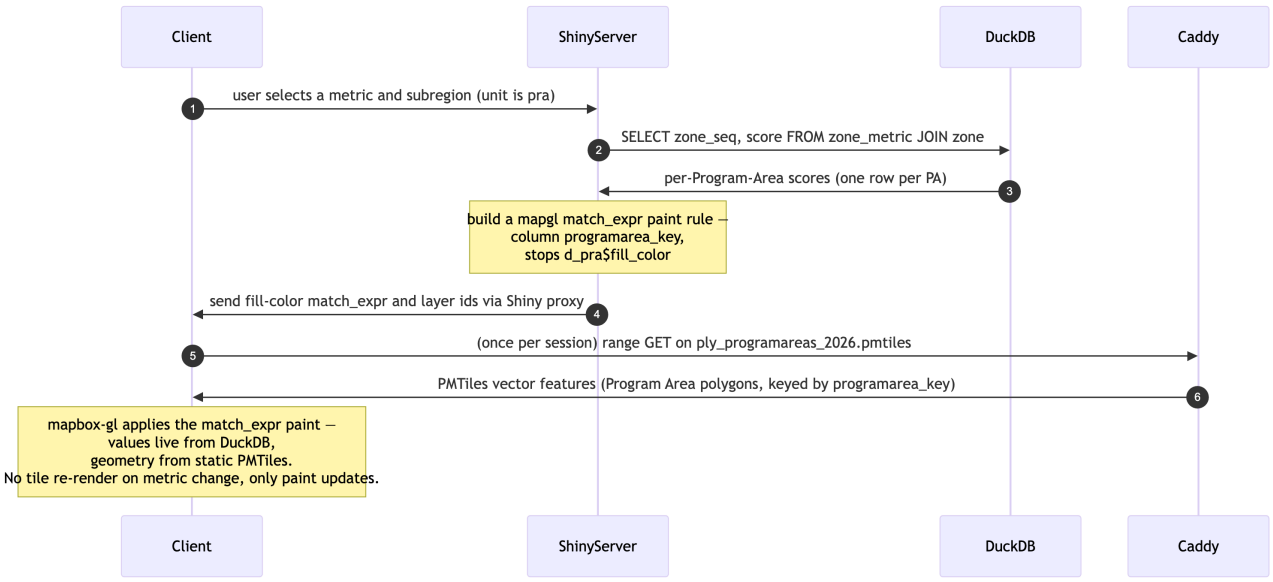


Figure 13.4: Sequence diagram: the Program Area choropleth. Polygon geometry is static (PMTiles from Caddy file server); per-area fill values come live from DuckDB's zone_metric table and are pushed into mapbox-gl's paint spec via a match expression. Switching metrics updates only the paint rule, not the tiles.

13.1.2.4 Data pipeline — how the source of truth is built

The serving paths above all sit downstream of one pipeline: ingest workflows write per-dataset distributions, the merge workflows combine them into one surface per taxon, the scoring workflows compute `cell_metric` and `zone_metric`, and the release workflow publishes the result as partitioned Parquet plus COGs, PMTiles and a STAC catalog. The whole thing is a `targets` graph generated from each notebook's own metadata, so adding a dataset means adding a notebook (Figure 13.5). See Chapter 16.

13.1.3 Github Repositories

repo	description
.github	organization README
analytics	NA
api	application programming interface (API) using R Plumber package

repo	description
app	app for viewing and summarizing MST composite scores and individual species distributions
apps	Shiny applications
atlas	Marine Sensitivity Atlas: one static browser app for scores, species, places and reports, reading the published marine-atlas release
docs	documentation for BOEM's offshore environmental sensitivity index products
manuscripts	Manuscripts with review of sensitivities by industry and receptors (species, habitats, human uses)
MarineSensitivity.github.io	default website
msens	R library of functions for mapping marine sensitivities, sponsored by BOEM
objectives	repository for issues spanning multiple repositories and doing big picture roadmapping
server	server setup for R Shiny apps, RStudio IDE, Caddy web server, h3 TiTiler service and Varnish cache
stac-sdm	STAC extension for Species Distribution Models (SDMs)
workflows	scripts for testing data analytics and visualization as well as production workflows

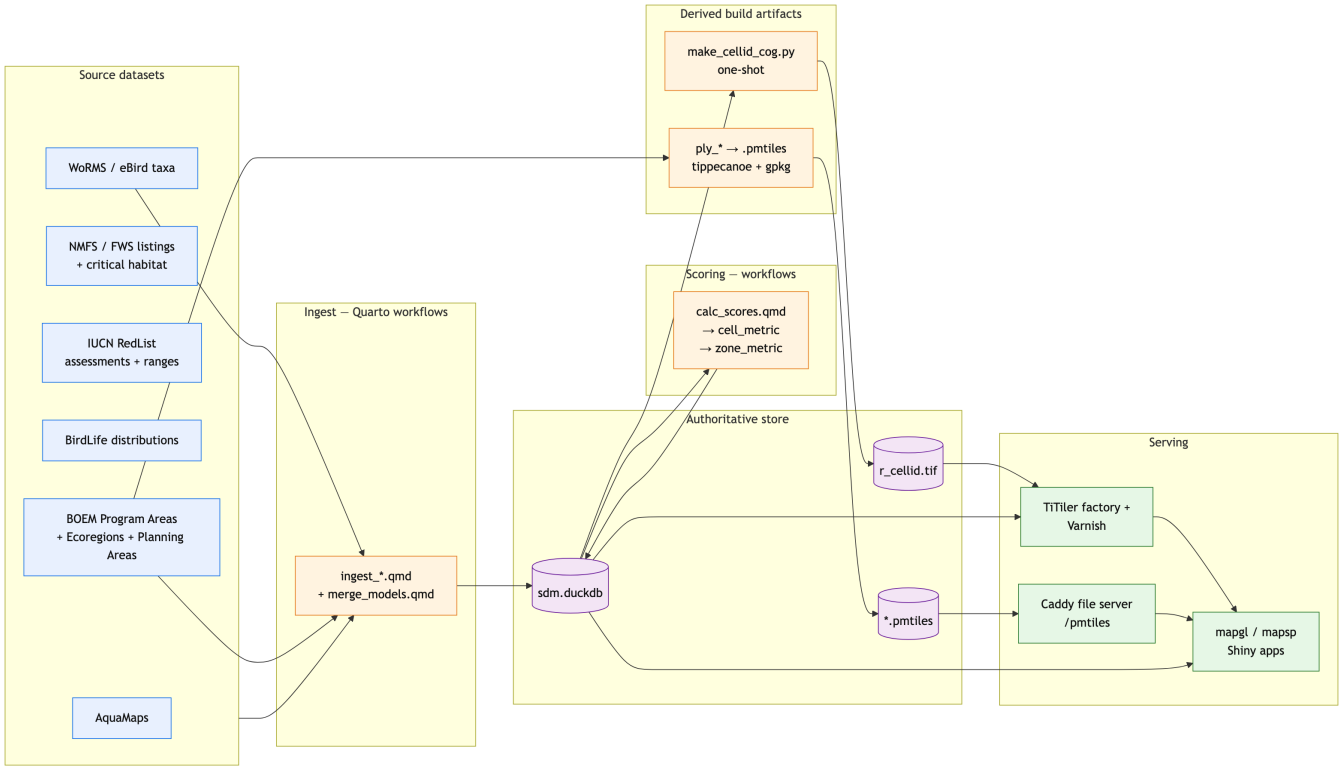


Figure 13.5: Data pipeline: ingest workflows populate DuckDB; scoring workflows compute cell_metric and zone_metric back into DuckDB; a one-shot COG regen and offline PMTiles build produce the derived artifacts the serving tier reads.

13.1.4 Software Components



APIs
application programming interfaces (APIs)



Apps
interactive applications using Shiny



Database
PostgreSQL database extended spatially with PostGIS



Docs
technical documentation



Libraries
documented functions as an R package



Server
server software configuration using Docker



Workflows
scripts for exploring plus production workflows

14 Server

The server is for serving up any web services outside those of Github (e.g., website, docs and R package msens) using Docker (see the docker-compose.yml; with reverse proxying from subdomains to ports by Caddy).

The full architecture is summarized in the Software overview. This chapter documents the concrete services running on the host and how Caddy maps subdomains to container ports.

14.1 Setup

For the latest instructions on launching an Amazon instance and installing the server software, see Server Setup · MarineSensitivity/server Wiki, which is pasted below for convenience…

on AWS as EC2 instance using Docker

14.1.1 launch instance

name: **msens1**:

- Software Image (AMI)
  Canonical, **Ubuntu**, 22.04 LTS, amd64 jammy image build on 2023-09-19 ami-0fc5d935ebf8bc3bc
- Virtual server type (instance type)
  t2.xlarge (4 vCPU, 16 GB memory)

- Firewall (security group)
  New security group
- Storage (volumes)
  2 volume(s)
  - **20 GB**
    `/` server software, disposable
  - **60 GB**
    `/share` for all data, persistent and to be backed up

14.1.1.1 allocate IP address

- Elastic IP addresses | EC2 | us-east-1 for persistent IP address
- Allocated IPv4 address: `100.25.173.0`
- Associate Elastic IP address

14.1.2 ssh to server

```
pem='/Users/bbest/My Drive/private/msens_key_pair.pem'
ssh -i $pem ubuntu@msens1.marinesensitivity.org
```

14.1.2.1 set hostname

- Change the hostname of your Amazon Linux instance - Amazon Elastic Compute Cloud

```
sudo vi /etc/cloud/cloud.cfg
# preserve_hostname: true
sudo hostnamectl set-hostname msens1.marinesensitivity.org
sudo reboot
```

14.1.2.2 mount volume

The extra volume (60 GB for `/share`) was added during EC2 launch instance wizard, but needs to be mounted before available for use.

- Make an Amazon EBS volume available for use on Linux - Amazon Elastic Compute Cloud

```
df -H
```

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        21G  2.3G   19G  11% /
tmpfs           8.4G     0  8.4G   0% /dev/shm
tmpfs           3.4G  898k  3.4G   1% /run
tmpfs           5.3M     0  5.3M   0% /run/lock
/dev/xvda15     110M  6.4M  104M   6% /boot/efi
tmpfs           1.7G  4.1k  1.7G   1% /run/user/1000
```

```
lsblk
```

```
NAME     MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
loop0      7:0    0  24.6M  1 loop /snap/amazon-ssm-agent/7528
loop1      7:1    0  55.7M  1 loop /snap/core18/2790
loop2      7:2    0  63.5M  1 loop /snap/core20/2015
loop3      7:3    0 111.9M  1 loop /snap/lxd/24322
loop4      7:4    0  40.8M  1 loop /snap/snapd/20092
xvda     202:0    0    20G  0 disk
 xvda1  202:1    0  19.9G  0 part /
 xvda14 202:14   0     4M  0 part
 xvda15 202:15   0   106M  0 part /boot/efi
xvdb     202:16   0    60G  0 disk
```

```
sudo file -s /dev/xvdb
# /dev/xvdb: data
```

So no file system on `/dev/xvdb` yet.

```
sudo mkfs -t xfs /dev/xvdb
sudo mkdir /share
sudo mount /dev/xvdb /share
```

```
sudo cp /etc/fstab /etc/fstab.orig
sudo blkid
# /dev/xvdb: UUID="bc766dfb-1c42-49cf-9320-2242a2d48a2e" BLOCK_SIZE="512" TYPE="xfs"
sudo vim /etc/fstab
# UUID=bc766dfb-1c42-49cf-9320-2242a2d48a2e  /share  xfs  defaults,nofail  0  2

df -h
sudo umount /share ; df -h
sudo mount -a      ; df -h
```

14.1.3 install docker

Following:

- Step-by-Step Guide to Install Docker on Ubuntu in AWS | by Srija Anaparthy | Medium

```
sudo apt-get update
#OLD: sudo apt-get install docker.io -y
```

NEW: [[Migrate to `docker compose`]]

```
sudo systemctl start docker
sudo docker run hello-world
sudo systemctl enable docker
docker --version
# Docker version 24.0.6, build ed223bc
sudo usermod -a -G docker $(whoami)
```

14.1.3.1 run docker compose

- /Users/bbest/My Drive/private/msens_server_env-password.txt

```
sudo chown -R ubuntu:ubuntu /share
mkdir -p /share/github/MarineSensitivity
cd /share/github/MarineSensitivity
# clone server repo
git clone https://github.com/MarineSensitivity/server.git
cd server

# add password, used as $PASSWORD in docker-compose.yml
echo 'PASSWORD=******' > .env

# launch docker instances
sudo docker-compose up -d
```

14.1.4 Backup `/share` with snapshots

Per Automate snapshot lifecycles - Amazon Elastic Compute Cloud, created two policies:

- **bkup__msens-share__daily** every 24 hrs at 09:00 UTC, max of 7
- **bkup__msens-share__weekly** every Monday 09:00 UTC, max of 8

14.2 Docker compose

The Docker compose file is used to define and run multi-container Docker applications. Here is the `docker-compose.yml` file for the server pasted for convenience …

```yaml
version: "3.9"

services:
  caddy:
    container_name: caddy
    build: ./caddy
    ports:
      - 80:80
      - 443:443
    restart: unless-stopped
    environment:
      # Cloudflare Access team domain + application AUD tag for the signed-in
      # preview.marinesensitivity.org vhost (caddy/Caddyfile jwtauth). Set both
      # in the server .env once the Access application exists
      # (server/cloudflare/). Unset, the vhost fails CLOSED: the defaults match
      # no real JWT, so every request there gets 401 rather than the app.
      CF_ACCESS_TEAM: ${CF_ACCESS_TEAM:-unconfigured}
      CF_ACCESS_AUD: ${CF_ACCESS_AUD:-unconfigured}
      # the restricted releases, as a regex alternation ("v8|v9"): the PUBLIC app
      # vhost 302s /{ver}/scores|species|atlas for these to the review host
      # instead of silently rendering the promoted release (or, for atlas, a
      # bare 404 -- this host has no other route for /{ver}/atlas/ at all).
      # Kept in the server .env beside PREVIEW_REVIEWERS_<VER>; DEPLOY_CADDY
      # asserts it against versions.json. Unset, nothing matches and the old
      # fallback behaviour stands.
      PREVIEW_RESTRICTED_VERSIONS: ${PREVIEW_RESTRICTED_VERSIONS:-__none__}
    volumes:
      - ./caddy/Caddyfile:/etc/caddy/Caddyfile
      # the preview host's routes, in their own file so caddy/test/run.sh can
      # exercise the very same file behind a test jwtauth (single-file bind
      # mount, same restart rule as the Caddyfile)
      - ./caddy/preview_routes.caddy:/etc/caddy/preview_routes.caddy
      # the version-path routes, shared by BOTH app vhosts (see the Caddyfile)
      - ./caddy/app_version_routes.caddy:/etc/caddy/app_version_routes.caddy
      # the atlas app's routes on the preview host, imported from
      # preview_routes.caddy (single-file bind mount, same restart rule)
      - ./caddy/atlas_preview_routes.caddy:/etc/caddy/atlas_preview_routes.caddy
```

```
    - /share:/share
    # the atlas-preview clone, explicitly read-only: caddy only ever serves
    # from it, atlas-preview (below) is the only writer
    - /share/atlas_preview:/share/atlas_preview:ro
    - /share/caddy/data:/data
    - /share/caddy/config:/config

rstudio:
  container_name: rstudio
  build:
    context: ./rstudio
    args:
      # which msens the IMAGE carries. Default `main` = never stale; the deploy
      # (rstudio/build.sh) passes the resolved commit instead, which both
      # records what was built and busts Docker's layer cache. The running
      # container additionally reconciles with the /share checkout at start
      # (rstudio/cont-init/03_msens_from_share).
      MSENS_REF: ${MSENS_REF:-main}
  environment:
    ROOT: 'true'
    # The login is `rstudio`, and DEFAULT_USER must say so.
    #
    # rocker's /rocker_scripts/init_userconf.sh sets USER=${DEFAULT_USER}, but it
    # only ever CREATES that account inside `if [ "$USERID" -ne 1000 ]` (or the
    # rootless branch). At the default USERID=1000 neither runs, so the image's
    # built-in `rstudio` user is all that exists - and line 167,
    #   echo "$USER:$PASSWORD" | chpasswd
    # then targets an account that was never created.
    #
    # With DEFAULT_USER: admin that failed silently on every recreate: no `admin`
    # to log in as, and `rstudio` still carrying whatever password the image was
    # built with, so BOTH logins were rejected while .env looked perfectly
    # correct. Verified by comparing the .env value against the shadow hash: NO
    # MATCH.
    #
    # Do NOT "fix" this by setting USERID to force the rename - uid 1000 is what
    # matches host `ubuntu`, and scripts/srv_render.sh renders as `-u 1000:1000`
    # precisely because of it.
    DEFAULT_USER: rstudio
    PASSWORD: ${PASSWORD}
    ADD: shiny
    # usage-log Sheet endpoint (Apps Script /exec) for the Shiny apps'
```

```
    # msens::ga_js() beacon. Set MSENS_LOG_URL in the server .env alongside
    # PASSWORD; if unset the Sheet leg is a silent no-op and only GA4 receives
    # events. The apps hold no credential - the endpoint is the whole secret,
    # so it belongs in .env (untracked), never in the app source.
    MSENS_LOG_URL: ${MSENS_LOG_URL:-}
    # signs the version token a Shiny page embeds so the session renders the
    # version its page was served for (msens::ver_token_*). Set in .env
    # (`openssl rand -hex 32`); reaches the app processes via rocker's
    # Renviron.site. Unset, msens uses a per-process random secret, which is
    # still correct -- only a session straddling a process restart is asked
    # to reload.
    MS_TOKEN_SECRET: ${MS_TOKEN_SECRET:-}
    # shared secret the PREVIEW app instance sends to the API as
    # X-MS-Preview-Token, so /report will render a restricted (under-review)
    # release for a signed-in reviewer and for nobody else. Same value in the
    # plumber service below; unset = restricted reports refused everywhere.
    MSENS_PREVIEW_TOKEN: ${MSENS_PREVIEW_TOKEN:-}
  ports:
    - 8787:8787  # rstudio
    - 3838:3838  # shiny
  restart: unless-stopped
  # MEMORY CEILING (2026-09-24 outage): a Shiny Server app worker (uid 996 `shiny`) grew to
  # 7.8 GB resident inside this container; with no cgroup limit and no swap the host spent
  # hour reclaiming page cache (ssh, apps, STAC, API all unreachable) before the OOM killer
  # fired, and only an EC2 reboot ended it. With a ceiling, a runaway worker is killed INSI
  # this container (its Shiny session errors; everything else keeps serving). 9 GB leaves
  # ~6 GB for erddap/titiler/plumber/stac-api/caddy on the 16 GB host at their measured idl
  # (~3.2 GB). memswap_limit == mem_limit: this container never spills into the host swap
  # (host/swap.sh); swap exists so the REST of the host degrades instead of wedging.
  mem_limit: 9g
  memswap_limit: 9g
  volumes:
    - /share:/share
    - /share/shiny_apps:/srv/shiny-server
    # Shiny Server config with a SECOND server block (:3839) for the preview
    # instance -- same app code, MS_PREVIEW=1 -- that only Caddy's signed-in
    # preview.marinesensitivity.org vhost proxies to. Single-file bind mount:
    # a change needs `docker compose restart rstudio` (inode rule, as Caddyfile).
    - ./rstudio/shiny-server.conf:/etc/shiny-server/shiny-server.conf:ro
    # the preview instance's site_dir: 3-line wrapper app.R per app that sets
    # MS_PREVIEW=1 and shinyAppDir()s the real app; `touch restart.txt` here
    # reloads it (DEPLOY_APPS does). Not published as a port on purpose.
```

```
    - ./rstudio/shiny_apps_preview:/srv/shiny-server-preview
    # Release notebooks rendered INSIDE this container publish to S3. The image
    # carries the AWS CLI, but a CLI with no credentials just fails later and
    # louder: `NoCredentials` at the publish step, after the build work is done.
    # Read-only, and the container already runs as uid 1000 = host `ubuntu`,
    # which owns these.
    - /home/ubuntu/.aws:/home/rstudio/.aws:ro

# warm: keep the versions people actually open already built, so the first
# visitor after a deploy or a quiet night does not pay the ~13-17 s cold start
# (app_idle_timeout can only keep alive what someone already started). Reads
# the registry for WHICH versions - promoted + restricted - and hits the
# container ports directly, so no Cloudflare round trip and no Access token.
# See warm/warm.sh.
warm:
  container_name: warm
  image: alpine:3.22
  restart: unless-stopped
  depends_on: [rstudio]
  entrypoint: ["/bin/sh", "-c"]
  command: ["apk add --no-cache curl jq >/dev/null && exec /warm.sh"]
  volumes:
    - ./warm/warm.sh:/warm.sh:ro

# docs-preview: the rendered books of RESTRICTED docs releases. GitHub Pages
# cannot be gated, so the docs CI publishes restricted versions to the
# `gh-pages-preview` branch (public versions go to gh-pages as always) and
# Caddy's signed-in preview vhost serves this clone at /docs/{ver}/. The
# sidecar keeps the clone current by polling -- no inbound secret, no manual
# step; a docs push shows up within 5 min. Runs as uid 1000 (= host `ubuntu`)
# so nothing under /share ends up root-owned. The branch may not exist yet on
# a fresh install: the loop just retries.
docs-preview:
  container_name: docs-preview
  image: alpine/git:v2.49.1
  restart: unless-stopped
  user: "1000:1000"
  environment:
    HOME: /tmp
    GIT_TERMINAL_PROMPT: "0"
  entrypoint: ["/bin/sh", "-c"]
  command:
```

```
    - |
      REPO=https://github.com/MarineSensitivity/docs.git
      while true; do
        if [ ! -d /repo/.git ]; then
          git clone --quiet --branch gh-pages-preview --single-branch "$$REPO" /repo \
            && echo "cloned gh-pages-preview" \
            || echo "clone failed (gh-pages-preview may not exist yet); retrying in 5 min"
        else
          git -C /repo pull --ff-only --quiet || echo "pull failed; retrying in 5 min"
        fi
        sleep 300
      done
  volumes:
    # HOST STATE: this directory must be owned by uid 1000 (`sudo chown
    # 1000:1000 /share/docs_preview`). Docker auto-creates a missing bind-mount
    # source as root:root, and the sidecar then cannot write its clone.
    - /share/docs_preview:/repo

# atlas-preview: the STATIC BUILD of the atlas app, for RESTRICTED releases.
# GitHub Pages serves `gh-pages` publicly and cannot be gated, so the preview
# host's Caddy vhost (caddy/atlas_preview_routes.caddy) serves this clone at
# /{ver}/atlas/ instead -- the SAME dist/, the SAME commit, just a different
# door (atlas/CLAUDE.md: the build is base-relative and router-free, so it
# runs unmodified under either mount point). This sidecar keeps the clone
# current by polling the same `gh-pages` branch the public host reads -- no
# inbound secret, no manual step; an atlas push shows up within 5 min. Runs as
# uid 1000 (= host `ubuntu`) so nothing under /share ends up root-owned. The
# branch may not exist yet before atlas-0's first CI publish: the loop just
# retries.
atlas-preview:
  container_name: atlas-preview
  image: alpine/git:v2.49.1
  restart: unless-stopped
  user: "1000:1000"
  environment:
    HOME: /tmp
    GIT_TERMINAL_PROMPT: "0"
  entrypoint: ["/bin/sh", "-c"]
  command:
    - |
      REPO=https://github.com/MarineSensitivity/atlas.git
      while true; do
```

```
        if [ ! -d /repo/.git ]; then
          git clone --quiet --branch gh-pages --single-branch "$$REPO" /repo \
            && echo "cloned gh-pages" \
            || echo "clone failed (gh-pages may not exist yet); retrying in 5 min"
        else
          git -C /repo pull --ff-only --quiet || echo "pull failed; retrying in 5 min"
        fi
        sleep 300
      done
  volumes:
    # HOST STATE: this directory must be owned by uid 1000 (`sudo chown
    # 1000:1000 /share/atlas_preview`). Docker auto-creates a missing
    # bind-mount source as root:root, and the sidecar then cannot write its
    # clone.
    - /share/atlas_preview:/repo

# plumber: DuckDB-backed custom API (postgres stack retired 2026-06). Serves
# the OBIS H3 summary endpoint /h3 (obisindicators::obis_h3t_sql) from the same
# store as the h3t tile service. Caddy already routes
# api.marinesensitivity.org -> plumber:8888. No postgis dependency: the
# postgres-backed endpoints degrade (see api/plumber.R lazy `con`).
plumber:
  container_name: plumber
  build: ./plumber
  environment:
    MSENS_OBIS_DUCKDB: /share/data/obis/obis_h3.duckdb
    # the other half of the review gate's report check: /report renders a
    # restricted (under-review) release only for a caller presenting this
    # secret in X-MS-Preview-Token -- i.e. the preview Shiny instance. Must
    # match the rstudio service's value; unset = restricted reports refused.
    MSENS_PREVIEW_TOKEN: ${MSENS_PREVIEW_TOKEN:-}
  ports:
    - 8888:8888  # api
  volumes:
    - /share:/share
  restart: unless-stopped

# --- postgres stack retired (2026-06) to free resources for OBIS h3 build ---
# plumber:
#   container_name: plumber
#   build: ./plumber
#   ports:
```

```
#     - 8888:8888  # api
#   restart: unless-stopped
#   volumes:
#     - /share:/share
#   depends_on:
#     - postgis

# postgis:
#   container_name: postgis
#   image: postgis/postgis:latest
#   environment:
#     POSTGRES_DB: msens
#     POSTGRES_USER: admin
#     POSTGRES_PASSWORD: ${PASSWORD}
#     ANON_PASSWORD: ${ANON_PASSWORD}
#     PGDATA: /share/postgis/data
#   volumes:
#     - ./postgis/init.sh:/docker-entrypoint-initdb.d/init.sh
#     - /share:/share
#     - /share/postgis:/var/lib/postgresql
#   restart: unless-stopped
#   healthcheck:
#     test: 'exit 0'
#   ports:
#     - 5432:5432

# pgadmin:
#   container_name: pgadmin
#   image: dpage/pgadmin4:8.14
#   restart: always
#   environment:
#     PGADMIN_DEFAULT_EMAIL: ben@ecoquants.com
#     PGADMIN_DEFAULT_PASSWORD: ${PASSWORD}
#     PGADMIN_LISTEN_PORT: 8088
#   ports:
#     - 8088:8088
#   volumes:
#     - /share/pgadmin:/var/lib/pgadmin
#   depends_on:
#     - postgis

# pgbkups:
```

```
#   container_name: pgbkups
#   image: prodrigestivill/postgres-backup-local
#   restart: always
#   user: postgres:postgres
#   volumes:
#     - /share/postgis_backups:/backups
#   links:
#     - postgis
#   depends_on:
#     - postgis
#   environment:
#     - POSTGRES_HOST=postgis
#     - POSTGRES_DB=msens
#     - POSTGRES_USER=admin
#     - POSTGRES_PASSWORD=${PASSWORD}
#     - POSTGRES_EXTRA_OPTS=-Z6 --blobs
#     - SCHEDULE=@daily
#     - HEALTHCHECK_PORT=8088
#     - BACKUP_KEEP_DAYS=2
#     - BACKUP_KEEP_WEEKS=1
#     - BACKUP_KEEP_MONTHS=2

# tile:
#   container_name: tile
#   environment:
#     DATABASE_URL: 'postgresql://admin:${PASSWORD}@postgis:5432/msens'
#   image: pramsey/pg_tileserv:latest
#   depends_on:
#     - postgis
#   ports:
#     - 7800:7800

# tilecache:
#   container_name: tilecache
#   image: varnish:latest
#   volumes:
#     - /share:/share
#   ports:
#     - 6081:6081
#   environment:
#     VARNISH_BACKEND_HOST: tile
#     VARNISH_BACKEND_PORT: 7800
```

```
#     VARNISH_HTTP_PORT: 6081
#   restart: always
#   depends_on:
#     - "tile"
# --- end postgres stack ---

# rest:
#   container_name: rest
#   environment:
#     PGRST_DB_URI: 'postgresql://anon:${ANON_PASSWORD}@postgis:5432/msens'
#     PGRST_OPENAPI_SERVER_PROXY_URI: http://127.0.0.1:3000
#     PGRST_DB_ANON_ROLE: anon
#   image: postgrest/postgrest
#   depends_on:
#     - postgis
#   ports:
#     - "3000:3000"

# swagger:
#   container_name: swagger
#   image: swaggerapi/swagger-ui
#   depends_on:
#     - rest
#   ports:
#     - "8080:8080"
#   expose:
#     - "8080"
#   environment:
#     API_URL: https://rest.MarineSensitivity.org/

titiler:
  container_name: titiler
  build: ./titiler
  environment:
    PORT:               8000
    # WORKERS_PER_CORE: 1
    MSENS_CELLID_COG: /share/data/derived/r_cellid.tif
    MSENS_DUCKDB:     /share/data/big/latest/sdm.duckdb
    MSENS_MAX_ROWS:   "1000000"
    MSENS_LRU_SIZE:   "128"
  volumes:
    - /share:/share
```

```
    ports:
      - "8000:8000"
    restart: always

  # v8 parallel serving: a tiny view-only DuckDB whose model_cell is a VIEW over the
  # partitioned Parquet on S3 (marine-atlas/v8), rendered on the global [-180,180] cell COG.
  # v7 titiler above stays fully live (A/B). serve.duckdb is KB - no multi-GB rsync.
  # Stock titiler: /cog serves the precomputed COGs every app layer now uses.
  #
  # The custom /msens cells factory (DuckDB query -> tile) is RETIRED but KEPT:
  # set MSENS_FACTORY=1 to mount it for exploratory on-the-fly queries. It is off
  # by default because nothing reaches it - the scores app's SQL fallback cannot
  # fire (every metric on v1-v8 has a COG) and the species app's needed only the
  # 192 taxa that now have merged COGs. Verified equivalent first: 100/105 tiles
  # byte-identical, the rest differing by 3-6 px of 262,144 from float32 storage.
  #
  # Deliberately NOT proxied through titilecache (Varnish): a query-driven tile
  # is not cache-friendly the way an immutable COG is, and caching an exploratory
  # endpoint mostly serves stale answers to whoever is iterating on the query.
  # The MSENS_* vars below stay so enabling the factory needs no other change.
  titiler-v8:
    container_name: titiler-v8
    build: ./titiler
    environment:
      PORT:             8000
      MSENS_CELLID_COG: /share/data/derived/r_cellid_global.tif
      MSENS_DUCKDB:     /share/data/big/v8/serve.duckdb
      MSENS_MAX_ROWS:   "1000000"
      MSENS_LRU_SIZE:   "128"
    volumes:
      - /share:/share
    ports:
      - "8001:8000"
    restart: always

  # stac-api: a SEARCHABLE STAC API with one Item per model, complementing (not
  # replacing) the static catalog at file.marinesensitivity.org/stac.
  #
  # The static catalog is dataset-level: one Item per collection whose assets are
  # S3 PREFIXES (…/native/am_native/, which 403s) and whose bbox is the whole
  # dataset envelope - so it can answer neither "the asset for this model" nor
  # "which models cover this area". This serves 25,643 per-model Items with real
```

```
  # asset hrefs and real footprints.
  #
  # Backend reads stac-geoparquet via DuckDB - the same Parquet-on-DuckDB shape as
  # the rest of this stack. Items and collection.json come from the tree that
  # workflows/publish_stac_api.qmd builds and rsyncs; PARQUET_URLS_JSON (collection
  # id -> parquet path) is passed by that notebook from parquet_urls.json, so
  # adding a dataset needs no edit here.
  #
  # Pinned to an exact upstream commit built straight from GitHub - the project is
  # marked EXPERIMENTAL by both its repo and the stac-fastapi docs (pgstac and
  # Elasticsearch are the production-ready backends), so a moving default branch
  # is not something to serve from. Treat this as a trial surface, not load-bearing.
  stac-api:
    container_name: stac-api
    build: ./stac-api          # upstream pinned + ST_Intersects(WKB) patch; see its Dockerfi
    environment:
      APP_HOST:          0.0.0.0
      APP_PORT:          "8084"
      ENVIRONMENT:       production
      WEB_CONCURRENCY:   "4"
      BACKEND:           duckdb
      STAC_FILE_PATH:    /app/stac_collections
      PARQUET_URLS_JSON: ${PARQUET_URLS_JSON:?written to .env by workflows/publish_stac_api.q
    volumes:
      # read-only: the API must never mutate the published catalog. The version is STAC_API_V
      # written by publish_stac_api.qmd beside PARQUET_URLS_JSON: a hardcoded v8 here once se
      # PARQUET_URLS_JSON against v8's tree, and every collection answered "No Parquet config
      - /share/data/derived/stac-api/${STAC_API_VER:?written to .env by workflows/publish_sta
    ports:
      - "8085:8084"
    restart: unless-stopped
    command: bash -c "python -m stac_fastapi.duckdb.app"

  titilecache:
    container_name: titilecache
    image: varnish:latest # 7.4.2  # last updated: 2023-12-26
    volumes:
      - /share:/share
      - "./varnish/titiler.vcl:/etc/varnish/default.vcl:ro"
    ports:
      - 6082:6082
    environment:
```

```
    VARNISH_BACKEND_HOST: titiler
    VARNISH_BACKEND_PORT: 8000
    VARNISH_HTTP_PORT:    6082
  command: "-p default_keep=604800" # 7d, matches vcl TTL
  restart: always
  depends_on:
    - "titiler"

# h3t: OBIS biodiversity-by-hex tile factory (DuckDB SELECT -> h3j JSON tiles)
h3t:
  container_name: h3t
  build: ./h3t
  environment:
    H3T_DBS:             "obis:/share/data/obis/obis_h3.duckdb"
    H3T_DEFAULT_DB:      obis
    H3T_MAX_ROWS:        "50000"
    # common filtered maps now hit precomputed idx_h3 / idx_h3_taxon layers;
    # these caps are the stopgap for the remaining live occ_h3 paths (year
    # filters, multi-value / fine-rank taxa). raise the timeout above the
    # app-side client timeout in apps/h3-db/app.R.
    H3T_STMT_TIMEOUT_MS: "8000"
    H3T_THREADS:         "4"      # cap serving CPU (don't peg all cores)
    H3T_MEMORY_LIMIT:    "6GB"    # cap serving RAM (occ_h3 rollups can spill)
    # /h3t/subtree + /h3t/taxon (obis-hex app): own DuckDB instance, capped
    # separately; see h3t/README.md "subtree endpoint"
    H3T_SUBTREE_THREADS:      "2"
    H3T_SUBTREE_MEMORY_LIMIT: "1GB"
    H3T_SUBTREE_TIMEOUT_S:    "60"
    H3T_SUBTREE_MAX_CELLS:    "200000"
  volumes:
    - /share:/share
  ports:
    - "8889:8889"
  restart: always

h3tcache:
  container_name: h3tcache
  image: varnish:latest # 7.4.2  # last updated: 2023-12-26
  volumes:
    - /share:/share
    - "./varnish/h3t.vcl:/etc/varnish/default.vcl:ro"
  ports:
```

```
    - 6083:6083
  environment:
    VARNISH_BACKEND_HOST: h3t
    VARNISH_BACKEND_PORT: 8889
    VARNISH_HTTP_PORT:    6083
  command: "-p default_keep=604800" # 7d, matches vcl TTL
  restart: always
  depends_on:
    - "h3t"
```

14.3 DNS

The domain name server (DNS) records are managed by SquareSpace. The subdomains point to the server on Amazon at `100.25.173.0`, whereas the main website is hosted by Github servers, per Managing a custom domain for your GitHub Pages site - GitHub Docs.

Host	Type	Data
@	A	185.199.111.153
@	A	185.199.110.153
@	A	185.199.109.153
@	A	185.199.108.153
api	A	100.25.173.0
app	A	100.25.173.0
file	A	100.25.173.0
msens1	A	100.25.173.0
pgadmin	A	100.25.173.0
pmtiles	A	100.25.173.0
rstudio	A	100.25.173.0
shiny	A	100.25.173.0
tile	A	100.25.173.0
tilecache	A	100.25.173.0
titiler	A	100.25.173.0
titilecache	A	100.25.173.0
www	CNAME	marinesensitivity.org

14.4 Caddyfile

The Caddyfile parameterizes the reverse proxying between the external subdomains and the Docker's internal ports. Here is the `Caddyfile` pasted for convenience ...

```
# docker exec caddy caddy reload --config /etc/caddy/Caddyfile
{
    order pmtiles_proxy before file_server
    # jwtauth (ggicci/caddy-jwt, built into caddy/Dockerfile) must run before any
    # handler on the vhosts that use it -- see preview.marinesensitivity.org
    order jwtauth before basic_auth
}

(cors) {
    @origin header Origin *
    header @origin {
        Access-Control-Allow-Origin "*"
        Access-Control-Request-Method GET
    }
}

# CORS for the storage.* redirect to S3 (2026-10-08): the 302 itself must carry
# the headers, and OPTIONS must be answered here, because a browser preflights
# before it ever follows the redirect. Not (cors): that one only answers when
# the request has an Origin and adds no Expose/Methods/Max-Age.
(s3_cors) {
    header {
        Access-Control-Allow-Origin "*"
        Access-Control-Allow-Methods "GET, HEAD"
        Access-Control-Expose-Headers "Content-Range, Content-Length, Accept-Ranges, ETag"
        Access-Control-Max-Age "3600"
    }
}

# Every vhost logs (2026-10-02 egress review: only storage/preview/file did, so
# nobody could say who was crawling the other ~17). Every byte served is EC2
# egress, so the log is what turns a surprise bill into a name.
#   import access_log <name>   -> /share/logs/caddy/<name>.log (JSON, rotated)
(access_log) {
    log {
        output file /share/logs/caddy/{args[0]}.log {
            roll_size 64MiB
            roll_keep 14
            roll_keep_for 2160h
        }
        format json
    }
```

```
}

# Crawler policy for the DATA hosts (tiles, COGs, API, files). These are used by
# browsers through the apps, never meant to be walked: tell well-behaved crawlers
# to stay out, and refuse the AI/SEO scrapers outright.
(robots_none) {
    handle /robots.txt {
        header Content-Type "text/plain; charset=utf-8"
        respond `User-agent: *
Disallow: /
` 200
    }
}

# 403 for AI-training / SEO scrapers by User-Agent (case-insensitive substring).
# Deliberately NOT listed: Claude-User, ChatGPT-User -- a person asking an
# assistant to fetch one URL is wanted. Arguments are the paths exempt from the
# block (always /robots.txt; the file host also exempts its small browsable
# trees). This is a `handle`, not a bare `respond`, so that it also runs on a
# vhost made of sibling `handle` blocks (file), where a respond would sort AFTER
# them and never fire.
(no_ai_bots) {
    @ai_bots {
        header_regexp ai_bot User-Agent (?i)(GPTBot|OAI-SearchBot|ClaudeBot|Claude-SearchBot
        not path {args[:]}
    }
    handle @ai_bots {
        header Content-Type "text/plain; charset=utf-8"
        respond "Automated crawling of these data hosts is not permitted. See https://marines
    }
}

api.marinesensitivity.org {
    import access_log api
    import robots_none
    import no_ai_bots /robots.txt

    reverse_proxy plumber:8888
}

file.marinesensitivity.org {
    import cors
```

```
    # This host had no access log and no robots.txt while it listed ~80 GB of
    # data as browsable directories (2026-10-02 egress review). Every byte it
    # serves is VM egress, so: log it, tell crawlers to stay out of the data,
    # and serve files by exact URL only -- no directory listings on the data
    # trees. The browsable front door for data is the S3 index at
    # storage.marinesensitivity.org; /stac/ and /branding/ stay browsable (small).
    import access_log file
    import no_ai_bots /robots.txt /stac/* /branding/*

    handle /robots.txt {
        header Content-Type "text/plain; charset=utf-8"
        respond `# The STAC catalog and the brand set: crawl away. Data files: please do not
User-agent: *
Crawl-delay: 10
Allow: /stac/
Allow: /branding/
Disallow: /
` 200
    }

    # the brand set, served straight from the repo checkout: a `git pull` on the
    # server publishes it, so there is no copy step to forget. Used by the
    # Cloudflare Access login page (which takes a logo URL), the docs, the apps
    # and any deck. See server/branding/README.md.
    handle_path /branding/* {
        root * /share/github/MarineSensitivity/server/branding
        header Cache-Control "public, max-age=3600"
        file_server browse
    }

    # raw pmtiles for client-side PMTiles protocol (add_pmtiles_source)
    handle_path /pmtiles/* {
        root * /share/data/derived/pmtiles
        file_server
    }

    # The per-model PMTiles of v8 and v9 were served from /share/data/pmtiles_native
    # (moved out of derived/ on 2026-10-02 after the BOEM mirror's rclone loop re-read
    # them hourly, ~150 GB/day) until the asset-store prune of 2026-10-07 removed them:
    # every published v8/v9 `native_asset` row names a store object on S3
    # (native/{ds_key}/{hash}.pmtiles). The old URLs answer 410 here rather than
    # falling through to /share/data/derived/pmtiles, so a stray copy in the old
```

```
    # place can never be served again (caddy/test/file_host_local.sh proves it).
    handle /pmtiles/v8/* {
        respond "Gone: per-model PMTiles are served from the asset store; see the release's r
    }
    handle /pmtiles/v9/* {
        respond "Gone: per-model PMTiles are served from the asset store; see the release's r
    }

    # legacy versioned pmtiles routes (backwards compat for old apps)
    handle_path /pmtiles/v4/* {
        root * /share/data/derived/v4/pmtiles
        file_server
    }
    handle_path /pmtiles/v3/* {
        root * /share/data/derived/v3/pmtiles
        file_server
    }

    # stac catalog (sdm extension) - static JSON tree
    handle_path /stac/* {
        root * /share/data/derived/stac
        file_server browse
    }

    # versioned derived data (geoparquet, cogs, gpkg) referenced by stac assets
    handle_path /derived/* {
        root * /share/data/derived
        file_server
    }

    # force reports to download (especially HTML which otherwise renders inline)
    handle_path /reports/* {
        root * /share/public/reports
        header Content-Disposition "attachment"
        file_server
    }

    handle {
        root * /share/public
        file_server {
            precompressed zstd br gzip
        }
```

```
    }
}

# Browsable front door for the PUBLIC S3 bucket -- folder PAGES only.
#
# Why this exists: S3 serves OBJECTS, not directories. A URL ending in "/" 404s
# unless an object literally has that key, and anonymous ListBucket is DENIED on
# this bucket (403 AccessDenied), so a browser cannot enumerate anything. The
# workflows notebook publish_storage_index.qmd generates a real index.html per
# directory; this vhost rewrites folder URLs to them:
#
#   storage.marinesensitivity.org/marine-atlas/v8/  -> .../v8/index.html  (proxied)
#   storage.marinesensitivity.org/marine-atlas/latest.txt -> 302 to the object
#
# ONLY the small pages transit this server: folder URLs, */index.html and
# */README.md. EVERY OTHER object is a 302 to the bucket's own HTTPS endpoint,
# so none of its bytes -- Range requests included, a redirect ignores Range --
# is billed as VM egress (it used to be proxied: 18 GB across ~90,000 objects,
# 2026-10-02 egress review). The index pages already link files at that endpoint
# directly. 302, not 301, because the store may move to another provider and a
# cached 301 would pin clients to the old one.
#
# The redirect carries CORS (2026-10-08): the apps' pmtiles / COG / Parquet range
# requests to this host failed with status 0 because the bare 302 (and a bare 302
# on the OPTIONS preflight) had no Access-Control-* headers. The 302 now has
# them (snippet s3_cors) and OPTIONS is a 204 here; the bucket itself answers
# with Access-Control-Allow-Origin: * once the browser lands on it.
#
# Named "storage" rather than "s3" so the URL survives a move to another
# provider.
# storage.oceanmetrics.io is the same bucket under the Ocean Metrics brand (the
# gazetteer/ prefix is published there first; marine-atlas/ keeps its old links).
storage.marinesensitivity.org, storage.oceanmetrics.io {
    encode zstd gzip

    import access_log storage

    # robots.txt inline, BEFORE the bucket handler, so it cannot be lost by an
    # index regeneration. The catalog pages are welcome in search results; the
    # objects behind them are not (they are now redirected to S3, but a crawler
    # that follows the redirect still walks 90,000 of them).
    handle /robots.txt {
```

```
        header Content-Type "text/plain; charset=utf-8"
        respond `# Catalog pages: crawl away. Data objects: please do not.
User-agent: *
Crawl-delay: 10
Disallow: /*.tif$
Disallow: /*.parquet$
Disallow: /*.duckdb$
Disallow: /*.pmtiles$
Disallow: /*.gz$
Disallow: /*.zip$
` 200
    }

    # ONLY the published atlas. The same bucket holds backups/ -- including a
    # multi-GB database dump -- which is technically public but must not be
    # advertised or reachable here, even by exact path. This allow-list is the
    # server-side half of the restriction the index generator applies.
    @atlas path_regexp atlas ^/(marine-atlas|gazetteer)(/|$)
    handle @atlas {
        # CORS preflight, answered here (a redirect on OPTIONS fails the preflight).
        # Suffix ranges (bytes=-N, how Parquet readers fetch the footer) and
        # If-Match / If-None-Match are not CORS-safelisted, so they DO preflight.
        @preflight method OPTIONS
        handle @preflight {
            import s3_cors
            header Access-Control-Allow-Headers "Range, If-Match, If-None-Match, If-Range, C
            respond 204
        }

        # folder URL -> the index.html object standing in for a listing
        @dir path */
        handle @dir {
            rewrite * {path}index.html
            reverse_proxy https://s3.us-east-1.amazonaws.com {
                header_up Host s3.us-east-1.amazonaws.com
                rewrite /oceanmetrics.io-public{uri}
            }
        }

        # the pages themselves: small, and what a browser needs on this host
        @page path */index.html */README.md
        handle @page {
```

```
            reverse_proxy https://s3.us-east-1.amazonaws.com {
                header_up Host s3.us-east-1.amazonaws.com
                rewrite /oceanmetrics.io-public{uri}
            }
        }

        # everything else is a data object: hand the client to S3, move no bytes
        handle {
            import s3_cors
            header Cache-Control "public, max-age=86400"
            redir https://s3.us-east-1.amazonaws.com/oceanmetrics.io-public{uri} 302
        }
    }

    # the bare host lands on the atlas index
    redir / /marine-atlas/ 302

    # anything else: refuse, and say where to go
    handle {
        header Content-Type "text/plain; charset=utf-8"
        respond `Not found.

Browsable: /marine-atlas/ and /gazetteer/
Start here: https://storage.marinesensitivity.org/marine-atlas/
` 404
    }
}

pgadmin.marinesensitivity.org {
    import access_log pgadmin

    reverse_proxy pgadmin:8088
}

# rest.marinesensitivity.org {
#   reverse_proxy rest:3000
# }

# searchable STAC API (one Item per model) - complements the STATIC catalog served
# from file.marinesensitivity.org/stac, which stays exactly as it is.
stac-api.marinesensitivity.org {
    import access_log stac-api
```

```
    import robots_none

    reverse_proxy stac-api:8084
}

rstudio.marinesensitivity.org {
    import access_log rstudio

    reverse_proxy rstudio:8787
}

shiny.marinesensitivity.org {
    import access_log shiny

    reverse_proxy rstudio:3838
}

app.marinesensitivity.org {
    import access_log app

    # ---- MST app cutover ------------------------------------------------
    #
    # /scores and /species are now ONE app that renders any published release
    # from ?ver=. Every former per-version instance redirects to it rather than
    # 404ing, so old links, bookmarks and anything in a published report keep
    # resolving - to the SAME release they always showed.
    #
    # The v7 apps were `mapgl` (scores) and `mapsp` (species); v1-v6 were
    # `mapgl_v{n}` / `mapsp_v{n}`; v6 and v8 also had `scores_v{n}` /
    # `species_v{n}` aliases. All of them land on /scores or /species with the
    # version they represent.
    #
    # 301 (permanent): these instances are not coming back, and a permanent
    # redirect lets browsers and crawlers stop asking.
    #
    # `&{query}` IS THE POINT OF THE REDIRECT. A redir target that carries its own
    # query string REPLACES the incoming one, so these rules used to answer
    # /mapsp/?mdl_seq=1434 with /species/?ver=v7 -- dropping the very thing the
    # link was about. The app then opened on its default taxon, which is why every
    # deep link in the final report, and the ones in issue #5, silently showed the
    # leatherback turtle. Every published per-species link is of this form, so a
    # redirect that keeps the path and loses the id is worse than a 404: it looks
```

```
    # like it worked (MarineSensitivity/apps#6).
    #
    # An empty {query} leaves a harmless trailing "&", which parses to nothing.
    @scores_ver path_regexp scv ^/(?:mapgl|scores)_(v[0-9]+[a-z]?)(?:/(.*))?$
    redir @scores_ver /{re.scv.1}/scores/?{query} 301

    @species_ver path_regexp spv ^/(?:mapsp|species)_(v[0-9]+[a-z]?)(?:/(.*))?$
    redir @species_ver /{re.spv.1}/species/?{query} 301

    # the unsuffixed v7 instances: `mapgl`/`mapsp` were what /scores and
    # /species pointed at, so they carry v7 explicitly
    @scores_v7 path_regexp sc7 ^/mapgl(?:/(.*))?$
    redir @scores_v7 /v7/scores/?{query} 301

    @species_v7 path_regexp sp7 ^/mapsp(?:/(.*))?$
    redir @species_v7 /v7/species/?{query} 301

    # A RESTRICTED release asked for on the PUBLIC host goes to the review host
    # (2026-08-28). The app process here cannot resolve a restricted version
    # (msens::atlas_allow_access), so /v8/scores/ used to render the promoted
    # release with no hint -- fail-closed, but from a published link it simply
    # looked like the wrong version (the homepage's v8 links "oddly ended up on
    # v7"). Every spelling funnels through the path form first -- the retirement
    # 301s above, the ?ver= 301s in app_version_routes.caddy -- so this ONE
    # matcher catches them all and {uri} carries the query (deep links keep
    # their mdl_key). 302, not 301: a version stops being restricted when it is
    # released, and a cached 301 would keep sending visitors to the review host.
    #
    # `atlas` joined this matcher 2026-09-24 (atlas-9): the static atlas app has
    # no canonical URL on THIS host (its public home is marinesensitivity.org/
    # atlas/?ver=, GitHub Pages -- a different origin this Caddyfile does not
    # serve at all), so nothing here would otherwise answer /{ver}/atlas/ except
    # falling through to `reverse_proxy rstudio:3838`, which does not know that
    # path. Covering it keeps the same guarantee scores/species already have --
    # a reviewer who guesses the /v9/atlas/ shape by analogy lands on the review
    # host, never a 404 that looks like "this version doesn't have an atlas" --
    # and it is what sends a restricted version's atlas request to the ONE place
    # that can actually serve it (preview_routes.caddy's atlas import).
    #
    # PREVIEW_RESTRICTED_VERSIONS is a regex alternation of the restricted
    # versions ("v8|v9") from the server .env -- the same manual step as
    # PREVIEW_REVIEWERS_<VER> and CF_ACCESS_AUD -- and DEPLOY_CADDY asserts it
```

```
    # against versions.json before restarting. Unset, it matches nothing and the
    # prior behaviour (fall back to the promoted release) stands.
    @restricted_app path_regexp rapp ^/(?:{$PREVIEW_RESTRICTED_VERSIONS:__none__})/(?:scores
    redir @restricted_app https://preview.marinesensitivity.org{uri} 302

    # THE VERSION IS THE PATH here too (2026-08-27). One definition, shared with
    # the preview vhost, so the two hosts cannot drift: /v7/scores/ is canonical,
    # /scores/?ver=v7 301s to it, and the app reads X-MS-Version. A bare /scores/
    # (no version) still resolves the promoted release inside the app -- Caddy
    # cannot read latest.txt, and the app canonicalises the URL once it knows.
    import app_version_routes.caddy rstudio:3838

    # X-MS-User (the signed-in reviewer's identity) and X-MS-Version (the version
    # from the preview host's URL path) are set by the PREVIEW vhost from things
    # it has verified. Both are meaningful only there; a client must not be able
    # to supply either here, where the version comes from ?ver= and the policy
    # refuses restricted releases anyway.
    request_header -X-MS-User
    request_header -X-MS-Version

    reverse_proxy rstudio:3838
}

# The signed-in PREVIEW host: restricted (under-review) releases of the apps and
# docs, for SDM data providers and BOEM/NOAA reviewers.
#
# How the gate works (workflows/.claude/plans/2026-08-15 pre-release review gate …):
#   1. DNS for this ONE hostname is proxied through Cloudflare, and a Cloudflare
#      Access application in front of it lets a request through only after the
#      visitor signs in (emailed one-time PIN, Google). Access then stamps every
#      request -- page GETs, static assets, the Shiny websocket upgrade -- with
#      Cf-Access-Jwt-Assertion, an RS256 JWT for this app's AUD.
#   2. jwtauth below is the ORIGIN-SIDE check Cloudflare documents: verify that
#      JWT against the team's JWKS + issuer + audience. A request that reaches
#      this box without going through Access (origin IP, wrong hostname, a
#      forged header) has no valid JWT and gets 401 -- so proxying only this
#      hostname is safe.
#   3. What is behind it (caddy/preview_routes.caddy): the PREVIEW Shiny Server
#      instance (rstudio:3839, the same app code as /scores + /species with
#      MS_PREVIEW=1 -- see rstudio/shiny-server.conf) at /{ver}/scores/ and
#      /{ver}/species/ -- THE VERSION IS THE PATH, so Access can hold one
#      reviewer policy per version -- the rendered books of restricted docs
```

```
#      versions at /docs/{ver}/ (docs CI publishes those to the gh-pages-preview
#      branch, cloned at /share/docs_preview by the docs-preview sidecar), and a
#      landing page.
#
# CF_ACCESS_TEAM / CF_ACCESS_AUD come from the server .env. Their DEFAULT is a
# value nothing can match, so an unconfigured deploy fails CLOSED (401 for
# everyone) rather than open. Which versions are restricted is NOT known here:
# that is `access` in versions.json, enforced by the app process's MS_PREVIEW.
preview.marinesensitivity.org {
    import access_log preview

    jwtauth {
        jwk_url https://{$CF_ACCESS_TEAM:unconfigured}.cloudflareaccess.com/cdn-cgi/access/c
        from_header Cf-Access-Jwt-Assertion
        from_cookies CF_Authorization
        issuer_whitelist https://{$CF_ACCESS_TEAM:unconfigured}.cloudflareaccess.com
        audience_whitelist {$CF_ACCESS_AUD:unconfigured}
        # A HUMAN's Access JWT carries `email`; a SERVICE TOKEN's carries
        # `common_name` (its client id) and no email at all. caddy-jwt needs to
        # find one of these or it rejects the request -- which is how the release
        # notebook's own check token got a 401 from the ORIGIN after Cloudflare
        # had already admitted it. (Cloudflare rewrites `server:` on proxied
        # responses, so `server: cloudflare` on that 401 was not evidence that
        # Cloudflare produced it; the body was this vhost's own error page.)
        user_claims email common_name sub
    }

    # the routes themselves live in a file the functional test can import too
    # (caddy/test/run.sh) -- see it for the URL scheme: the version is the PATH
    import /etc/caddy/preview_routes.caddy

    # jwtauth answers a bare 401 with an EMPTY body -- a blank page in a browser,
    # indistinguishable from "broken". Say what this host is instead. (Cloudflare
    # Access intercepts before the origin, so once it is in front a human never
    # sees this; it is what a request that BYPASSED Access gets.)
    handle_errors {
        @unauth expression {http.error.status_code} == 401
        handle @unauth {
            header Content-Type "text/html; charset=utf-8"
            respond <<HTML
                <!doctype html><meta charset="utf-8"><meta name="robots" content="noindex">
                <title>Marine Sensitivity - preview (sign-in required)</title>
```

```
                <style>body{font:16px/1.5 system-ui,sans-serif;max-width:40rem;margin:4rem au
                <h1>Marine Sensitivity - preview</h1>
                <p>This host serves <b>restricted, pre-release</b> versions of the apps and d
                to invited reviewers, and requires sign-in through the project's access gatew
                which is not yet in front of this address, so nothing here can be viewed at t
                <p>Public releases: <a href="https://app.marinesensitivity.org/scores/">Score
                <a href="https://app.marinesensitivity.org/species/">Species</a> ·
                <a href="https://marinesensitivity.org/docs/">Documentation</a>.</p>
                HTML 401
        }
    }
}

shiny.oceanmetrics.io {
    import access_log shiny-oceanmetrics

    reverse_proxy rstudio:3838
}

# swagger.marinesensitivity.org {
#   reverse_proxy swagger:8080
# }

tile.marinesensitivity.org {
    import access_log tile
    import robots_none
    import no_ai_bots /robots.txt

    reverse_proxy tile:7800
}

tilecache.marinesensitivity.org {
    import access_log tilecache
    import robots_none
    import no_ai_bots /robots.txt

    reverse_proxy tilecache:6081
}

titiler.marinesensitivity.org {
    import access_log titiler
    import robots_none
```

```
    import no_ai_bots /robots.txt

    reverse_proxy titiler:8000
}

titiler-v8.marinesensitivity.org {
    import access_log titiler-v8
    import robots_none
    import no_ai_bots /robots.txt

    reverse_proxy titiler-v8:8000
}

titilecache.marinesensitivity.org {
    import access_log titilecache
    import robots_none
    import no_ai_bots /robots.txt

    reverse_proxy titilecache:6082
}

h3t.marinesensitivity.org {
    import access_log h3t
    import robots_none
    import no_ai_bots /robots.txt

    reverse_proxy h3t:8889
}

h3tcache.marinesensitivity.org {
    import access_log h3tcache
    import robots_none
    import no_ai_bots /robots.txt

    reverse_proxy h3tcache:6083
}

pmtiles.marinesensitivity.org {
    import access_log pmtiles
    import cors
    import robots_none
    import no_ai_bots /robots.txt
```

```
    # server-side tile decoding: ZXY + TileJSON (for QGIS, leaflet, etc.)
    handle_path /tiles/* {
        pmtiles_proxy {
            bucket file:///share/data/derived/v4/pmtiles
            # note: v3 pmtiles also available at /share/data/derived/v3/pmtiles
            cache_size 256
            public_url https://pmtiles.marinesensitivity.org/tiles
        }
    }
}

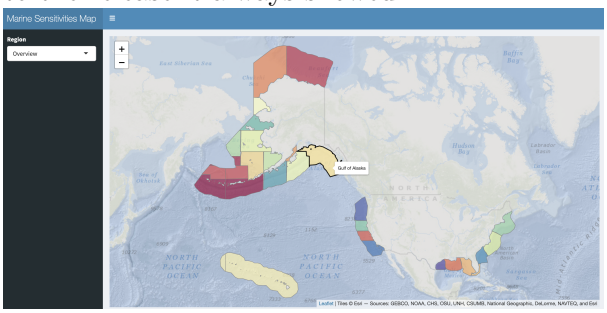
# erddap.oceanmetrics.io: the vhost lives in its own repo (oceanmetrics/erddap,
# checked out under /share/github, which this container mounts); the erddap
# compose joins this stack's `server_default` network so `erddap:8080` resolves.
import /share/github/oceanmetrics/erddap/caddy/Caddyfile.erddap
```

14.5 Services

The server is running the following services. Subdomains with are internal (admin / development); others are public.

14.5.1 App + data plane

- **app** / **shiny**
  interactive Shiny applications — proxied to the `rstudio` container's Shiny Server on internal port 3838. There are now **two** applications, `/scores` and `/species`, each rendering any published release from the URL path (e.g. `/v8/scores/`). The former per-version instances (`mapgl`, `mapsp`, `mapgl_v1`…`mapsp_v6`, `scores_v6`/`v8`, `species_v6`/`v8`) are retired, with Caddy issuing a **301 to the equivalent path** so every existing link — including any printed in a published report or an older `?ver=` deep link — still resolves to the release it always showed.



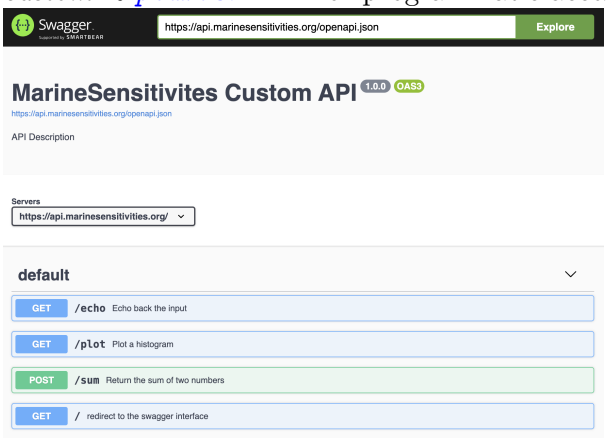
Shiny Server docs

- **file**
  static file server for PMTiles archives, report downloads, reference GeoPackages — served by Caddy's `file_server` directive directly from the `/share` bind-mount. This replaces the old `pg_tileserv` vector-tile path for Program Areas, Ecoregions, Planning Areas, protractions, blocks and aliquots; each layer is a single `.pmtiles` file that the browser range-reads for only the tiles it needs. It no longer hosts the per-species range tiles (the `pmtiles/{v8,v9}/` copies): from the asset-store migration (2026-10) those are served from the S3 store (`native/{ds_key}/{hash}.pmtiles`, Section 15.3), which answers range requests with CORS. The file host keeps serving the zone and reference layers, report downloads and reference GeoPackages, and the species copies are retained there until the prune that follows the migration.

- **titiler** + **titiler-v8**
  TiTiler serving the published raster assets.
  Every map layer the applications draw is now a **pre-rendered Cloud-Optimized GeoTIFF** read through titiler's stock `/cog` routes — score surfaces per metric × subregion, and per-model distributions. `titiler-v8` runs in parallel with the v7 instance, reading a kilobyte-sized view-only DuckDB whose `model_cell` is a *view* over the partitioned Parquet on S3, so no multi-gigabyte database is copied to the server.

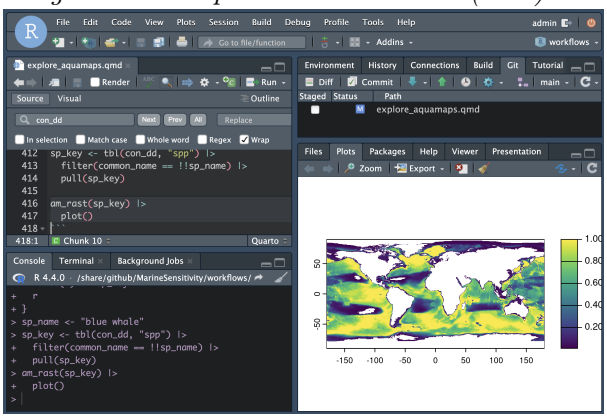
- **titilecache** (Varnish)
  Fronts the v7 `titiler` only. The v8 service is deliberately **not** proxied through it: an immutable COG is already cacheable by URL, and the one endpoint that would benefit is retired (below).

- **api**
  custom R `plumber` API for programmatic access to DuckDB — see Chapter 17



14.5.2 Admin

- **rstudio**
  integrated development environment (IDE) to code and debug directly on the server.



Posit RStudio Server

- **pgadmin** — *PostgreSQL administration interface*, commented out along with the PostgreSQL services it managed (see Retired, below).

14.5.3 Retired

- **The custom `/msens` tile factory** — kept in the image but **off by default**, behind `MSENS_FACTORY=1`.
  It rendered a tile by executing a validated SELECT against DuckDB and looking the result up against a cell-id COG. Pre-rendered COGs replaced it: nothing reaches it any more, because every metric on every release now has a COG. The two paths were verified equivalent before the switch — 100 of 105 sample tiles byte-identical, the remainder differing by 3–6 pixels out of 262,144 from float32 storage. It remains available for exploratory on-the-fly queries.

- **postgis**, **tile** (`pg_tileserv`), **tilecache**, **rest** (PostgREST) and **swagger** — all **commented out** in `docker-compose.yml` and **not running**. PostgreSQL is no longer part of any serving path; the authoritative store is DuckDB, with the published release as partitioned Parquet on S3. Vector geometry is served as static PMTiles by Caddy's `file` subdomain, and the `pgadmin` entry above is likewise vestigial.

•••

15 Database

The **source of truth** for the Marine Sensitivity Toolkit is DuckDB — a columnar, embedded analytic database. It is fast enough to answer live per-cell queries while remaining a single-file artifact that can be copied, versioned and served read-only to many processes at once.

The pipeline writes a working database (`sdm.duckdb`) on the machine that runs it; a **release** is the published, immutable form of that database's tables. Since the marine-atlas release format, each version is published as **partitioned Parquet on S3** under `marine-atlas/{ver}/tables/`, and serving reads it through a tiny view-only DuckDB (`serve.duckdb`, kilobytes) whose tables are *views* over those files — never a multi-gigabyte copy.

Everything below is read from the release this build documents (**v3**), so the tables, columns and row counts are those of that version rather than of whichever version happened to be current when the page was written.

> **Note**
>
> **Why DuckDB and not PostgreSQL?** Earlier versions of the stack used PostgreSQL + PostGIS as the primary store. As the analysis stabilized around a fixed 0.05° cell grid and a small set of zone layers, live geometry operations and concurrent writes stopped being needed — only fast analytic reads. DuckDB scans tens of millions of rows in well under a second with no separate server process. PostGIS is no longer part of the serving path (see Chapter 14).

15.1 Table and column naming conventions

- Table names are singular and lower case.
- Unique identifier columns are suffixed with:
  - `*_id` for unique integer keys (stable across rebuilds);
  - `*_key` for unique string keys (human-readable and stable across releases);
  - `*_seq` for auto-incrementing sequence integer keys (internal join keys, **not** stable across rebuilds).
- Column names are singular and use `snake_case`.

- Foreign keys use the singular form of the referenced table, followed by the primary-key suffix (`_id`, `_key`, or `_seq`).

> **!** `value` is a reserved word in DuckDB
>
> Cell values are stored in a column named `val`, not `value`. The serving views expose `val AS value` for backward compatibility with callers written against the older schema.

Table 15.1: Tables published by v3, with row counts read from the release itself.

Table	Rows	Purpose
`cell`	661,372	Grid cells of the study area, with environmental covariates and membership flags.
`taxon`	17,561	One row per taxon: identity, category, extinction risk and validity flags.
`model`	30,430	Registry of every distribution model, keyed by the stable model identifier.
`metric`	35	Metric registry: key and description for each scored quantity.
`cell_metric`	21,990,083	Per-cell score for each metric — the surface the maps draw.
`zone`	76	Zone registry keyed by source table, field and value, so every spatial unit shares one schema.
`zone_cell`	4,626,531	Zone membership: which cells belong to which zone.
`zone_metric`	774	Pre-aggregated per-zone score for each metric — what drives the choropleth.
`zone_taxon`	61,219	Per-zone species list with each species' contribution to the zone score.
`dataset`	8	Source datasets, with citations, vintages and whether each fed the scores.
`taxon_model`	29,805	Which models fed which taxon — the taxon to model relation.
`listing`	718	US federal listing lookup (ESA status, MMPA and MBTA protection).

> **i** Not published by v3
>
> `model_asset`, `native_asset`. Sections that depend on these are omitted from this build rather than described from another release.

15.2 Schema

The columns of each published table, read from the release:

15.2.1 `cell`

Column	Type
cell_id	INTEGER
depth_mean	DOUBLE
depth_min	DOUBLE
depth_max	DOUBLE
oxy_b_mean	DOUBLE
oxy_mean	DOUBLE
prim_prod_mean	DOUBLE
ice_con_ann	DOUBLE
salinity_b_mean	DOUBLE
salinity_mean	DOUBLE
sbt_an_mean	DOUBLE
sst_an_mean	DOUBLE
fao_area_m	DOUBLE
area_km2	DOUBLE
width_km2	DOUBLE

15.2.2 `taxon`

Column	Type
taxon_id	DOUBLE
taxon_authority	VARCHAR
n_ds	INTEGER
ms_merge	INTEGER
sp_cat	VARCHAR
mdl_seq	INTEGER
scientific_name	VARCHAR
common_name	VARCHAR
worms_id	INTEGER
redlist_code	VARCHAR
extrisk_code	VARCHAR
er_score	INTEGER
is_mmpa	BOOLEAN
is_mbta	BOOLEAN
is_bcc	BOOLEAN
worms_is_marine	BOOLEAN
worms_is_extinct	BOOLEAN
esa_code	VARCHAR

Column	Type
esa_source	VARCHAR
is_ok	BOOLEAN
worms_taxonomic_status	VARCHAR

15.2.3 `model`

Column	Type
mdl_seq	INTEGER
ds_key	VARCHAR
taxa	VARCHAR
time_period	VARCHAR
region	VARCHAR
mdl_type	VARCHAR
description	VARCHAR
date_created	DATE

15.2.4 `metric`

Column	Type
metric_seq	INTEGER
metric_key	VARCHAR
description	VARCHAR
date_created	DATE
metric_title	VARCHAR
metric_abbrev	VARCHAR

15.2.5 `cell_metric`

Column	Type
cell_id	INTEGER
metric_seq	INTEGER
value	DOUBLE

15.2.6 `zone`

Column	Type
zone_seq	INTEGER
tbl	VARCHAR
fld	VARCHAR
value	VARCHAR
date_created	DATE

15.2.7 `zone_cell`

Column	Type
zone_seq	INTEGER
cell_id	INTEGER
pct_covered	INTEGER

15.2.8 `zone_metric`

Column	Type
zone_seq	INTEGER
metric_seq	INTEGER
value	DOUBLE

15.2.9 `zone_taxon`

Column	Type
zone_tbl	VARCHAR
zone_fld	VARCHAR
zone_value	VARCHAR
mdl_seq	INTEGER
sp_cat	VARCHAR
sp_common	VARCHAR
sp_scientific	VARCHAR
taxon_id	DOUBLE
taxon_authority	VARCHAR

Column	Type
rl_code	VARCHAR
er_score	INTEGER
is_mmpa	BOOLEAN
is_mbta	BOOLEAN
is_bcc	BOOLEAN
esa_code	VARCHAR
esa_source	VARCHAR
area_km2	DOUBLE
avg_suit	DOUBLE
suit_rl	DOUBLE
suit_rl_area	DOUBLE
cat_suit_rl_area	DOUBLE
pct_cat	DOUBLE

15.2.10 `dataset`

Column	Type
ds_key	VARCHAR
name_short	VARCHAR
name_original	VARCHAR
description	VARCHAR
citation	VARCHAR
source_broad	VARCHAR
source_detail	VARCHAR
regions	VARCHAR
response_type	VARCHAR
taxa_groups	VARCHAR
year_pub	INTEGER
date_obs_beg	DATE
date_obs_end	DATE
date_env_beg	DATE
date_env_end	DATE
link_info	VARCHAR
link_download	VARCHAR
link_metadata	VARCHAR
links_other	VARCHAR
spatial_res_deg	DOUBLE
temporal_res	VARCHAR
date_created	DATE

Column	Type
name_display	VARCHAR
value_info	VARCHAR
is_mask	BOOLEAN
sort_order	INTEGER

15.2.11 `taxon_model`

Column	Type
taxon_id	DOUBLE
ds_key	VARCHAR
mdl_seq	INTEGER

15.2.12 `listing`

Column	Type
spp_id	DOUBLE
worms_id	DOUBLE
botw_id	DOUBLE
extrisk_code	VARCHAR
er_score	INTEGER
is_mmpa	BOOLEAN
is_mbta	BOOLEAN
is_bcc	BOOLEAN
common_name	VARCHAR

15.3 Asset store

i Old copies remain until the prune

Every published release (v7, v7b, v8, v9) has pointed only into the store since the **asset-store migration of 2026-10-02**. The copies the v8 and v9 releases carried before it still exist under their release prefixes (`v8/native/`, `v9/native/`) and on the file host, unreferenced, until a prune that follows a soak period. The migration's listing on 2026-10-01 found 145,452 objects (33.08 GB) under those two prefixes, which collapse to 86,421 distinct keys (19.63 GB) in the store, because the two releases carry the same files for

most datasets.

A release owns pointers, never distribution files. Every cloud-optimized distribution file lives exactly once in an unversioned, content-addressed store, and each release publishes tables whose rows point into it. Nothing ending in `.tif` or `.pmtiles` is written under a release's `{ver}/` prefix. A release prefix holds only its manifest, its tables, its app bundle and its serving files.

There are two stores, both in the project's public S3 bucket:

Store	Key	Holds
`cog/`	`cog/{grid_id}/{hash}.tif`	rasters *on an analysis grid*: each input's gridded surface, merged models and score surfaces
`native/`	`native/{ds_key}/{hash}.{tif|pmtiles}`	each dataset's own resolution original: a raster (`.tif`) or a vector range (`.pmtiles`)

15.3.1 The key comes from the content

The `hash` in a key is computed **before** any file is built, from the *source content* of the model plus an **encoding tag**, and never from the bytes of the file or from the release:

- for a gridded surface, the model's `(cell_id, val)` rows;
- for a native raster, the source grid's `(cell_id, val)` rows;
- for a native vector range, the normalized geometry and the attributes that reach the tile;
- the encoding tag names how the file is written (data type, nodata, overviews, tile scheme), so a different encoding of the same content is deliberately a different object.

Two consequences follow. A surface that did not change between releases is found in the store by its key, so it costs neither a build nor an upload; a source dataset such as AquaMaps that is unchanged between releases is registered by a new pointer row and nothing else. And a merged model that equals its only input has the same rows, hence the same key, hence is the same object as that input, with no special case. A re-ingested dataset adds objects only for the models whose content changed; the old objects stay for the releases that still point at them. Scores are always new, because they are per-release tables.

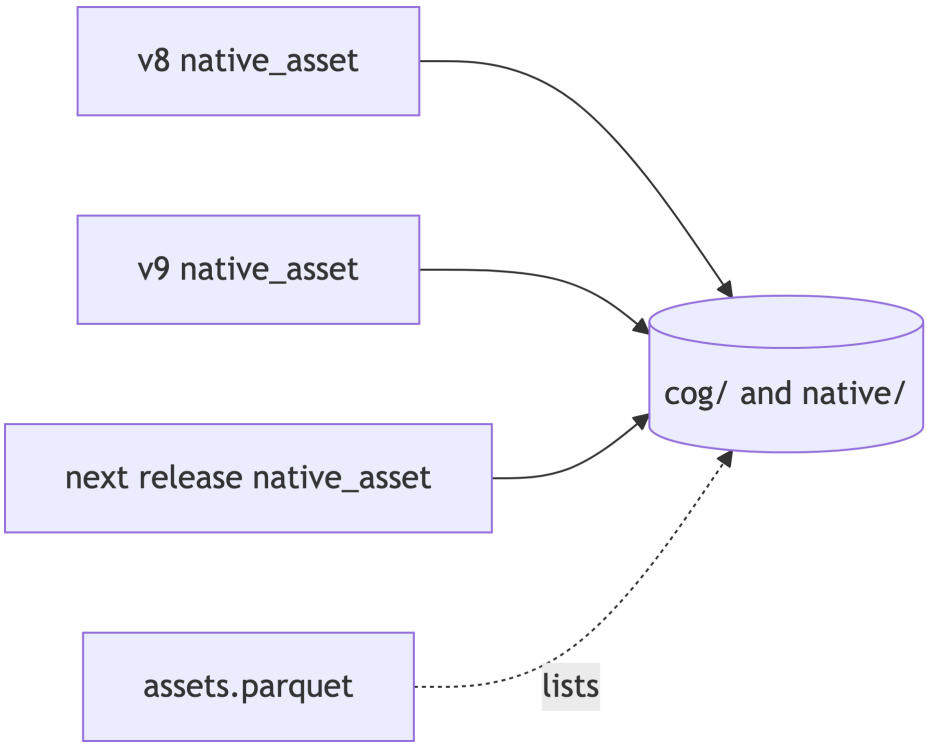


Figure 15.1: A release's pointer table refers to objects in the shared stores; the catalog lists every stored object.

15.3.2 The catalog and the pointer table

- `assets.parquet` at the atlas root (beside the `cog/` and `native/` stores, under the same `marine-atlas/` prefix as the releases) is the catalog, with one row per stored object (store, key, content hash, encoding tag, asset type, grid or dataset key, size, MD5, creation time and the first release that used it). "Is it already stored?" is an anti-join against this file, not a bucket listing.
- `native_asset` is the per-release pointer table (Table 15.1). Its `representation = model` rows point into `cog/`, its `native` rows point into `native/`, and a merged model is a row with `ds_key = ms_merge`. Each row carries the content hash and the asset URL; a PMTiles row also carries `source_key`, the model key the tile's features are tagged with, which is how a viewer filters one range out of the archive. Every published release has one in this shape; the earlier releases keep `model_asset` beside it so that old links still resolve.
- An object no release points at is garbage. Nothing is deleted except through a check that joins the catalog against every release's pointer tables.

15.3.3 Finding a file

The bucket denies anonymous listing, so a file is found through a pointer, not by browsing:

1. read the catalog `assets.parquet` to see what exists and where;
2. read a release's `native_asset` table (`{ver}/tables/native_asset.parquet`) to see which objects that release uses for each model;
3. or ask STAC: the searchable STAC API is built from the pointer table, so each Item's `model` and `native` assets carry the same store URLs, and the static STAC catalog of a release links them once its tree is rebuilt from the pointer table (every published release today).

15.3.4 Verify before store

An object is admitted to the store under key K only if its decoded pixels equal the rows K was computed from. In the migration, a file that failed the check was repainted from the model's current rows and uploaded under its key (2,934 of the raster objects; the rest were copied once their pixels had been verified). A publish applies the same check to every new file before it is uploaded. This keeps what a map draws identical to what the scores used.

See Section 16.3 for how a publish writes to the store and what the publish gate checks.

15.4 How the tables feed the serving tier

- **Score rasters** — per-cell scores are published as **Cloud-Optimized GeoTIFFs**, one per metric × subregion, and the app reads the href and rescale range straight from the release manifest. Earlier releases rendered them by sending SQL to a tile factory per request; that path is retired (see Chapter 14).
- **Per-model distributions** — `model_cell` is published partitioned by a dense integer model id, so a tile request reads exactly one partition as a point read rather than scanning the surface.
- **Zone choropleths** — `zone_metric` joined to `zone` gives the per-zone score; the geometry comes from PMTiles published per zone-set **vintage**, and the manifest names the vintage each release used. Only the paint specification changes when the reader switches metric.
- **Per-cell and per-area species lists** — the same rows partitioned by a spatial tile instead (`cell_model`), because a per-cell question against a model-partitioned surface would scan everything.

See Chapter 16 for the notebooks that build each of these tables.

16 Workflows

Workflows are Quarto notebooks that ingest source datasets, resolve taxonomy, merge models, compute scores and publish a release. They are the only writers of the pipeline database; everything downstream reads published artifacts.

Links:

- marinesensitivity.org/workflows — every notebook, rendered, with the current dependency graph
- github.com/MarineSensitivity/workflows — source

The notebook list is deliberately **not** reproduced here: it is generated from the pipeline's own metadata and changes whenever a dataset is added. What is documented here is the contract that generates it.

16.1 The pipeline is a graph, generated from the notebooks

The pipeline is a `targets` directed acyclic graph, and the graph is **not hand-maintained**. Each notebook declares its own place in it, in YAML front-matter:

```yaml
msens:
  target_name: ingest_aquamaps
  workflow_type: ingest
  dependency: [build_cell_grid]
  output: data/manifests/ingest_aquamaps.json
  dataset: {ds_key: am, response_type: suitability, source_authority: AquaMaps,
            temporal_interval: static, native_format: raster}
```

The target list is built by parsing those blocks, so **adding a dataset means adding a notebook** — not editing a pipeline definition that could disagree with it. The `dataset:` block is also the single source of truth from which the `dataset` registry table is consolidated; a dataset that is ingested without one fails the build loudly rather than appearing in the data with no metadata.

16.2 Stages

Stage	What it does
grid	Builds the global 0.05° cell grid — the sampling unit every model is rasterized onto — with environmental covariates and study-area membership.
ingest	One notebook per source dataset, writing one distribution per model keyed by this release's model id (`mdl_seq`). Vector ranges are rasterized by exact area coverage; rasters are interpolated onto the grid.
merge	Resolves every model to a common taxon, computes each taxon's governing extinction risk, and merges the per-dataset surfaces into one surface per taxon (Chapter 9).
score	Computes per-cell metrics, rescales within ecoregions, and aggregates to each spatial unit (Chapter 10).
schema	Consolidates the registry tables — `dataset`, `model`, `taxon_model`, `listing` — that describe what the release contains.
release	Publishes the release to S3 as partitioned Parquet, COGs, PMTiles and a STAC catalog, writes the version manifest, and (gated) deploys the serving tier. The COG and PMTiles files are written to the shared store, not to the release (Section 15.3).

16.3 Publishing to the asset store

The publisher of distribution files (`publish_native.qmd`) writes only to the content-addressed stores (`cog/`, `native/`) and to the catalog `assets.parquet`; a release prefix receives tables, a manifest and an app bundle, never a distribution file (Section 15.3). The order is fixed: the key of every file is computed from its content first, a file whose key is already in the catalog is neither built nor uploaded, each new file is decoded and checked against its key before upload, the uploaded objects are appended to the catalog, and only then is the release's `native_asset` table written. A **publish gate** stops the run before that last step if any URL in the pointer table is not a catalogued store object, or if any `.tif` or `.pmtiles` object is listed under `{ver}/` (for v8 and v9 that check is a warning rather than a stop until their pre-migration copies are pruned).

16.4 Reproducibility rules

These are load-bearing, not stylistic. Each one exists because its absence caused a specific failure.

Every process is a chunk in a committed notebook, run by rendering to HTML. Including the parts that touch the outside world — the S3 sync, the server deploy, the app reload. Not an ad-hoc shell command. Rendering is what produces the tracked HTML output and the content-hash checkpoint; executing the code without rendering skips both, so it is a diagnostic, never a run.

Side effects are opt-in. Anything expensive or outward-facing sits behind an environment flag that defaults to off, so a routine render cannot publish or restart anything by accident.

The orchestration is part of the process. A notebook parameterized by version is only half reproducible if the loop *over* versions lives in a throwaway shell script, so the multi-version drivers are committed too. Likewise, forcing a rebuild is a flag — never moving output directories aside by hand, which leaves no record of what was rebuilt or why.

A version number is not proof. A notebook whose logic lives in the `msens` package is only as current as the *installed* package. The server once reported the right package version while running an older function, and seven release manifests were regenerated missing their zone tiles as a result — with nothing failing. So every behavior change gets a version bump *and* a changelog entry, and the container pins move with it.

Reproducibility beats uptime. Where a reproducible fix and a fast in-place workaround are both available, the reproducible one wins even at the cost of brief downtime. Drift between a running container and the image it was built from is a bug to fix immediately, not a reason to avoid rebuilding.

16.5 Content-addressed checkpoints

Each notebook writes a manifest recording a **content hash** of what it produced. A downstream stage can then tell whether its input actually changed, rather than trusting a timestamp — which is what makes a re-run cheap when nothing moved, and what makes an unexpected change visible.

16.6 Guarding the science

Scientific rules live in the `msens` package as documented functions, not as SQL strings inside a notebook — so a notebook *calls* the rule and a unit test *asserts* it, and the two cannot drift.

Every non-trivial rule has a test with a small fixture asserting its exact expected output, and every bug that is found becomes a permanent named assertion.

This matters more than the aggregate checks. Whole-pipeline validation — comparing a release's zone scores against the previous one — catches broad drift but hides rule-level breakage: one lost exclusion affected ~750 species while barely moving any aggregate score (Chapter 9).

See Chapter 15 for the schema these workflows produce, and Chapter 2 for what each release changed.

17 APIs

Three services sit in front of the published data:

1. **titiler.marinesensitivity.org** — TiTiler, serving the pre-rendered Cloud-Optimized GeoTIFFs that every map layer now draws.
2. **api.marinesensitivity.org** — a custom R `plumber` API for programmatic (non-tile) data access.
3. **A searchable STAC API** — one Item per model, with real asset hrefs and real footprints, so a client can ask *which models cover this area*.

Static vector data (PMTiles) is served directly by Caddy's file server and needs no API — see Chapter 14. The published Parquet tables can also be range-read straight from S3, including by DuckDB-WASM in a browser.

17.1 Raster tiles: COGs, not SQL per request

Score surfaces are published as one COG per metric × subregion, and per-model distributions as one COG per model. A release's manifest carries each raster's URL and its build-time rescale range, so an application reads the href and hands it to titiler's stock `/cog` routes — no custom endpoint, no query in the tile path.

The COGs are stored **without overviews**, deliberately: the renderer decimates from full resolution on each request, so a pre-built pyramid would disagree with it at low zoom. Their object keys are **content-addressed including the encoding**, because rewriting an object at a stable URL once left `/vsicurl` serving a cached header for bytes that no longer existed — fine at z5+, HTTP 500 at z2–z4.

17.2 The `/msens` factory (retired, kept)

> **Important**
>
> The custom factory below is **no longer in the serving path**. It is retained in the image behind `MSENS_FACTORY=1` for exploratory on-the-fly queries, and is documented here because published reports reference its endpoints.
> It was replaced by pre-rendered COGs after verifying the two produce the same picture: of 105 sample tiles, 100 were byte-identical and the rest differed by 3–6 pixels out of 262,144, from float32 storage. Nothing reaches it now — every metric on every release has a COG.

The factory is a FastAPI subclass of `titiler.core.factory.TilerFactory`, registered at the `/msens` prefix on top of the default TiTiler application.

Source: server/titiler/factory.py.

17.2.1 Endpoints

All `/msens/*` routes accept a `sql` query parameter: a **base64url-encoded SELECT** returning exactly two columns, `cell_id` (integer) and `value` (numeric). The SQL is parsed with `sqlglot` (SELECT-only, no write ops, no dynamic I/O like `read_*` / `httpfs` / `load_*`), executed against a read-only DuckDB connection, and the resulting `cell_id → value` map is cached in-process with an LRU keyed on the canonical SQL string. A `mtime` parameter can be passed for cache-busting (typically the mtime of `sdm.duckdb`).

Endpoint	Purpose
`GET /msens/bounds`	Geographic (EPSG:4326) bounds of the cell-id COG.
`GET /msens/statistics?sql=…`	`{n, min, max, mean, std, p2, p50, p98}` of the SQL result — used by clients to set a stable legend rescale without fetching any tiles. Cached by Varnish.
`GET /msens/tilejson.json?sql=…&colormap=…&rescale=…`	TileJSON 2.2.0 document with the tile URL template — lets any TileJSON-aware client (QGIS, MapLibre, Mapbox GL) consume the layer.

Endpoint	Purpose
`GET /msens/tiles/{z}/{x}/{y}.png?sql=…&colormap=…&rescale=…`	The tile endpoint. Renders a 256×256 RGBA PNG by (a) executing the SQL and LRU-caching the `cell_id → value` map; (b) reading the z/x/y window of the cell-id COG with nearest-neighbor resampling; (c) looking the value up per pixel, rescaling and colorizing; (d) returning PNG with `Cache-Control: public, max-age=604800`.
`GET /msens/tiles/{z}/{x}/{y}.png?sql=…&color=#rrggbb[aa]`	**Single-color mask** variant: every valid (finite-value) pixel is painted with the given hex color; `colormap` and `rescale` are ignored. Used for binary overlays like "cells outside Program Areas".
`GET /msens/debug/cog`	Inspect the underlying cell-id COG (bounds, CRS, dtype, nodata, overviews, `rio_tiler` / `rasterio` versions).
`GET /msens/debug/tile/{z}/{x}/{y}`	Inspect what `rio_tiler.Reader.tile()` returns for a given tile (cell-id range, mask valid-pixel count, top-10 cell-ids by count).

Tile URLs were composed via `msens::cell_tile_url()` with stats from `msens::cell_stats()`; see Chapter 18.

17.2.2 Caching (v7 factory only)

Varnish fronts the factory at `titilecache:6082`. The custom VCL (`server/varnish/titiler.vcl`):

- Normalizes query-param order via `std.querysort()` so semantically-identical URLs share a cache entry.
- Strips `Cookie` / `Authorization` (the factory is stateless).
- Caches 2xx responses on `/msens/(tiles/|tilejson|bounds|statistics)` for 7 days (`grace=1h`, `keep=24h`); errors for 60 s to prevent stampede.
- Adds `X-Cache: HIT | MISS` for debugging.

When the source DuckDB regenerates, the Shiny app bumps the `mtime` query parameter to invalidate all cached tiles at once.

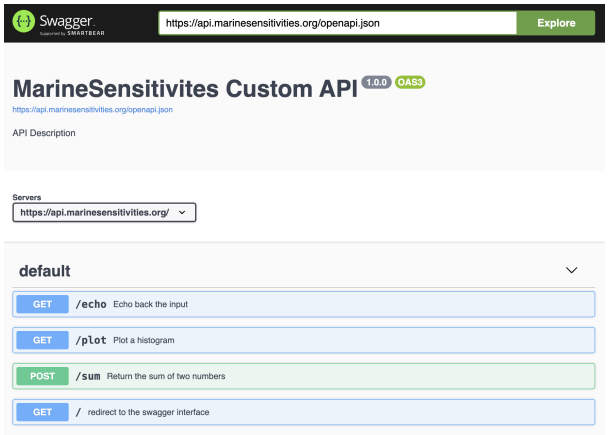
17.2.3 Derived cell-id COG

The factory looks values up against a single-band uint32 **cell-id COG** — one per grid, since the grid changed at v8: `r_cellid.tif` for the v1–v7 regional grid, `r_cellid_global.tif` for the v8 global one.

> **Warning**
>
> These are **lookup images**: a pixel's value is a cell id, not a measurement. A grid's geometry therefore cannot be inferred from the COG — use the grid registry (`msens::grid_spec_for()`), which is the authority on extent, size and longitude convention.

17.3 Custom plumber API

The plumber API at api.marinesensitivity.org covers non-tile use cases — CSV exports, species lookups by feature, on-demand PDF/DOCX report generation. Source: MarineSensitivity/api.



17.4 STAC

Each release publishes a SpatioTemporal Asset Catalog describing its distributions and score surfaces, with a project extension for species-distribution-model metadata. Alongside the static catalog there is a **searchable STAC API** with one Item per model — the static catalog is dataset-level, with assets that are S3 prefixes and a bounding box covering a whole dataset, so it can answer neither "the asset for this model" nor "which models cover this area".

The API backend reads stac-geoparquet through DuckDB, the same shape as the rest of the stack. It is pinned to an exact upstream commit and marked experimental upstream; treat it as a trial surface rather than load-bearing.

17.5 Retired APIs

Removed from the serving path entirely — the containers are commented out in `docker-compose.yml` and are not running:

- `swagger.marinesensitivity.org` — PostgREST over PostgreSQL.
- `tile.marinesensitivity.org` — `pg_tileserv` vector tiles from PostGIS, superseded by static PMTiles served by Caddy's `file` subdomain (see Chapter 13).

18 Libraries

By creating an R package, we document functions and make them easily available across workflows and apps.

- `msens`
  R library of functions for mapping marine sensitivities, sponsored by BOEM

Functions `read` data from Chapter 15, `analyze` it, and `visualize` results. The Shiny apps in Chapter 19 consume `msens::*` helpers exclusively — app code never talks to the tile factory or PMTiles URL templates directly.

18.1 Map-tile helpers (as of 0.3.4)

The following helpers bridge the Shiny app to the msens TiTiler factory (Chapter 17):

Helper	Purpose
`cell_tile_url(sql, colormap, rescale, color, mtime, base)`	Compose an XYZ tile URL template for the factory, base64url-encoding `sql` and omitting `colormap`/`rescale` when `color=` is set (single-color mask).
`cell_stats(sql, mtime, base)`	GET `/msens/statistics` — returns `list(n, min, max, mean, std, p2, p50, p98)` for a stable legend rescale.
`add_cell_tiles(m, tile_url, ...)`	Wraps `mapgl::add_raster_source()` + `mapgl::add_raster_layer()` so the Shiny app adds a cell-values (or mask) layer to a map in one call.
`add_cells(m, r, colors, ...)` (retained)	Legacy image-source path for static report figures that need terra `SpatRaster` rendering (e.g., Quarto-to-DOCX output that can't reach the tile server).

The SQL passed to `cell_tile_url()` and `cell_stats()` must return exactly two columns, `cell_id` (integer) and `value` (numeric) — see the examples in Chapter 15.

19 Apps

The MST provides three interactive web applications — two Shiny apps (Scores, Species) plus a newer, static, in-browser Atlas — along with several legacy applications from earlier development phases.

> **One application, every release**
>
> Every one of the three current apps renders **any published release** rather than shipping a per-version fork. Scores and Species do this from the URL (`/v3/scores/`; older `?ver=v3` links redirect there) — the version is data, settled by the release's published manifest, so a fix reaches every version at once, and the 18 former per-version Shiny instances are retired with their URLs redirecting permanently to the equivalent path. The Atlas follows the same rule with a different mechanism (Appendix B): one static build reads the same published `versions.json`, resolving the release from `?ver=` on its public host, and — its preview route shipping with 0.10.59 — from the URL path on its signed-in preview host.

These tools allow stakeholders, scientists, and decision-makers to understand the underlying components contributing to an area's sensitivity to offshore energy development, supporting better planning and impact assessment.

19.1 Current Applications

19.1.1 Atlas (beta) — static, in-browser scores, species and places

The Atlas is a newer, third way to explore the MST, currently labelled **beta**: one static Svelte/MapLibre app with **no application server of its own** — scores, clicks, flowers and reports are computed in your browser by a DuckDB-WASM instance, though score tiles, zone outlines and the basemap still come from the project's titiler, PMTiles and CARTO. It combines what Scores and Species do separately — mapping composite scores and mapping individual species — with a capability neither Shiny app has, **Places**: drawing, typing or uploading any area of interest (not only the built-in Program Areas) and getting the same scores, species tables and an exportable report for it, shareable entirely through the URL.

See Appendix B for the full walkthrough, including how a drawn place is encoded into a shareable link.

19.1.2 Scores — mapper of aggregate sensitivity scores

The Scores application is the primary tool for exploring composite sensitivity scores across BOEM Program Areas. Key features include:

- **Multi-resolution viewing**: toggle between cell-level (0.05° grid) and Program Area aggregations
- **Study area selection**: focus on specific Program Areas or subregions
- **Layer selection**: view individual species category scores or composite metrics
- **Flower plot**: interactive summary showing scores by species category with weighted center value
- **Species table**: sortable table of species present in selected area with extinction risk details
- **Treemap**: hierarchical view of species contributions to sensitivity
- **CSV export**: download data for selected area or species

Under the hood the Scores app composes three kinds of map layer (Figure 13.1):

- **Score raster** — a pre-rendered COG per metric × subregion, whose URL and rescale range come straight from the release manifest. The browser fetches only the tiles intersecting its viewport; no full-raster transfer on load (see Figure 13.2).
- **"Cells outside Program Areas" mask** — a single-color overlay published the same way.
- **Zone choropleth** — polygon geometry from a static PMTiles archive published per geometry **vintage**, with per-zone fill colors pushed live into the paint spec from the release's `zone_metric` (see Figure 13.4).

Because every one of those is resolved from the manifest rather than hardcoded, switching the URL's version segment (e.g. `/v3/` → `/v4/`) switches all three together — including drawing the boundary vintage that release actually used.

See Appendix D for details.

19.1.3 Species — mapper of individual species distributions

The Species application enables exploration of individual species distribution models. Key features include:

- **Species selection**: search by common or scientific name, filter by species category
- **Model display**: view merged models or individual source dataset layers

- **Representation toggle**: compare the native-resolution original with the interpolated surface used in scoring
- **Masking toggle**: compare the expert-range-constrained merged model with the unconstrained suitability
- **Species info panel**: ESA listing, IUCN Red List category, MMPA/MBTA protection flags, extinction risk score
- **Deep-linking**: URL parameters support direct links to specific species

See Appendix F for details.

19.2 Legacy Applications

Earlier prototype applications from the research phase are kept in the Appendix under Applications (Legacy) for reference. They are not part of the current serving path.

20 Apps User Guide

Step-by-step walkthrough of the Scores and Species applications

This guide walks through the two interactive applications — **Scores** (app.marinesensitivity.org/v3/scores/) and **Species** (app.marinesensitivity.org/v3/species/) — one control at a time. Each numbered step below mirrors a step in that app's built-in **Take a Tour** (click *About* → *Take a Tour* in either app to walk the same path live), with the relevant control highlighted on the screenshot.

For a feature-by-feature reference instead, see the Scores and Species appendix pages.

> How this guide stays in sync
>
> The steps, captions, and highlighted screenshots are generated directly from the apps' in-app tours (the `Conductor$new()$step(...)` chains in each `app.R`). Running `libs/apps-guide-regenerate.R` re-parses both tours, re-highlights the same CSS element each tour points at, and re-captures the screenshots, writing `figures/apps-guide/steps.json`. So whenever a tour changes, re-running that one script refreshes this whole page.

20.1 Scores app

20.1.1 1. Study Area

Pick a region to focus on. 'Full study area' shows all US federal waters from the Aleutians to the Caribbean and out to the Pacific Island Territories. The other choices zoom into a single subregion.

20.1.2 2. Spatial Units

Toggle between fine-grained raster cells (0.05°) and aggregated BOEM Program Area polygons. Cell mode lets you click any pixel; Program Area mode shows pre-aggregated zone scores.

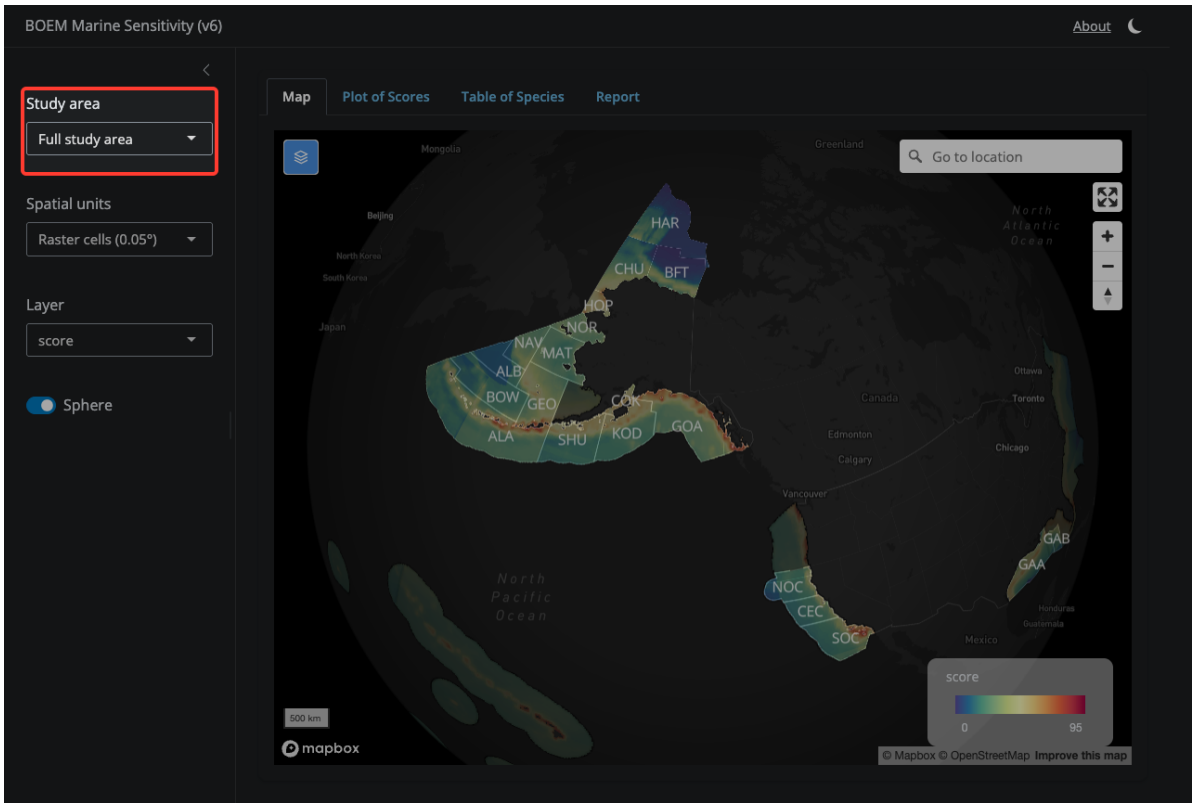


Figure 20.1: Study Area

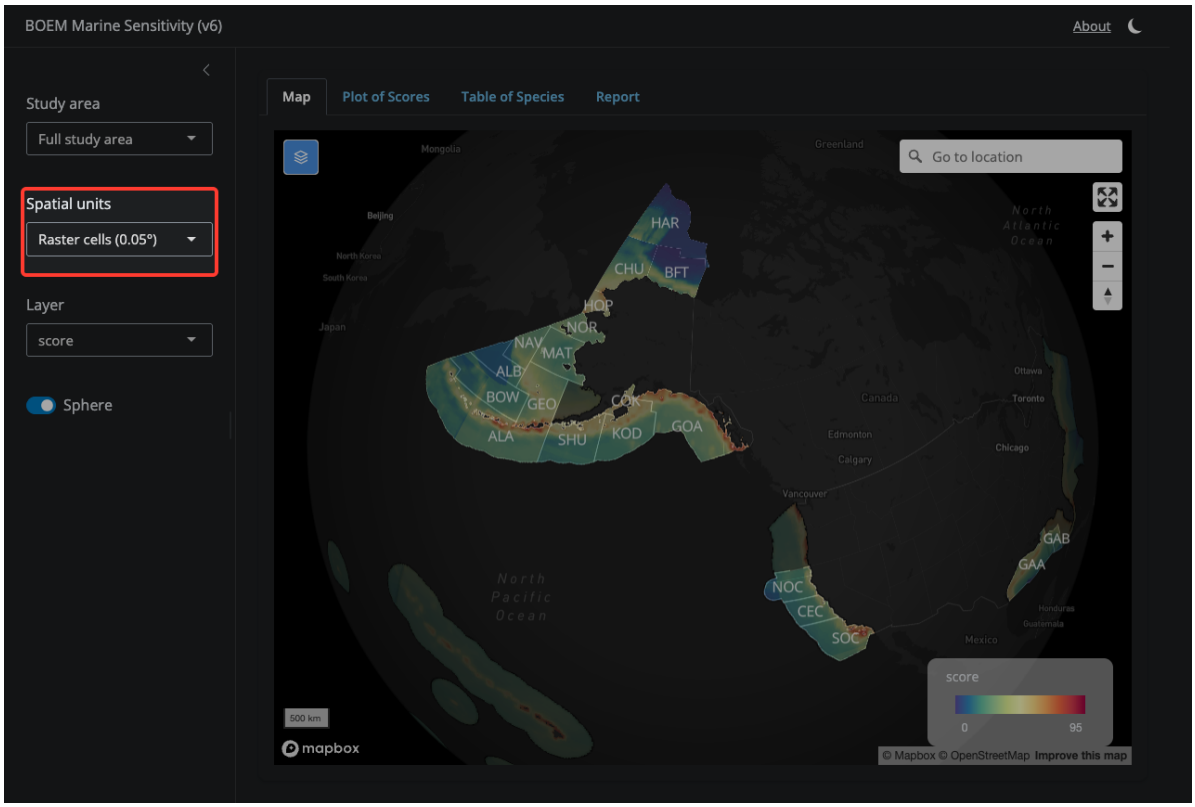


Figure 20.2: Spatial Units

20.1.3 3. Layer Selection

Choose which sensitivity metric to display — composite score, individual species categories (bird, fish, mammal, etc.), or primary productivity. Note that some cells (Atlantic, Gulf of America, Hawaii, Puerto Rico, Pacific Islands) have scores but lie outside any v6 BOEM Program Area; they appear dimmed under a 'Cells outside Program Areas' overlay.

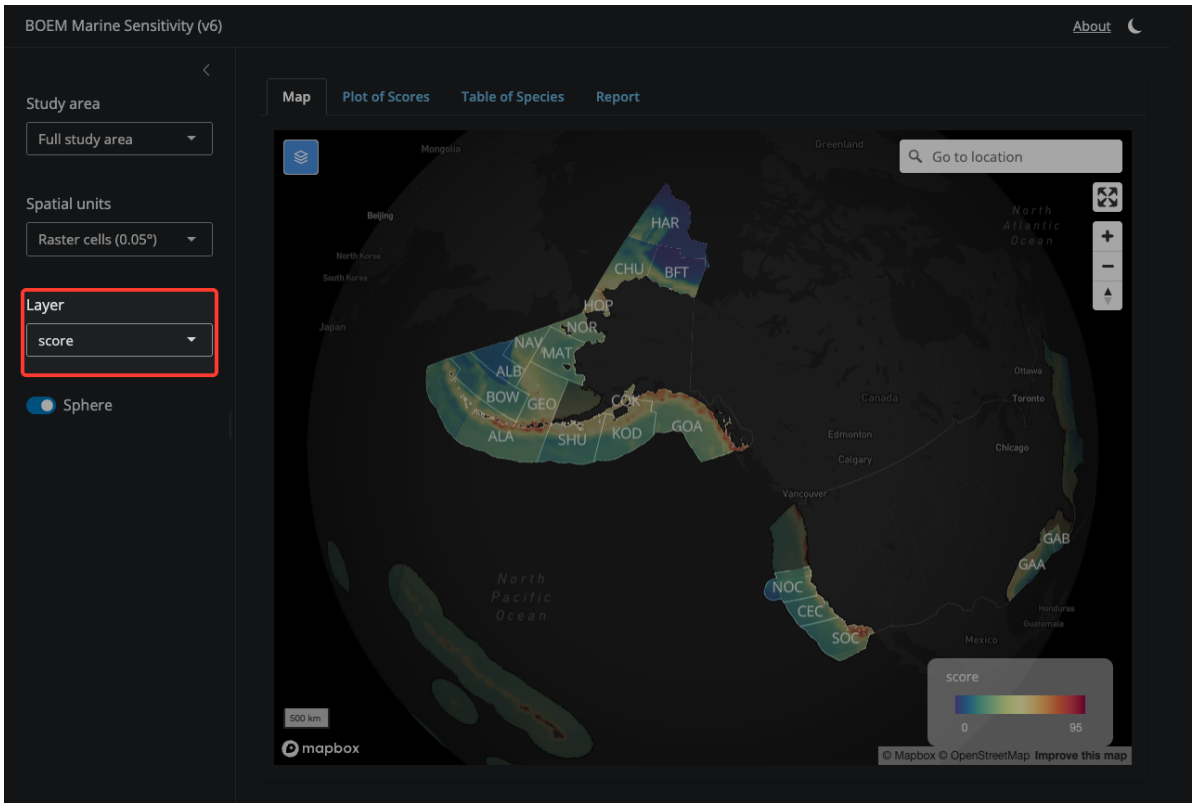


Figure 20.3: Layer Selection

20.1.4 4. Map tab

The Map tab is where you explore scores spatially — click cells or click Program Areas to see their scores.

20.1.5 5. Layers control

Toggle individual map layers on and off — program area outlines, ecoregion outlines, raster cells, and the gray 'Cells outside Program Areas' overlay.

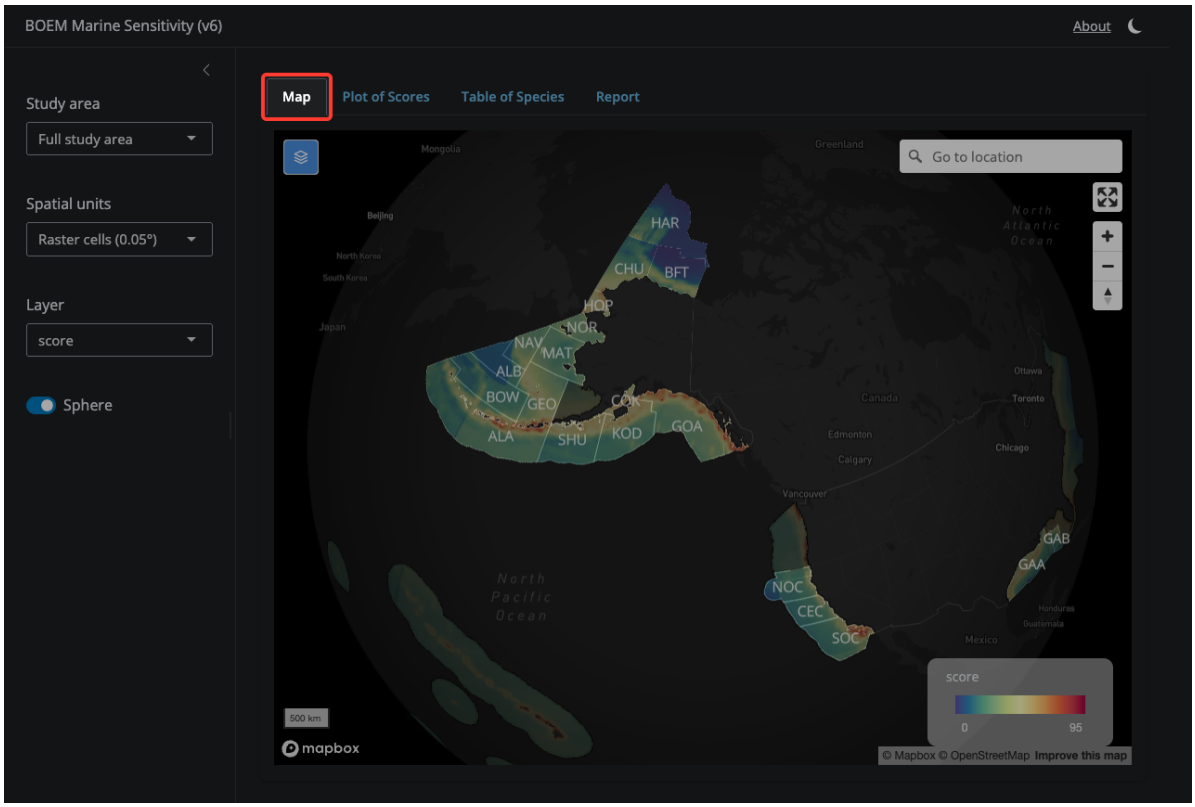


Figure 20.4: Map tab

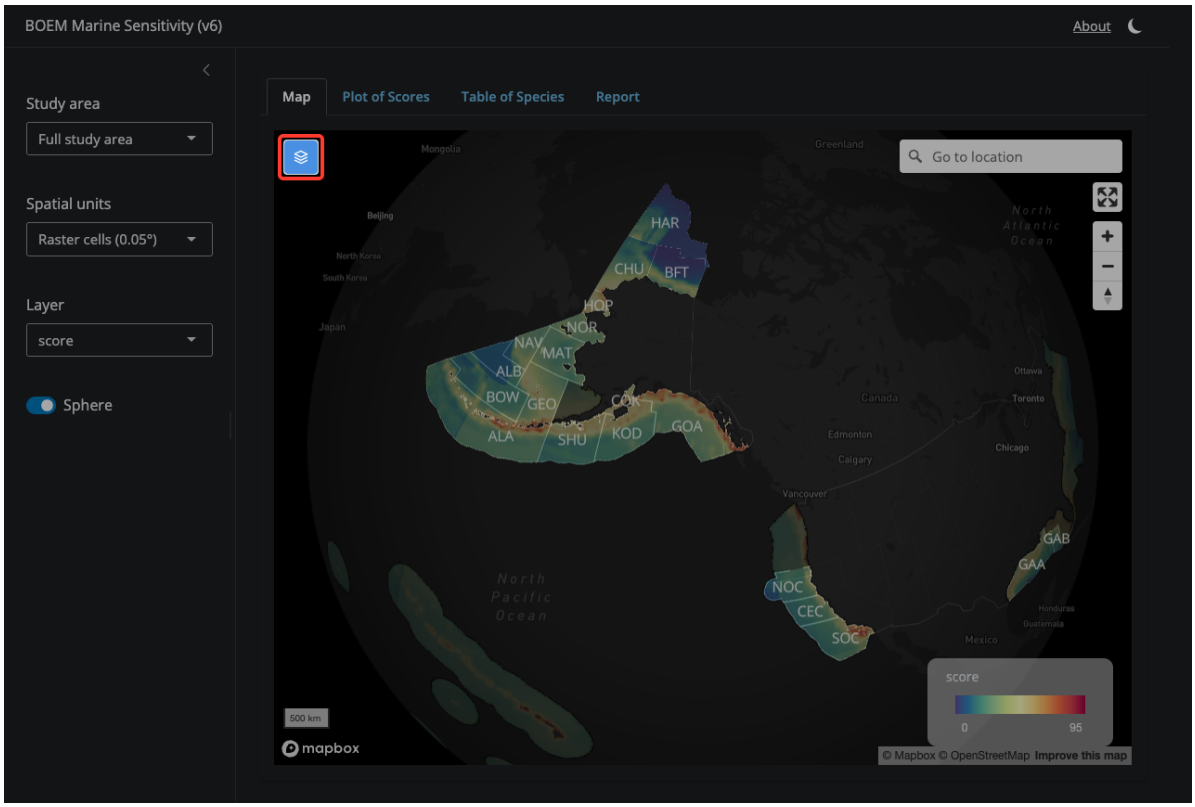


Figure 20.5: Layers control

20.1.6 6. Go to location

Search for a place name and the map will fly there — useful for jumping to a specific Program Area, port, or feature.

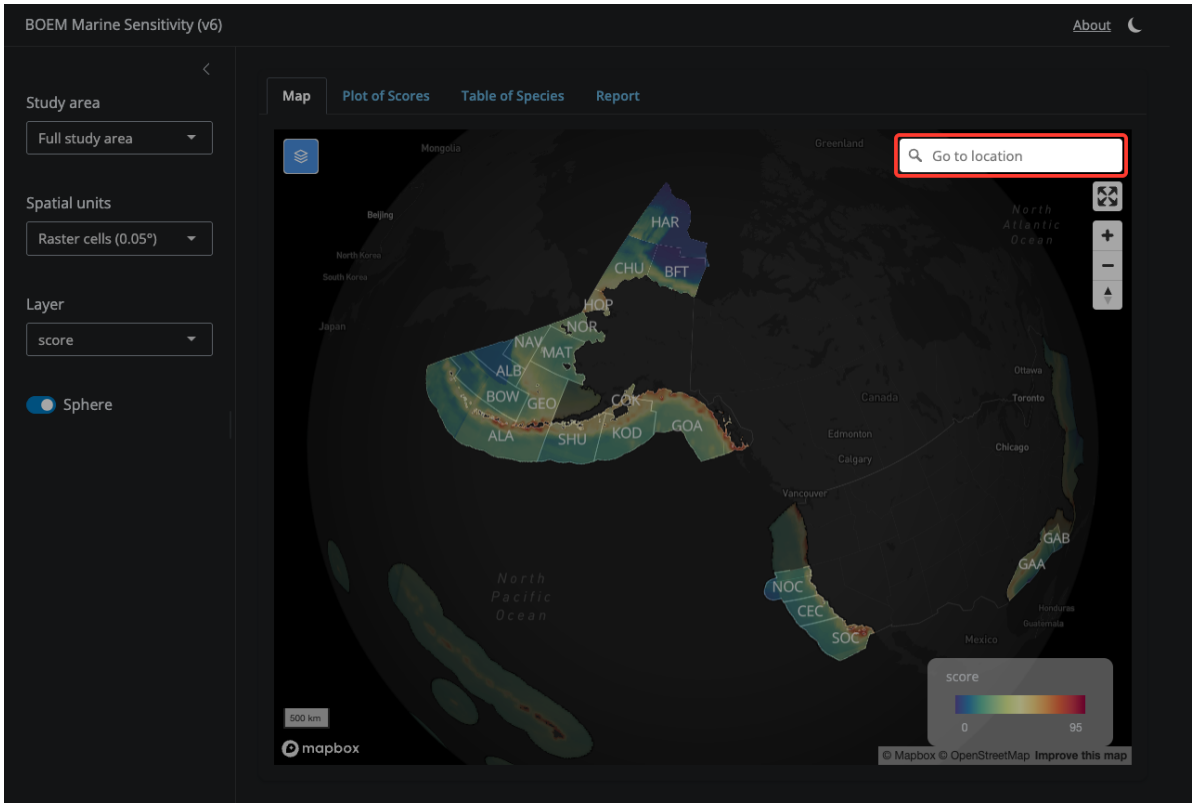


Figure 20.6: Go to location

20.1.7 7. Full screen

Expand the map to fill the entire window.

20.1.8 8. Zoom in / out

Zoom and reset the view. You can also use the mouse wheel, pinch gesture, or double-click.

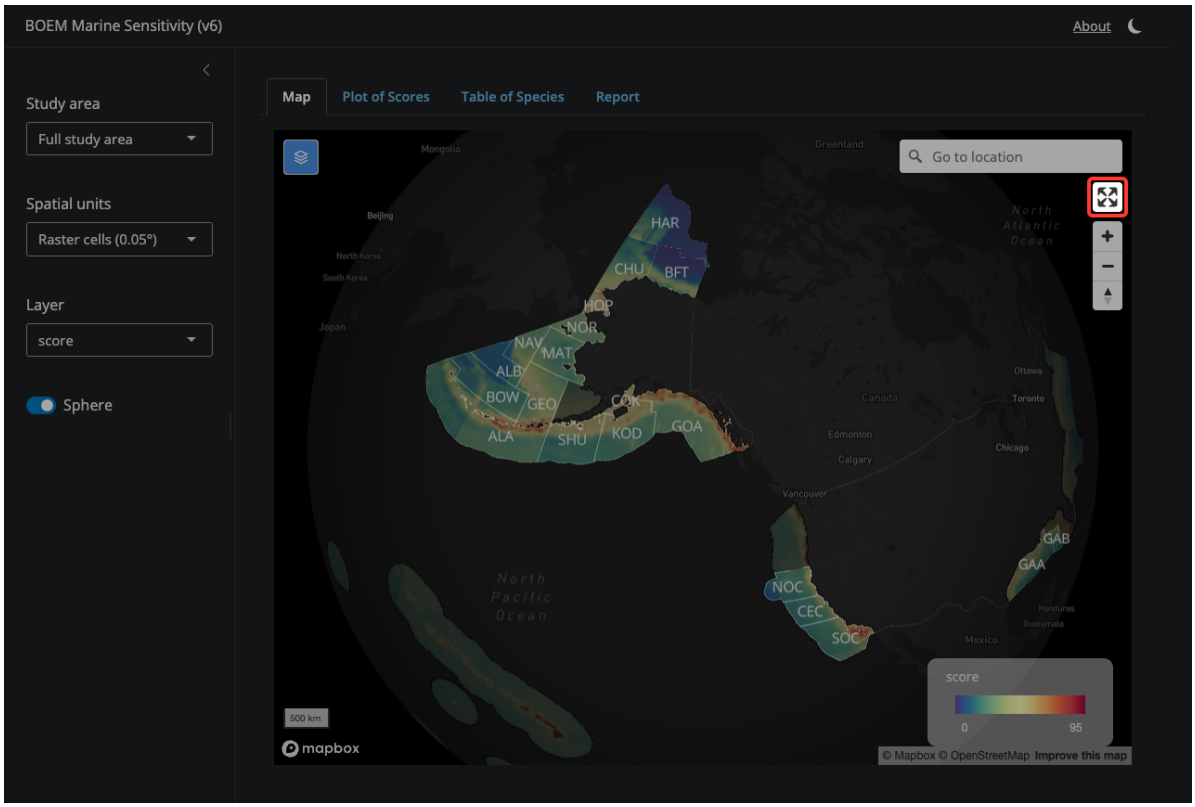


Figure 20.7: Full screen

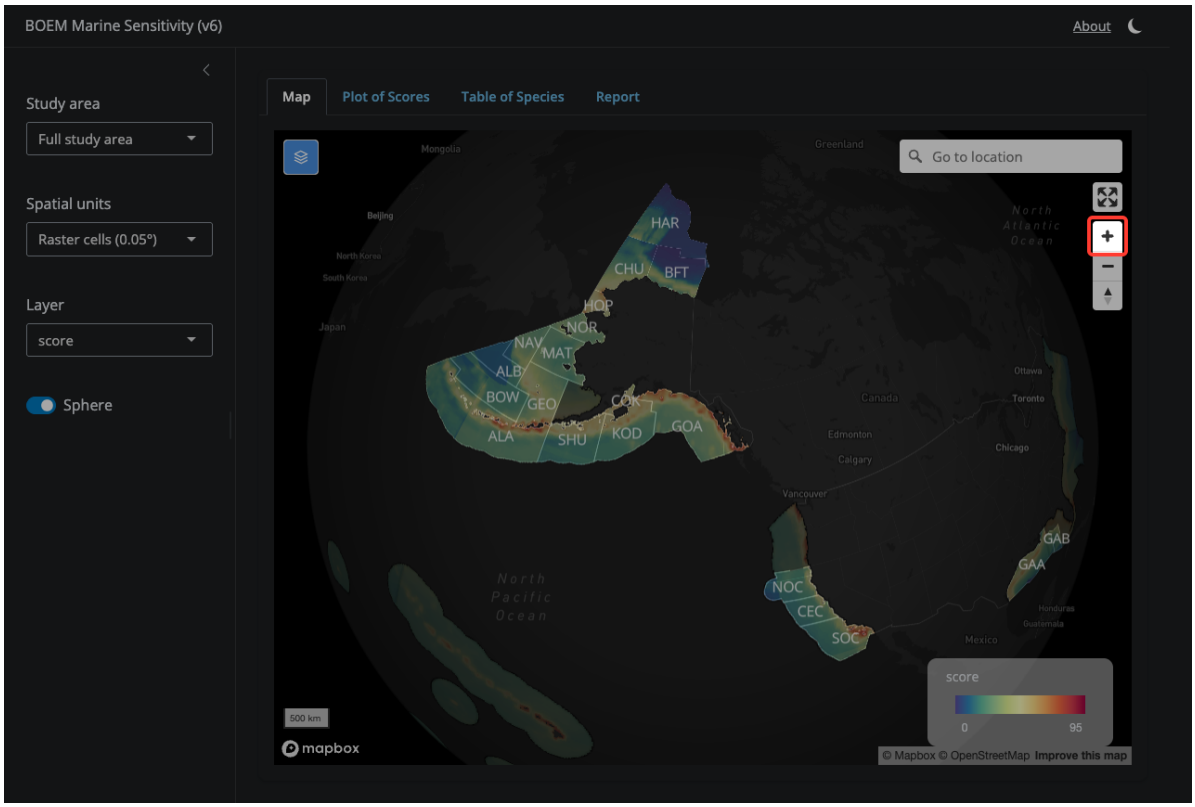


Figure 20.8: Zoom in / out

20.1.9 9. Plot of Scores tab

Switch here to see the flower plot of aggregated sensitivity scores for the current selection, broken out by species category. Petal length = score (0–100); center number = weighted mean. Defaults to 'All USA' until you click a cell or click a Program Area.

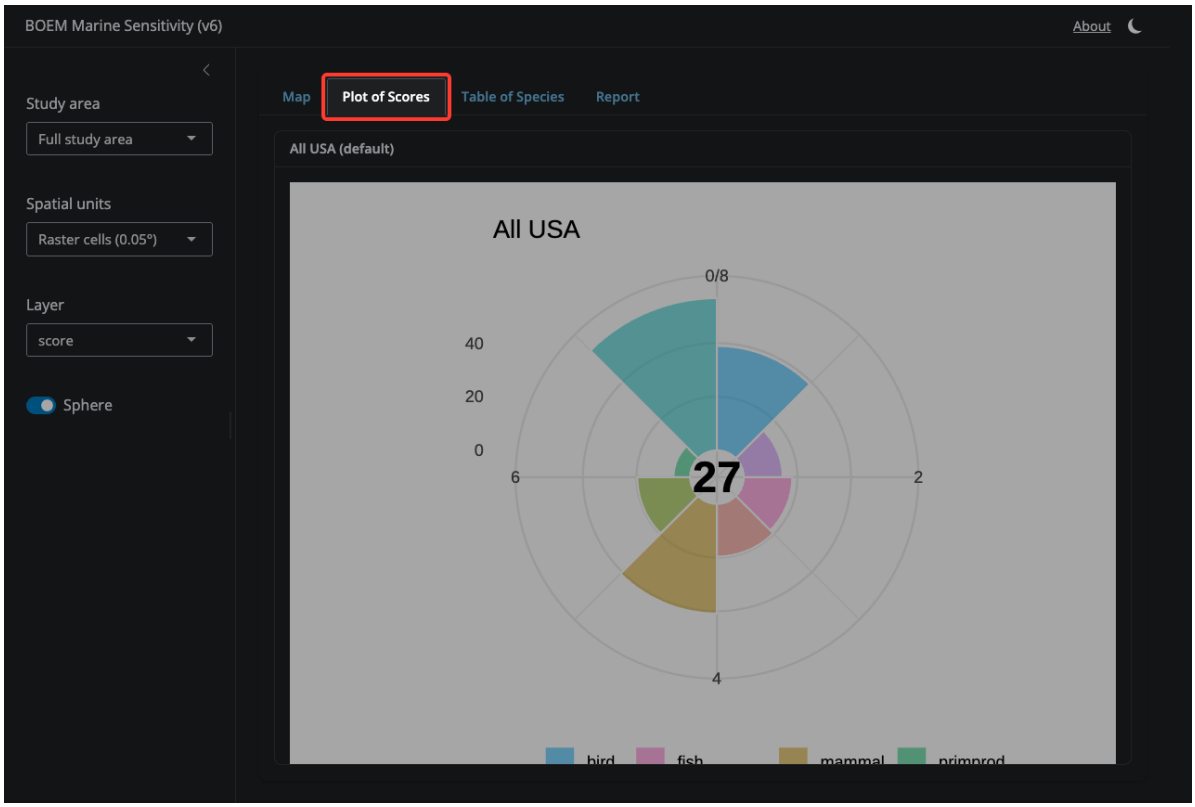


Figure 20.9: Plot of Scores tab

20.1.10 10. Table of Species tab

A sortable, downloadable table of every species in the currently selected area, with extinction-risk codes, areas, and per-category contributions.

20.1.11 11. Report tab

Generate a sensitivity report for one or more custom areas. Build up a list of labeled areas by drawing a polygon on the map (using the polygon tool) and/or by selecting a Program Area,

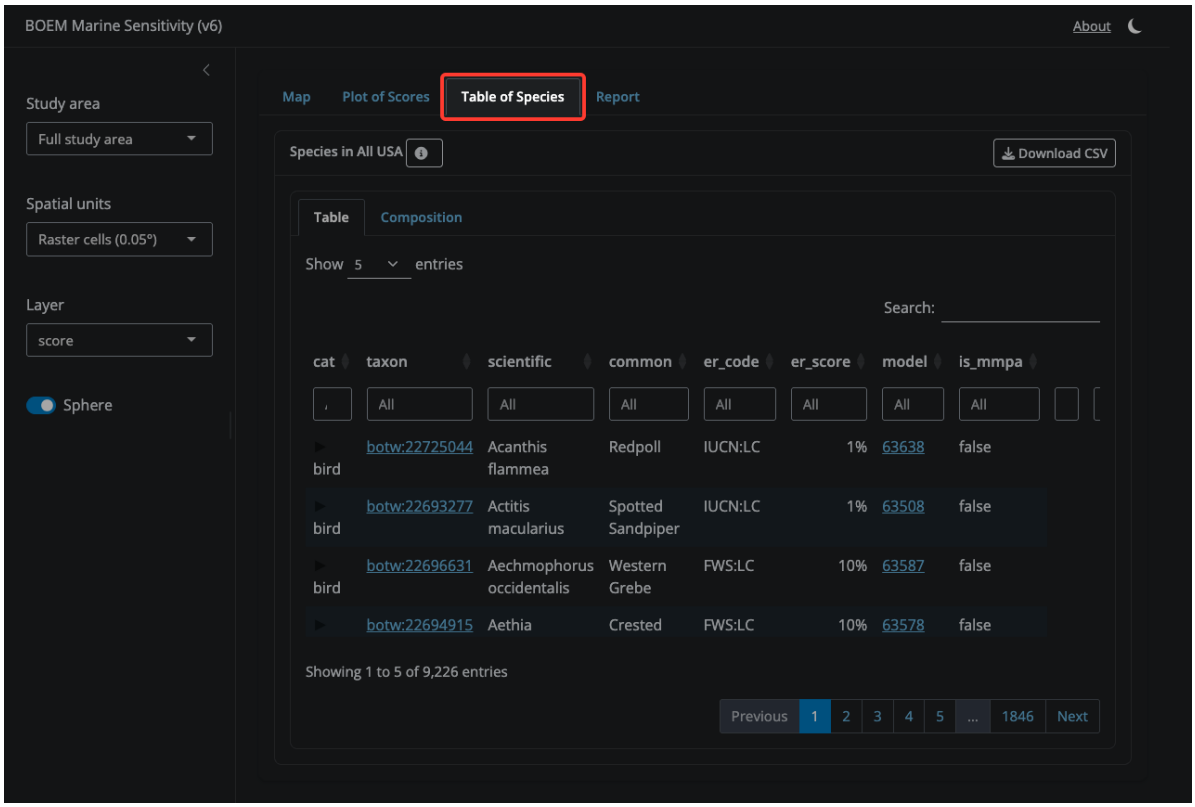


Figure 20.10: Table of Species tab

then clicking ‘Add’ for each. Set a title, data version, and output format (HTML, Word, or PDF), then click ‘Generate report’.

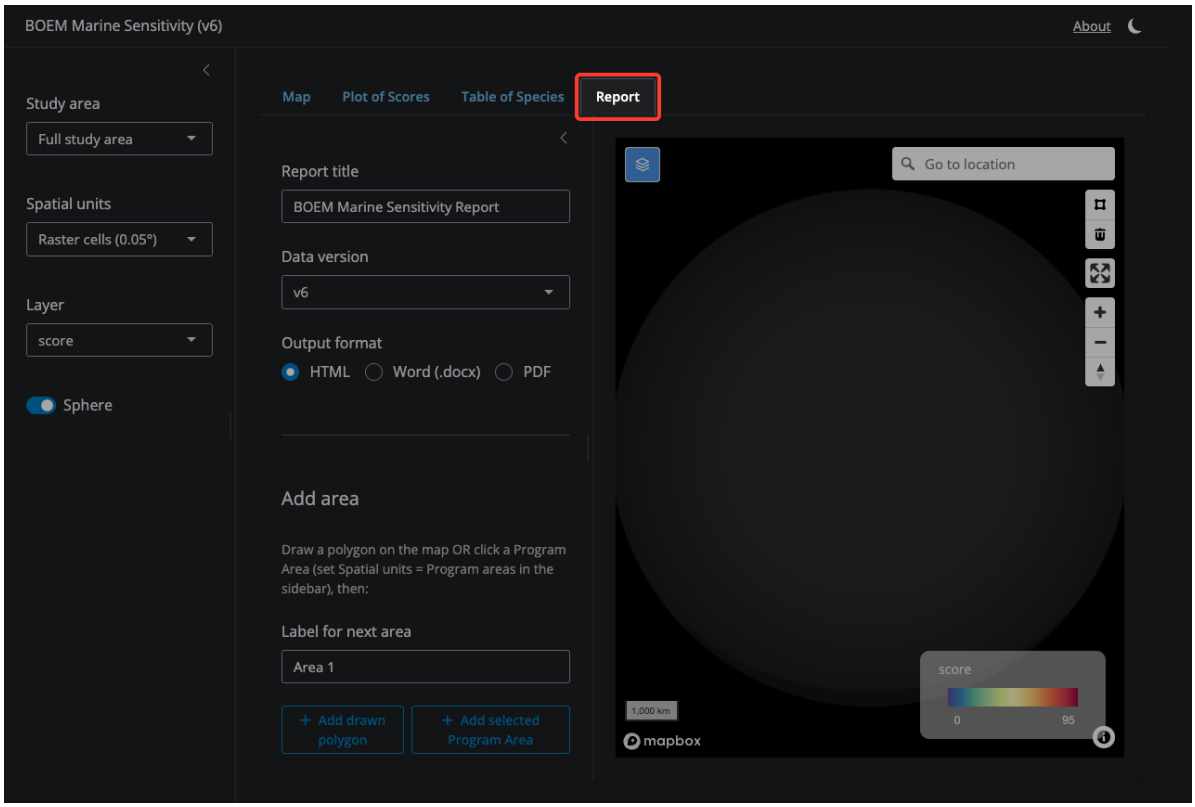


Figure 20.11: Report tab

20.2 Species app

20.2.1 1. Select a Species

Search by common or scientific name. Results are grouped by category (bird, fish, mammal, etc.).

20.2.2 2. Mask Selection

Choose whether to overlay BOEM Program Area or Ecoregion boundaries on the map.

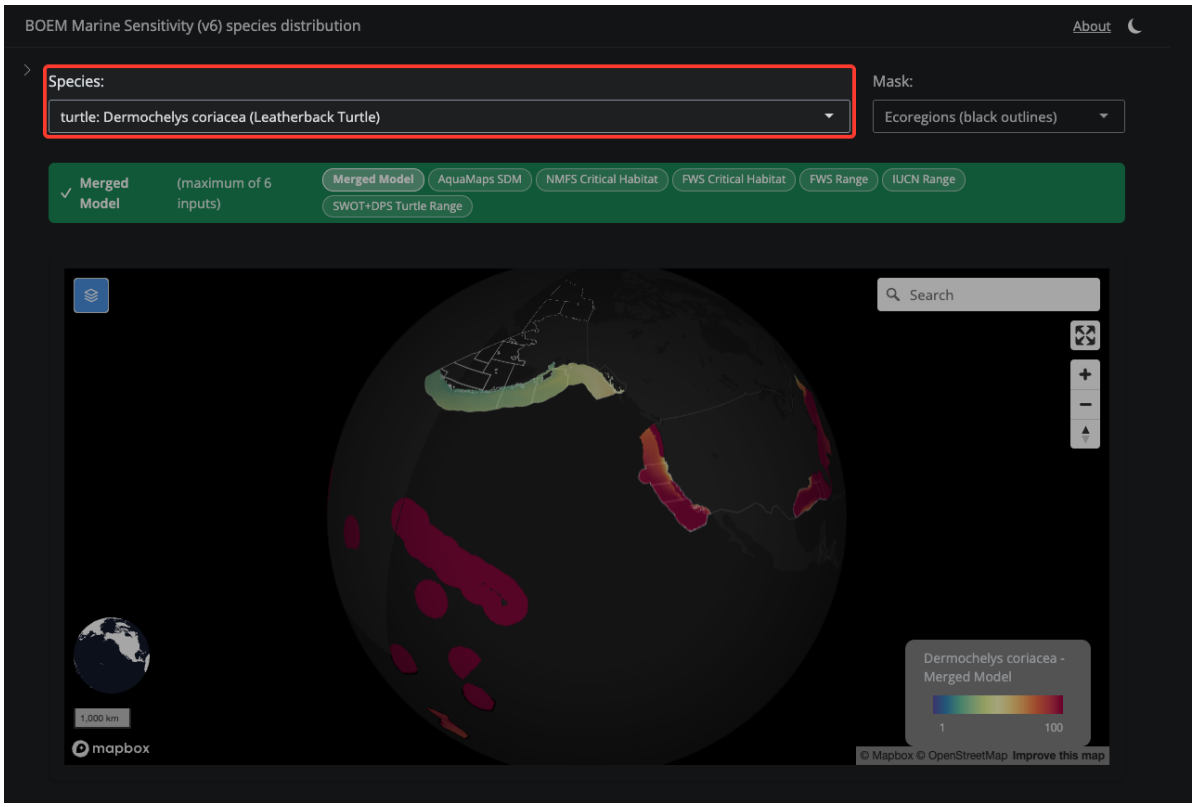


Figure 20.12: Select a Species

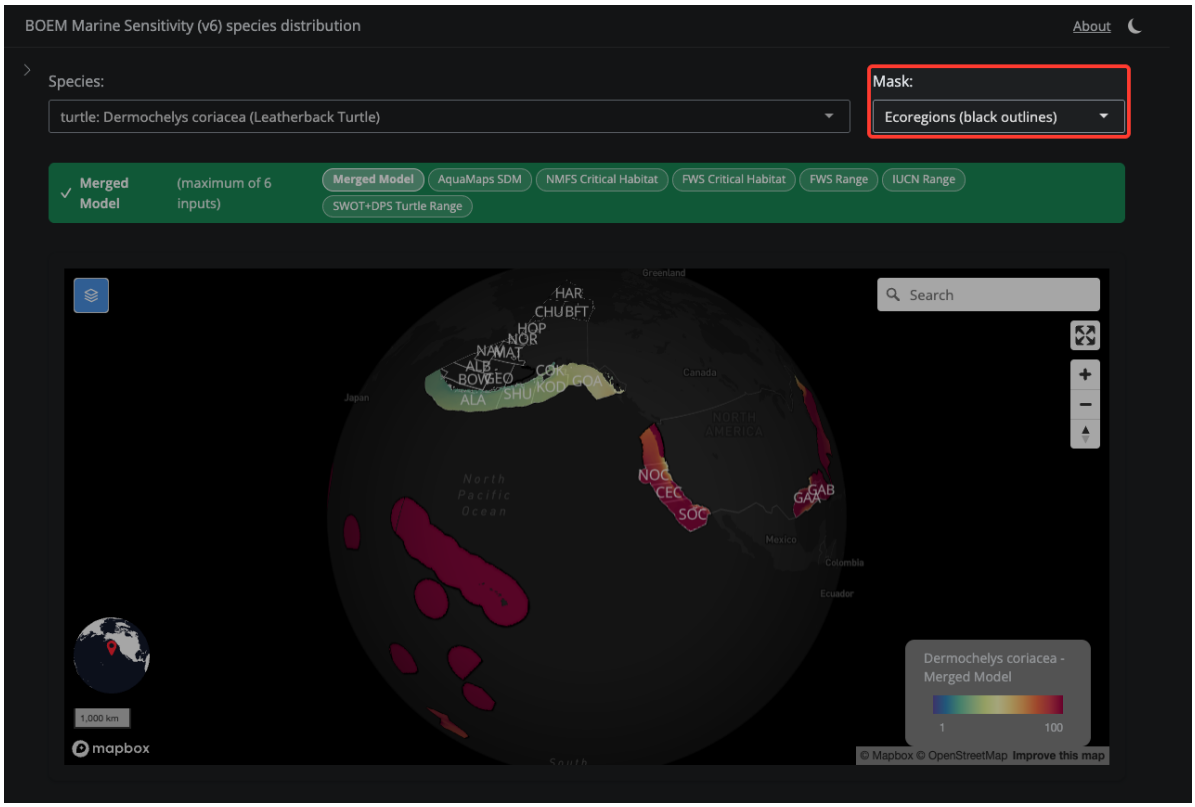


Figure 20.13: Mask Selection

20.2.3 3. Layer Selector

The colored bar shows which model layer is displayed. Green means the final Merged Model; orange means you are viewing a single input. Click the pills to switch layers.



Figure 20.14: Layer Selector

20.2.4 4. Species Map

The map shows the species distribution model. Cell values range from 1 (low sensitivity) to 100 (high sensitivity). Click cells for details.

20.2.5 5. Species Info

Open the sidebar to see ESA listing, IUCN status, MMPA/MBTA flags, and extinction risk score for the selected species.

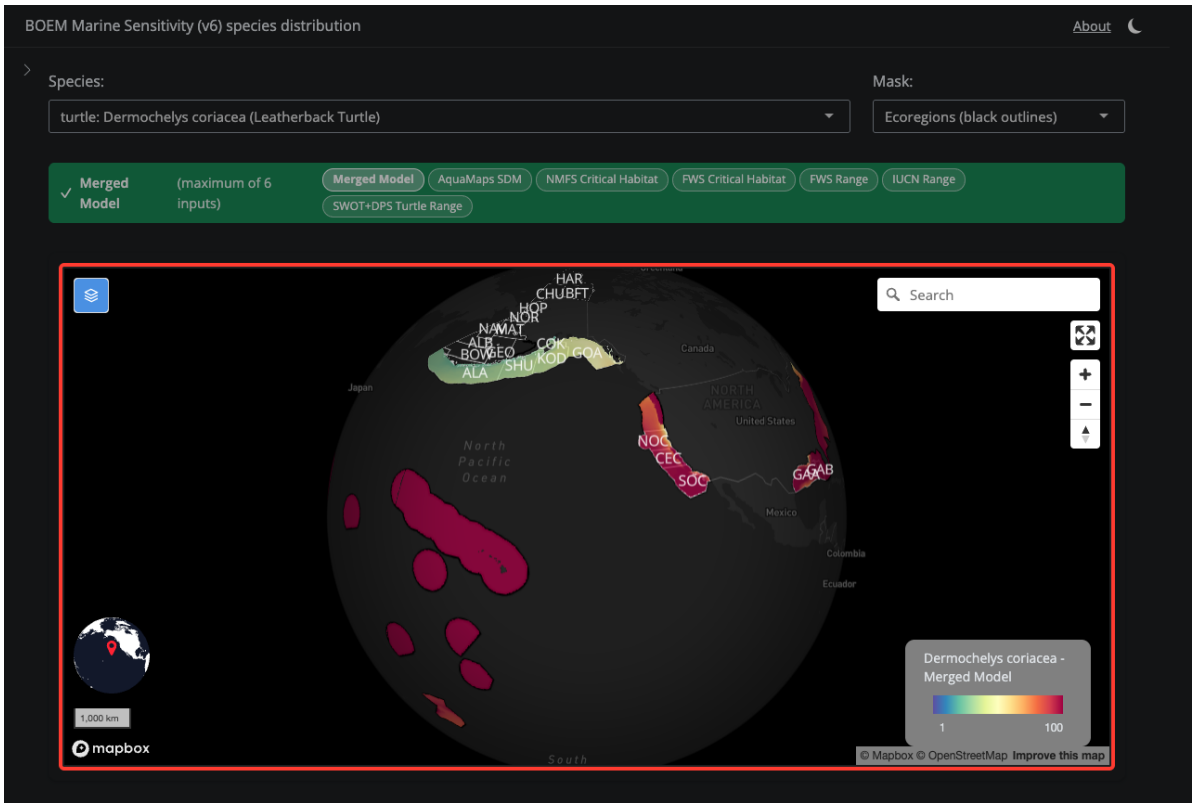


Figure 20.15: Species Map

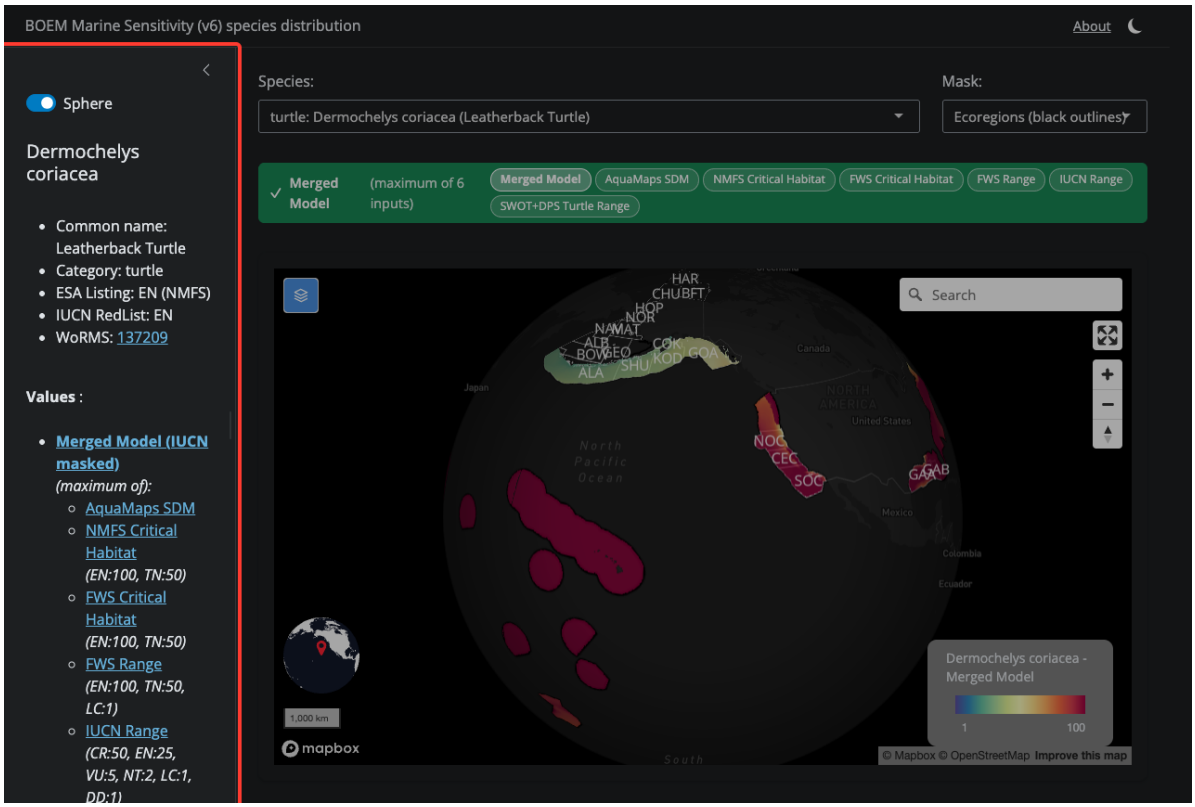


Figure 20.16: Species Info

21 Docs

Technical documentation is principally in this book:

- marinesensitivity.org/docs
  the main documentation site
- github.com/MarineSensitivity/docs
  source code in the Github repository

But there are also some other self-documenting resources:

- marinesensitivity.org/`msens`
  documented R functions

22 Summary

The Marine Sensitivity Toolkit (MST) provides a reproducible system for assessing the sensitivity of marine ecosystems to offshore energy development across US waters. This summary describes release **v3**; figures are read from that release.

22.1 What this release delivers

- **9,795 valid marine species** on a 0.05° grid (~4 km cells) across the US EEZ study area, of which **9,621** fall inside the **20 BOEM Program Areas**, integrating **7 source datasets** — continuous suitability models, regulatory habitat designations and expert range maps.
- **7 scored species categories** (`invertebrate`, `fish`, `other`, `coral`, `bird`, `mammal`, `turtle`), assigned from taxonomy rather than from the dataset a model arrived in.
- **Extinction-risk scoring** (1–100) folds in ESA listing, the MMPA floor (assigned by taxonomy, so no marine mammal is missed) and the MBTA floor (the FWS CFR 50 §10.13 species list, not all birds), falling back to the IUCN Red List where there is no US listing.
- **Model merging** produces one surface per taxon, with the **expert range constraining** suitability predictions rather than merely competing with them, and the governing extinction risk applied at the range cells. Sea turtles merge multiplicatively, so risk varies spatially within a species.
- **Ecoregional rescaling** normalizes each metric to 0–100 within a BOEM Ecoregion, so scores are comparable among Program Areas in one ecoregion without being dominated by naturally species-rich waters.
- **Interactive applications** — Scores and Species — render **any published release** from the URL path (e.g. `/v3/scores/`; an older `?ver=v3` deep link still resolves, 301'd to the equivalent path), so a reader can look at the release a report cited rather than only the newest one.

22.2 Delivered since the first release

Several capabilities listed as future work in earlier editions of this documentation have since shipped:

- **A DAG pipeline.** The workflow is a `targets` graph generated from each notebook's own metadata, so adding a dataset means adding a notebook, not editing a pipeline definition.
- **Cloud-Optimized GeoTIFFs and a STAC catalog.** Distributions and score surfaces are published as COGs with STAC metadata, discoverable through a searchable STAC API, and the score rasters replaced an on-demand SQL-per-tile factory.
- **Partitioned Parquet releases.** Each version is published as Parquet on S3 and read directly, including from the browser, rather than shipped as a monolithic database file.
- **One application across all releases.** The per-version application forks are retired; version is data, settled by a published manifest, not a code fork.
- **Whole-range models.** Distributions are captured globally rather than clipped to the study area, so a species map shows the species instead of the analysis extent.

The **H3 hexagonal grid** proposed earlier was evaluated and **not adopted**: v8 stays on a 0.05° raster grid, made global rather than regional. Interpolating suitability onto hexes added a resampling step without improving the analysis, and the raster grid aligns directly with the source environmental layers.

22.3 Future directions

- **OBIS/GBIF validation** — refine distributions against occurrence records, with temporal filters.
- **Density models** — NOAA SEFSC and NCCOS density surfaces are ingested and catalogued but excluded from scoring, pending an agreed mapping from individuals km^{-2} onto the 0–100 suitability scale. They are flagged `is_scored = FALSE` so no count on these pages includes them.
- **Habitat sensitivity** — benthic habitats (coral reefs, seamounts, hydrothermal vents) as a component alongside species and productivity.
- **Stressor–receptor matrices** — extend from sensitivity alone to full vulnerability by adding exposure for offshore wind, oil and gas, and other activities (see Chapter 12).
- **Application enhancements** — arbitrary area selection, taxonomic-tree browsing, automatic figure generation, and scrollytelling explainers.
- **Infrastructure transfer** — support moving the data and applications onto BOEM infrastructure for long-term operation.

References

Behrenfeld, Michael J., and Paul G. Falkowski. 1997. "Photosynthetic Rates Derived from Satellite-Based Chlorophyll Concentration." *Limnology and Oceanography* 42 (1): 1–20. https://doi.org/10.4319/lo.1997.42.1.0001.

Butchart, Stuart H. M., H. Resit Akçakaya, Janice Chanson, et al. 2007. "Improvements to the Red List Index." *PLoS ONE* 2 (1): e140. https://doi.org/10.1371/journal.pone.0000140.

Butchart, Stuart H. M., Alison J. Stattersfield, Leon A. Bennun, et al. 2004. "Measuring Global Trends in the Status of Biodiversity: Red List Indices for Birds." *PLoS Biology* 2 (12): e383. https://doi.org/10.1371/journal.pbio.0020383.

Chang, Winston, Joe Cheng, JJ Allaire, et al. 2024. *Shiny: Web Application Framework for r*. https://CRAN.R-project.org/package=shiny.

Goodale, M. Wing, and Anita Milman. 2016. "Cumulative Adverse Effects of Offshore Wind Energy Development on Wildlife." *Journal of Environmental Planning and Management* 0 (0): 1–21. https://doi.org/10.1080/09640568.2014.973483.

Halpern, Benjamin S., Catherine Longo, Darren Hardy, et al. 2012. "An Index to Assess the Health and Benefits of the Global Ocean." *Nature*, ahead of print. https://doi.org/10.1038/nature11397.

Halpern, Benjamin S., Kimberly A. Selkoe, Fiorenza Micheli, and Carrie V. Kappel. 2007. "Evaluating and Ranking the Vulnerability of Global Marine Ecosystems to Anthropogenic Threats." *Conservation Biology* 21: 1301–15. https://doi.org/10.1111/j.1523-1739.2007.00752.x.

Hare, Jonathan A., Wendy E. Morrison, Mark W. Nelson, et al. 2016. "A Vulnerability Assessment of Fish and Invertebrates to Climate Change on the Northeast u.s. Continental Shelf." *PLoS ONE* 11 (2): e0146756. https://doi.org/10.1371/journal.pone.0146756.

Intergovernmental Panel on Climate Change. 2014. *Climate Change 2014: Impacts, Adaptation, and Vulnerability. Contribution of Working Group II to the Fifth Assessment Report*. Cambridge University Press.

Kaschner, K., K. Kesner-Reyes, C. Garilao, et al. 2019. *AquaMaps: Predicted Range Maps for Aquatic Species*. https://www.aquamaps.org.

Lowndes, Julia S. Stewart, Benjamin D. Best, Courtney Scarborough, et al. 2017. "Our Path to Better Science in Less Time Using Open Data Science Tools." *Nature Ecology & Evolution* 1 (6): 0160. https://doi.org/10.1038/s41559-017-0160.

Mair, Louise, Leon A. Bennun, Thomas M. Brooks, et al. 2021. "A Metric for Spatially Explicit Contributions to Science-Based Species Targets." *Nature Ecology & Evolution* 5: 836–44. https://doi.org/10.1038/s41559-021-01432-0.

Maitner, Brian, Paul Efren Santos Andrade, Luna Lei, et al. 2024. "Code Sharing in Ecology and Evolution Increases Citation Rates but Remains Uncommon." *Ecology and Evolution* 14 (8): e70030. https://doi.org/10.1002/ece3.70030.

Mooers, Arne Ø., Daniel P. Faith, and Wayne P. Maddison. 2008. "Converting Endangered Species Categories to Probabilities of Extinction for Phylogenetic Conservation Prioritization." *PLoS ONE* 3 (11): e3700. https://doi.org/10.1371/journal.pone.0003700.

Morandi, A., S. Berkman, J. Rowe, et al. 2018. *Environmental Sensitivity and Associated Risk to Habitats and Species on the Pacific West Coast and Hawaii with Offshore Floating Wind Technologies*. OCS Study BOEM 2018-031. US Department of the Interior, Bureau of Ocean Energy Management, Pacific OCS Region. https://www.boem.gov/sites/default/files/environmental-stewardship/Environmental-Studies/Pacific-Region/Studies/BOEM-2018-031-Vol1.pdf.

Niedoroda, Alan W. 2014. *Renewable Energy Space Assessment Model (RESA)*. Bureau of Ocean Energy Management.

NOAA Fisheries. 2025. *Endangered Species Act Species Range Geodatabase*. National Oceanic; Atmospheric Administration (NOAA), National Marine Fisheries Service (NMFS), Office of Protected Resources (OPR). https://noaa.maps.arcgis.com/home/search.html?searchTerm=turtle+dps.

Ross, Pauline M., Elliot Scanes, Maria Byrne, et al. 2023. *Surviving the Anthropocene: The Resilience of Marine Animals to Climate Change*. CRC Press.

Wallace, Bryan P., Zachary A. Posnik, Bryan J. Hurley, et al. 2023. "Marine Turtle Regional Management Units 2.0." *Endangered Species Research* 52: 209–23. https://doi.org/10.3354/esr01243.

Wilkinson, Mark D., Michel Dumontier, IJsbrand Jan Aalbersberg, et al. 2016. "The FAIR Guiding Principles for Scientific Data Management and Stewardship." *Scientific Data* 3

(1): 160018. https://doi.org/10.1038/sdata.2016.18.

Glossary

acclimatisation the adjustment of an organism to environmental conditions in the field or environment rather than the laboratory without an adjustment in their genetics. Acclimatisation has been used to describe phenotypically plastic responses in natural conditions. Source: Ross et al. (2023).

adaptation the evolutionary mechanism where natural selection of traits is genetically passed on, typically over many generations, to create an organism suited to the environment. Source: [rossRoss et al. (2023)

adaptive capacity the capacity of the ecosystem or organism to improve and reorganise in response to stress such as climate change through phenotypic plasticity (acclimation, acclimatisation) or adaptation, distributional shifts, and rapid evolution of traits suited to new conditions. Source: Ross et al. (2023).

BOEM Bureau of Ocean Energy Management — US federal agency responsible for managing offshore energy and mineral resources on the Outer Continental Shelf.

COG Cloud-Optimized GeoTIFF — a raster file format optimized for efficient access over HTTP, enabling range requests for partial reads.

DuckDB an embedded analytical database engine used for storing and querying species distribution models, extinction risk scores, and sensitivity metrics.

ecoregion a BOEM-defined ecological region used as the spatial unit for rescaling sensitivity scores to enable meaningful cross-region comparison.

epigenetics the modification of phenotype plasticity of an organism through altered gene expression without an alteration to the DNA sequence. 'Epi' means above the DNA and includes DNA methylation, modification of histones, and non-coding RNA. Source: Ross et al. (2023).

ESA Endangered Species Act (1973) — US federal law providing protections for species listed as Endangered or Threatened by NMFS or FWS.

exposure the magnitude of the change in the environment

fecundity the maximum physiological potential reproductive output of an organism to produce offspring (reproductive output). This differs from fertility, which is the number of offspring born. Source: Ross et al. (2023)

MBON Marine Biodiversity Observation Network; see MarineBON.org

MBTA Migratory Bird Treaty Act (1918) — US federal law protecting migratory birds, including most seabird species. MBTA-protected species receive a minimum spatial floor score of 10.

MMPA Marine Mammal Protection Act (1972) — US federal law protecting all marine mammals in US waters. MMPA-protected species receive a minimum spatial floor score of

20.

MST Marine Sensitivity Toolkit — the cloud-native system developed for BOEM to assess marine ecosystem sensitivity to offshore energy development.

OCSLA Outer Continental Shelf Lands Act — US federal law governing the management of offshore resources, including the mandate to consider environmental sensitivity and marine productivity.

resilience the capacity of an ecosystem, society, or organism to absorb disturbance and reorganise while undergoing change so as to retain essentially the same function, structure, identity, and feedbacks. Resilience reflects the degree to which a complex adaptive system is determined by its capacity to reorganise and adapt in order to avoid being disturbed again. Source: Ross et al. (2023).

SDM Species Distribution Model — a statistical or environmental envelope model predicting the spatial distribution of a species based on environmental variables.

sensitivity the magnitude of response to the change

stressor the stimulus that causes stress to an organism

vulnerability combination of exposure and sensitivity

Part IV

Applications (Current)

A Atlas (beta) — the static, in-browser scores, species and places app

B Atlas

B.1 Overview

The **Atlas** (https://marinesensitivity.org/atlas/?ver=v3) is a third, newer way to explore the MST, currently labelled **beta**: **one static web app**, built with Svelte and MapLibre, that runs entirely in the browser. Where the Scores and Species Shiny apps are server-rendered R processes, the Atlas has **no application server of its own**: scores, flowers, tables and reports are computed in your browser, by a DuckDB instance compiled to WebAssembly (DuckDB-WASM) querying the release's published Parquet directly over HTTPS. It still depends on a few other services for what a browser cannot compute itself — score and species map tiles come from the project's **titiler**, zone outlines from **PMTiles** on S3, and the basemap from **CARTO** — but there is no Atlas-specific backend.

On the public host the release is a query parameter (`?ver=v7`), resolved client-side against the published `versions.json` and defaulting to the promoted release when absent. A `restricted` pre-release is served the same way the preview Scores and Species apps are, at `https://preview.marinesensitivity.org/{ver}/atlas/` (the version as the URL **path**, since Cloudflare Access scopes its reviewer policies by path, never by query).

Figure B.1 shows the default Scores view: the 0.05° cell grid colored by a composite sensitivity score across U.S. waters, dark theme (the app's default), with the tool rail on the left, the Layers panel open by default, and the basemap's attribution in the lower-left corner.

Desktop vs. phone layout. On a wide viewport the five rail tools open a single **dockable panel** that floats over the map; on a phone the identical five tools sit in a labelled **bottom tab bar**, and the panel becomes a **bottom sheet** with three heights (peek, half, full). Figure B.2 shows the phone layout, and Figure B.3 its top bar's More ( ) menu.

B.2 Quick start

Desktop:

1. Open **Places**, and either choose a Program Area from its list or draw a **Rectangle** (click, move, click, then **Done**).



Figure B.1: The Atlas, Scores lens, default view (desktop, dark theme).



Figure B.2: Phone: default view.

Figure B.3: Phone: the top bar's More ( ) menu.

2. Select the place in the list — Program Area or drawn — to see its results: a coverage note (or, for a Program Area, its published cell count, area and composite), a flower and component table, and (on request) its species table.
3. Rename it inline if you like.
4. **Places › Share › Copy link** — the fastest way to hand someone the same place.
5. **Places › Report** for a printable, downloadable document (Print, HTML, ZIP or Word).

Phone: the tool rail is the bottom tab bar; use the Program Area chooser or **Enter coordinates** rather than drawing, and the file picker rather than dropping a file on the map (that gesture needs a desktop pointer).

B.3 Getting around

The top bar, left to right on desktop: the MST mark and title; a **release chip** that opens the version picker; a **Scores | Species lens switch**; a **search field** (Program Areas and coordinates in the Scores lens, Section B.4; species in the Species lens); then, right-aligned, **Share**, **Report**, a **(?) Help** menu (Take a tour, keyboard shortcuts, a Docs link), a **Feedback** button, an **(i) About** button, and a sun/moon **theme toggle**. On a phone (below 900 px) the title collapses to the mark, and Share, Report, Help, Feedback, About and the theme toggle fold into one **More ( )** button beside a **search** button; below 381 px the mark hides too. The keyboard-shortcut list is desktop-only.

The tool rail is five controls, always in the same order, on every viewport: **Layers · Places · Flower plot · Table · Report**. In the Species lens the Flower control fades in place — it has no meaning for a single species — but stays visible with a tooltip explaining why.

The panel that a rail tool opens is one dockable surface: dock left, right or bottom; resize by dragging its map-facing edge or with arrow keys; maximize to the whole stage; or collapse to a small labelled pill (clicking any rail tool re-opens it). Its dock/size/collapsed state is chrome, remembered per viewport, never written into the shareable URL. On a phone the same panel is the bottom **sheet**, with peek/half/full detents. On a desktop the map's **legend** floats in a corner clear of the panel, on whichever side it is docked, and hides while the panel is maximized; on a phone it is a compact **Legend** chip above the sheet that opens the full legend, moving into the sheet's own header at full height.

B.4 Scores lens

The Scores lens shows the composite, extinction-risk-weighted score for every 0.05° cell (or, aggregated, every Program Area) of U.S. waters. Opening **Layers** (Figure B.4) expands the **Data** row into this lens's own controls — study area, spatial unit, the score layer, color palette, a **Sphere** switch (globe vs. flat projection), and a **Cells outside Program Areas** switch

where the release publishes that overlay — above the rest of the **layer stack**, listed top to bottom: **Selection** (pinned at the top), **Zone outlines**, **Data**, **Place labels**, **Roads & buildings**, **Boundaries**, **Bathymetry** (present but disabled until a relief layer is published) and **Land & water** at the bottom. Every row's visibility and opacity can be changed, and most can be reordered — **Data** can never move above **Zone outlines**, so an opaque score raster can never hide the boundaries it is measured against — and **Reset layers** restores the default stack.

Clicking a cell or a Program Area selects it: the URL gains a `sel=cell:<id>` or `sel=zone:<unit>:<key>` token, which the **Flower plot** and **Table** tools then describe (Section B.8). A click on land or outside the scored area shows *"No scored cell here"* with its coordinates and leaves the selection unchanged.

The top-bar **search field** also works here: it matches Program Areas (and any sub-region/ecoregion archive the release happens to publish) by key ("ALA") or by name ("Aleutian"), and coordinates typed as "lon, lat" (e.g. "-140, 57" — "lat, lon" is read correctly too, either by an explicit `lat/lon` label or because the default order would be out of range). Matching is entirely offline, against only what the release itself already publishes — no Nominatim, no third-party geocoder, no network call. Choosing a Program Area result selects it exactly as clicking it on the map would, and flies the camera there; choosing a coordinate result flies to the point and selects that cell. Results are a keyboard-navigable list (arrow keys, Enter to select, Esc to close), on both the desktop top-bar field and the phone's search modal.

B.5 Species lens

Switching to **Species** turns the top-bar search into a species picker (common or scientific name, with an "Only species in US waters" checkbox), and replaces the **Data** row's contents inside **Layers** with the species **card**: the merged model as a filled pill marked with a gold edge, plus one pill per contributing input dataset marked with a grey edge. An input's raster is found the same way the Species Shiny app finds it — through the release's **model-asset registry**, looked up by model key — and an input whose registry row is missing renders as a **struck-through, non-interactive pill**, never simply omitted. Where the selected input publishes both, one **representation** toggle appears ("Original" vs. "Interpolated," or "Delivered" vs. "As ingested" for a dataset already on the scoring grid). Selecting a species flies the map to that model's extent; clicking it then returns that cell's modelled value, or *"No scored cell here"* outside the model's range. The legend floats in a corner of the map (a legend chip on a phone), titled with the species' scientific name and its 1–100 range.

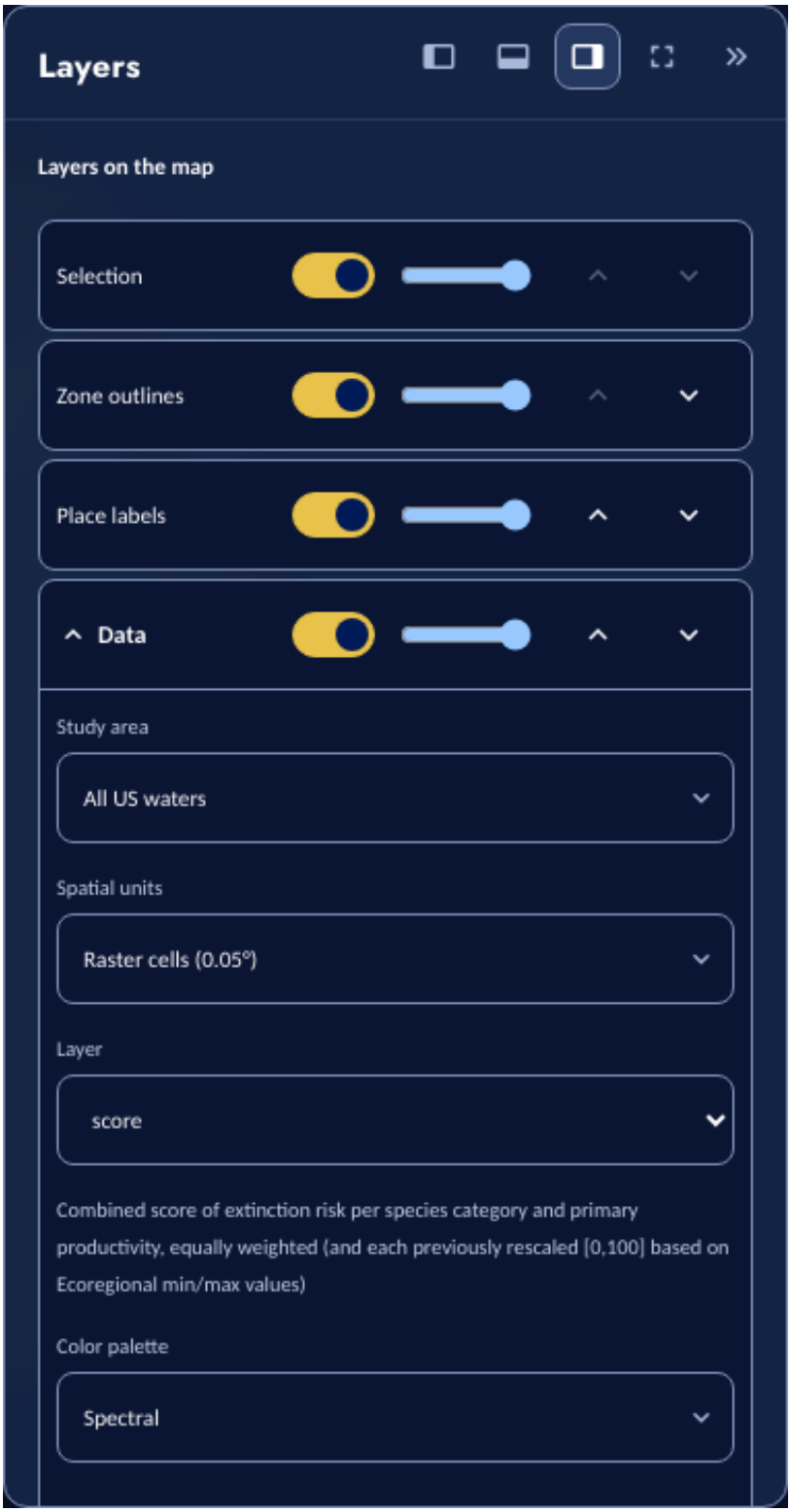


Figure B.4: The Layers panel docked, with Place labels moved above Data (one "move up" click) to show the reorder feature, and Data expanded to its study area, spatial units, layer and palette controls. 173

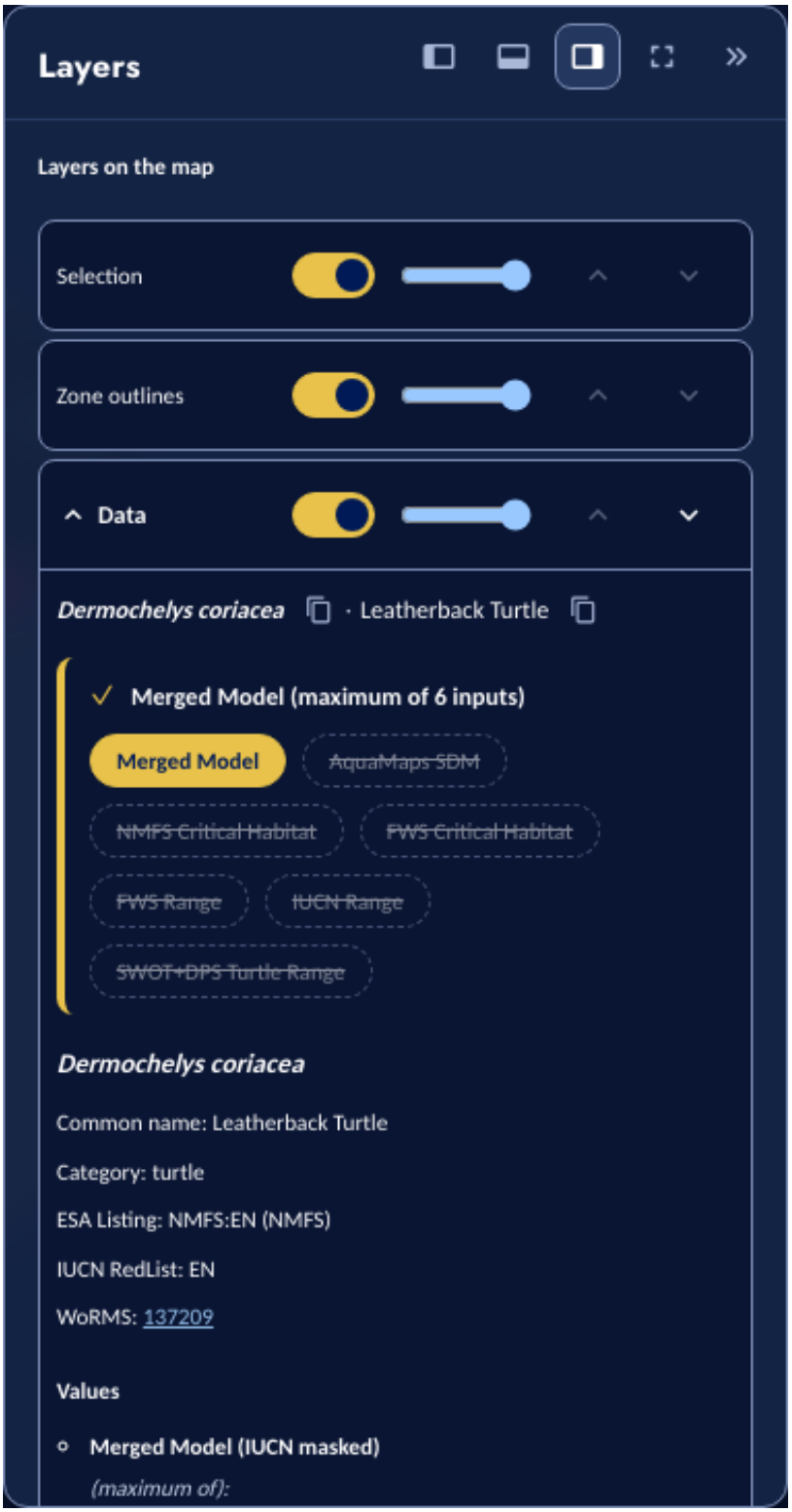


Figure B.5: The Species lens' card for the leatherback turtle (*Dermochelys coriacea*): the merged model with every one of its six inputs struck through — as of this writing, a stale v7 app bundle rather than genuinely missing registry row for all six (see Known Limitations).

B.6 Places

Places is the tool the two Shiny apps do not have: any area you outline gets the same scores and species results the built-in Program Areas get. The empty state reads exactly: *"No places yet. Choose a Program Area above, turn on pick mode and click one on the map, draw a shape, enter coordinates, or drop a file on the map."*

- **Pick a Program Area.** Turning on **pick mode** and clicking only highlights it — Ctrl/Cmd-click, Shift-click or long-press to highlight several — and **"Add to places (n)"** is the button that actually adds it; several picks become **one** place. Or use the panel's own **Program Area chooser**, which needs no pointer and works on a phone. Program Areas are shown as **"Full Name (KEY)"** — e.g. "St. George Basin (GEO)" — from an app-side name table generated off the canonical Program-Area geometry (no published release, v6 through v9, actually publishes a full name on its own Program Area rows; a release that does publish one is preferred when present), in the Places panel, the Scores-lens search and the Zones table and report; from 0.10.59, everywhere — today the Scores-lens map tooltip, Flower title and Table header for a clicked Program Area still show the bare key.
- **Draw** a Polygon, Rectangle or Circle: **click, move, click** for each vertex or corner (not a drag), then press **Done** — a polygon also closes when you click its first vertex again, or press Enter. A circle is stored as a 64-sided polygon. Dragging a finished shape's corner **edits that place in place** (same row, same name, re-analysed) rather than adding a new one.
- **Enter coordinates** — the keyboard/screen-reader alternative to drawing, in a modal dialog: a bounding box (`xmin, ymin, xmax, ymax`), a list of `lon, lat` vertices one per line, or pasted WKT/GeoJSON. A refusal is shown verbatim — what went wrong, why, and how to fix it.
- **Upload a file**, through the picker or by dropping it anywhere on the map: *"GeoJSON, zipped shapefile, KML, GPX, FlatGeobuf, WKT or GeoPackage, up to 10 MB."* **GeoPackage** is now read (polygons only, same as every other format), through the same resident DuckDB-WASM engine the scores boot, via its `spatial` extension — fetched live, with consent, from `extensions.duckdb.org` (the one deliberately non-mirrored third-party download, 22.4 MB); each `.gpkg` upload asks first, not just the first one in a session. A GeoPackage with no vector layer (raster tiles, or a plain attribute table) is refused when `ST_Read` fails to read it (see Known limitations), not by a dedicated check. Other rules that cause a refusal: polygons only (points and lines are refused, never buffered); a shapefile needs its `.prj` (projected coordinates are refused); at most 50,000 vertices; a zipped shapefile at most 50 MB uncompressed and 200 entries; and the shape must touch the U.S. study area. A feature's own name (KML's `<name>`, or a `name`/`title`/`label` property on GeoJSON/shapefile/etc., case-insensitive) — auto-detected from the file's **first** feature and applied to every feature — names each place; a file with no such property falls back to the file name, numbered. A file with several

features becomes one place per feature (up to 20), each named this way, or one merged place, named after the file, your choice.



Figure B.6: The Places panel with two places, both analysed: a Program Area ("Aleutian Arc (ALA)," 875,225 km², 28.3 composite) and a selected, entered-coordinates place ("bounding box," 98,973 km², 100% coverage, 27.1 composite) whose coverage note reads "100.0% of this place is inside the US study area (3,600 of 3,600 cells)"; the drop zone above the list names the accepted formats, GeoPackage included. Captured on 0.10.62.

Each row shows its area and, as soon as it is analysed, its data coverage percentage and composite score directly in the row — for a **drawn, typed or uploaded** place the same analysis runs for every such place in the list, not only the one currently selected; a **Program Area** row shows its own published composite the same way (e.g. "Aleutian Arc (ALA) … 28.3 composite"). Row actions: an inline **rename** (60 characters, drawn/uploaded places only), **zoom to**, **duplicate** (drawn/uploaded only), **delete**, and **Open in report**. Selecting **any** place — drawn, typed, uploaded, or a **Program Area** — expands the same **results** panel below the list, though the note at its top differs by kind: a drawn, typed or uploaded place shows a coverage note ("N% of this place is inside the US study area (a of b cells)", or "This place is outside US waters here: no scores can be computed"); a Program Area shows its own note instead ("N cells, A km²; published composite C."). Both then show a flower plot and a component table, and a species table on request — a drawn/typed/uploaded place behind

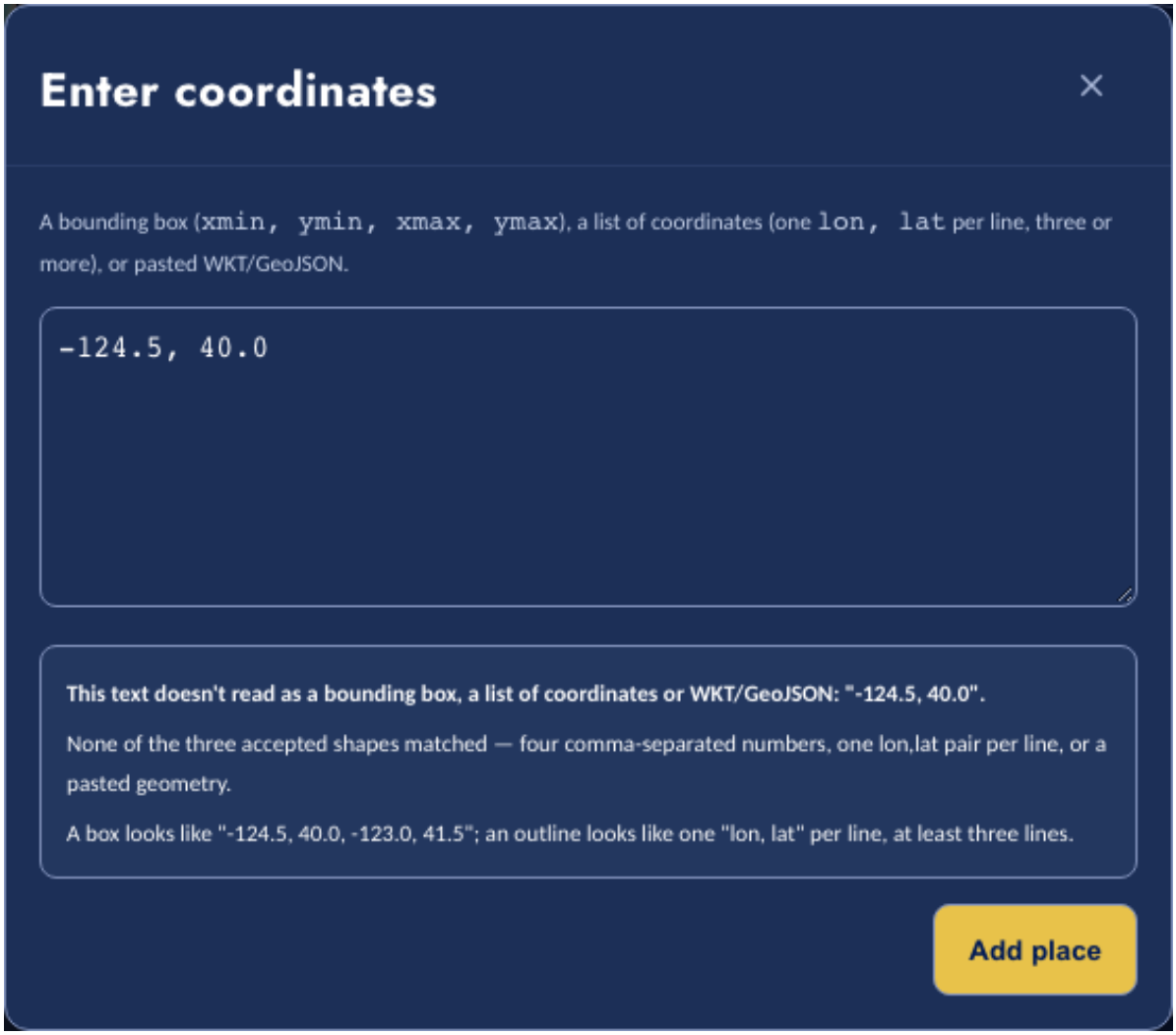


Figure B.7: The "Enter coordinates" dialog, showing a refusal for input that matches none of the three accepted shapes.

a "Loading species needs N tiles, about X MB. Continue?" prompt (species come from map tiles), a Program Area behind a plain **Load species** button with no such prompt (its species table is a precomputed release table, not tiles). A single selected Program Area reads all of this straight off the release (the same path the Scores lens's **Table › Zones** tool uses, Section B.8); a multi-Program-Area selection (several picks added as one place) still sums area and means the composite for the list row, but its flower/component/species panel asks you to select a single Program Area instead of guessing which one's to draw. **Show analysis cells** paints the analysed 0.05° cells (up to 20,000 of them) for a drawn, typed or uploaded place; for a Program Area it reads plainly *"Show analysis cells: not available for Program Areas yet"* — the app bundle the Atlas reads does not yet carry per-zone cells, even though the release's own tables do (see Known limitations) — rather than showing a disabled control with a misleading reason. The list caps at 20 places ("n / 20 places"); **Recent** (this browser only) offers to add a removed place back. The footer holds **Share**, **Download places** (GeoJSON) and **Report**, each disabled with zero places.

Two different Share buttons. The top-bar **Share** simply copies the current address bar to the clipboard — no dialog, no summary, no length check. **Places › Share** opens the real dialog described next, and is the one to use whenever places are involved.

B.7 How places are stored in the link

A place never lives on a server: it is encoded directly into the URL fragment (`#pl=`, never the query string, so it is never sent to a server or a referrer), so opening a shared link reproduces the sender's exact area. Every analysis runs on the **decoded** geometry, never the geometry as drawn, so the numbers can never diverge between sender and recipient. Places are joined with `~`:

```
#pl=z.pa.GAA~g1.Gulf%20box.EAMBAQSf_grYrQPQDwAAOA_PDwA
```

(the decoded form; the address bar itself percent-encodes `~` and other reserved characters)

- **`z.<set>.<key>[,<key>...]`** — a reference to a *built-in* zone (`pa` Program Area, `pl` Planning Area, `er` ecoregion, `sr` subregion); its geometry comes from the release's own PMTiles.
- **`g1.<name>.<base64url bytes>`** — a drawn, typed or uploaded polygon, carried inline as a compact binary encoding (see "For developers" below).
- **`u.<name>.<8-hex-digit digest>`** — the fallback for a place too large to carry (see below): only a name and a digest travel.

> **For developers: the `g1` byte layout**
>
> Each ring is quantized to a fixed precision (3 decimal digits, ~110 m — 4 digits when the place's bounding box is under 0.5° in **both** directions), delta-encoded vertex-to-vertex as zigzag varints running across rings and polygons, and base64url-encoded with no general-purpose compression (delta varints are already dense). Longitudes are stored unwrapped, so a place crossing the antimeridian round-trips without a special case. `msens::place_decode()` is the R decoder; the shared test fixture (`tests/fixtures/place_codec.json`) matches it byte for byte.

The link's length is checked only when you open Places › Share. Adding a place, or copying the top-bar Share link, carries the unsimplified geometry (already quantized to 3–4 decimals) regardless of length. In that dialog: up to **2,000 characters** silently; between 2,000 and 8,000 a note reads "This link is over 2,000 characters — some chat apps and older browsers truncate very long links," with **Copy anyway**; only **above 8,000** does the Douglas-Peucker ladder run — 0.001° doubling to 0.016°, a rung accepted only while the area changes by no more than 1% and every ring stays simple — **aiming to bring the link back under 2,000 characters**; it stops at the first rung that does, or, if none does, uses the coarsest accepted rung provided that still fits under 8,000. This is never silent: the dialog shows the before/after (vertices, area change, cells, Figure B.9), and you click **"Simplify and update the numbers"** to accept it — which is also what rewrites `#pl=`, so the copied link and the displayed numbers can never disagree. If even the coarsest rung is still too long, **Copy link anyway** copies **every** drawn, typed or uploaded place (zone places are unaffected) as a `u.` token instead — your own address bar and session keep the real shapes; the recipient of that copied link sees a named row with no results and a "not analysed yet" chip, never a prompt. The dialog's own advice at that point is to **download the place as GeoJSON** and send that alongside the link. A single small rectangle almost always fits in the silent budget; that is not guaranteed for several places together, or a long base URL.

The rest of the view travels the same URL-is-the-state way, in the query string: `ver`, `lens`, `lyr`/`sp`, `unit`, `area`, `map` (camera), `pal`, `sel`, `layers` (the stack deltas from Section B.4), `theme` and `tour` — each written only when it differs from its default.

B.8 Flower plot and Table

Flower plot and **Table** are Scores-lens tools for a clicked cell or Program Area; in the Species lens, Table is a placeholder, and a drawn Place's flower and species table live in the Places results panel instead (Section B.6), not here.

The flower draws one equal-width annular petal per score component, scaled to its rescaled score; the hub shows the **plain mean** of the components present (not a weighted mean). Since

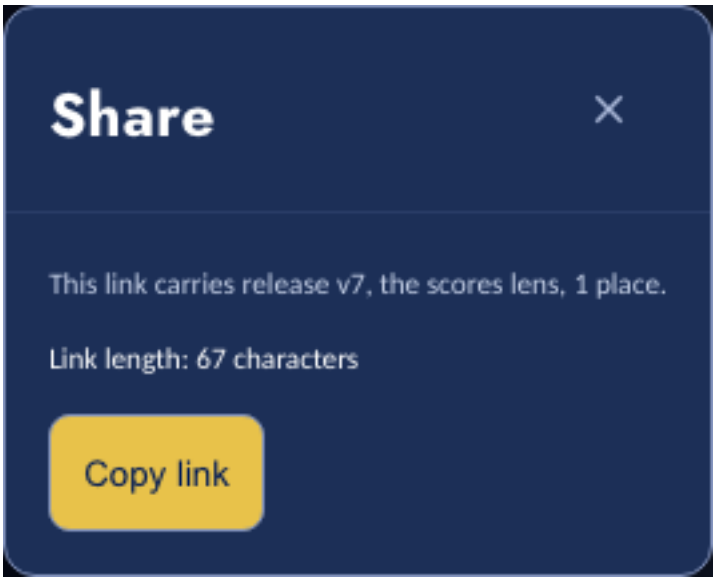


Figure B.8: The Share dialog: what the link carries and its exact length.

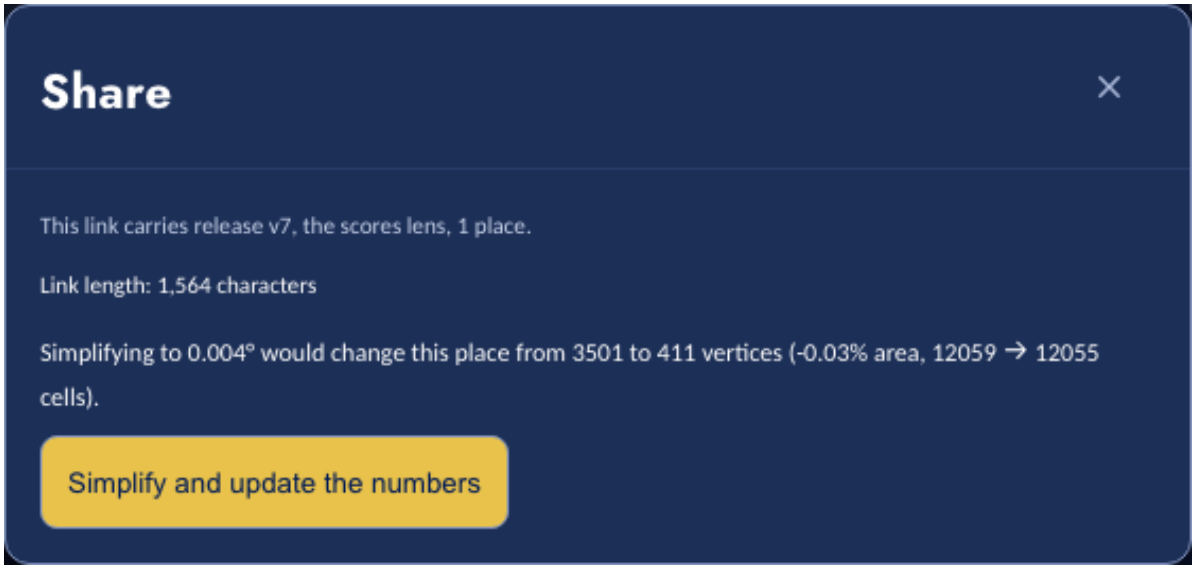


Figure B.9: The Share dialog's simplification ladder for an oversized place: before/after vertex count, area change and cell count, with a "Simplify and update the numbers" button.

0.10.41 the values also list in a real table (swatch, component, score, and a Mean row) below the plot.

Table opens three tabs: a sortable **species table** (Category, Taxon, Scientific name, Common name, **ER code**, ER score, Model, MMPA, MBTA, Area (km²), Avg. suitability, % of category — ESA and IUCN status are folded into ER code), a **zones table** (every Program Area ranked by the **current layer**, with "report on selected" once some are ticked), and a **composition** treemap (one box per species category, sized by suitability × extinction risk × area). Only the **species** table has the phone's compact default columns and a **Columns** control; both tables get sticky headers/first column and horizontal scrolling that never spills the panel. Only the species table's **Download CSV** exports it; the zones table and the composition treemap have no export yet. A drawn place's Components and Species tables in the Places results panel each have their own working **Export CSV**.

B.9 Report

Report reports **every place currently in the link** (`#pl=`); with no places, it reports the single selected Program Area; with neither, it shows a chooser. A row's own **Open in report** reports just that row's place. The document renders progressively as its data arrives:

- a **header band** — mark, release chip, generation time, permalink and QR code (a restricted release also gets a PREVIEW banner and print watermark);
- an **introduction**;
- a collapsible **Parameters** section (kind, keys or vertices, area, cell count, study-area share, token);
- a **map** — interactive on screen; a static snapshot only in print and the HTML export;
- one **flower plot** per place, under "Plot of Scores";
- a **Table of Scores** — component and overall scores, with a footnote when a component covers under 99% of the area, or is missing entirely because the release's own 5% coverage floor dropped it;
- a **Summary of Species** per place (a category × count cross-tab, plus the top 20 by score);
- **Sources and Method**, with the release's own citations — each query behind these numbers is a parity-tested twin (`npm run parity`, max$|\Delta|$ < 1e-9) of an `msens` R function; *"Reproduce in R"* prints the exact calls to `msens::place_decode()`, `cells_in_polygon_grid()` and `scores_for_cells()` that reproduce it, needing `msens` 0.44.0 and a local copy of the release's `sdm.duckdb`;
- a **Provenance** section (release, status/access, app build SHA, DuckDB-WASM version, tables read, SQL run).

Four exports, all built client-side: **Print** (the browser's dialog; roughly five Letter pages); **Download HTML** (one self-contained file — the flowers stay inline SVG, only

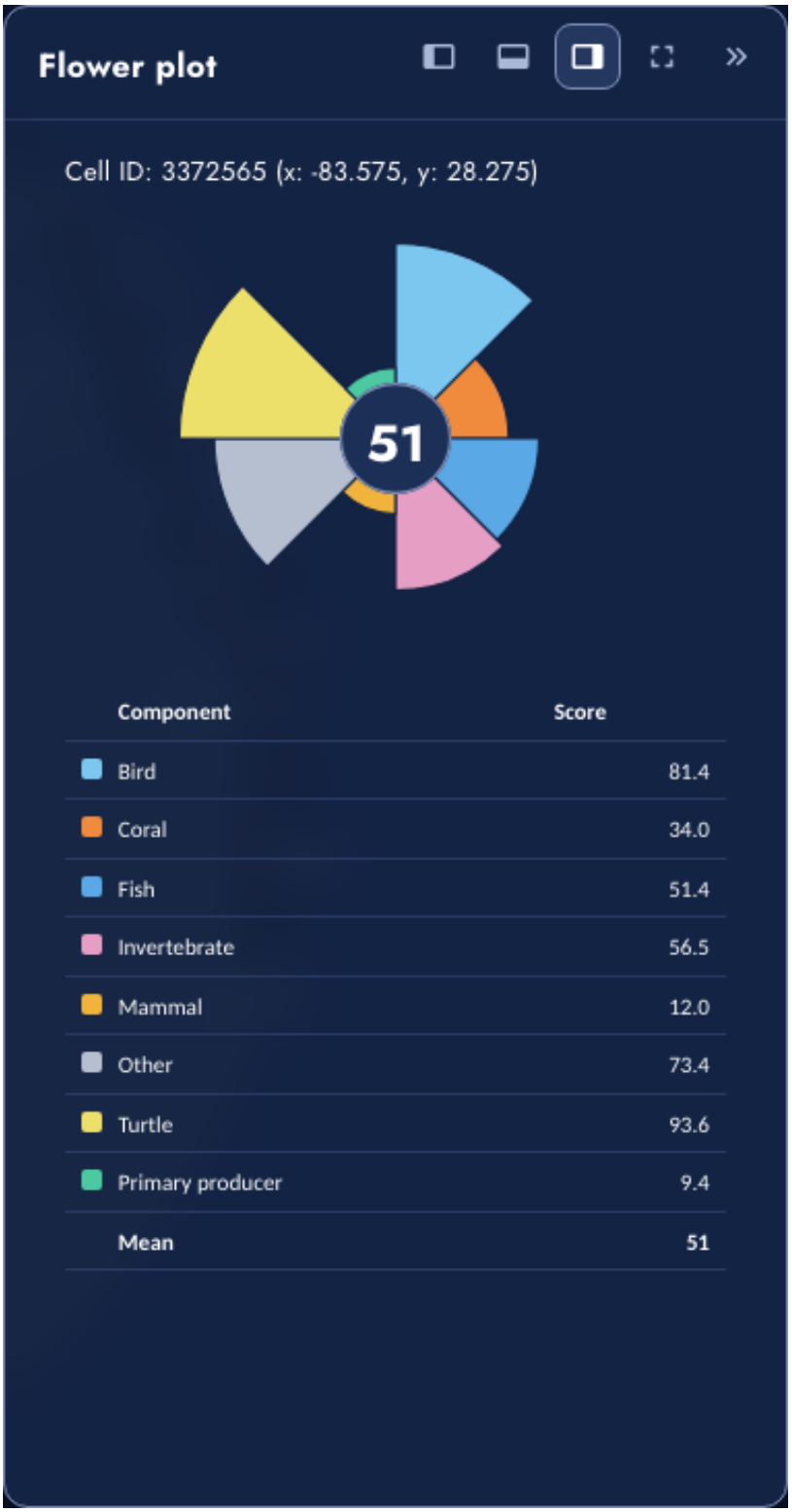


Figure B.10: The Flower plot for a clicked cell in the Gulf of America: one petal per score component, and the values table with a Mean row below it.



Figure B.11: The Table tool's species tab.

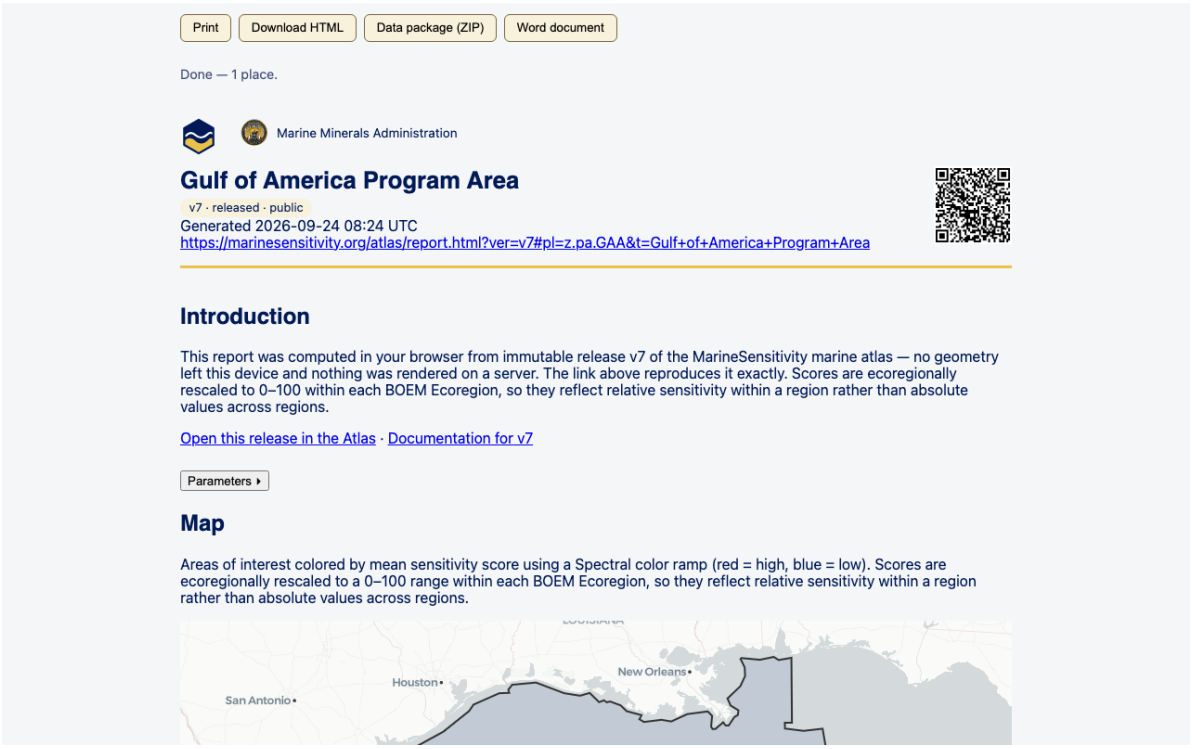


Figure B.12: The report document's header band and introduction: the MMA seal, a permalink and QR code, Parameters, and the start of the interactive map.

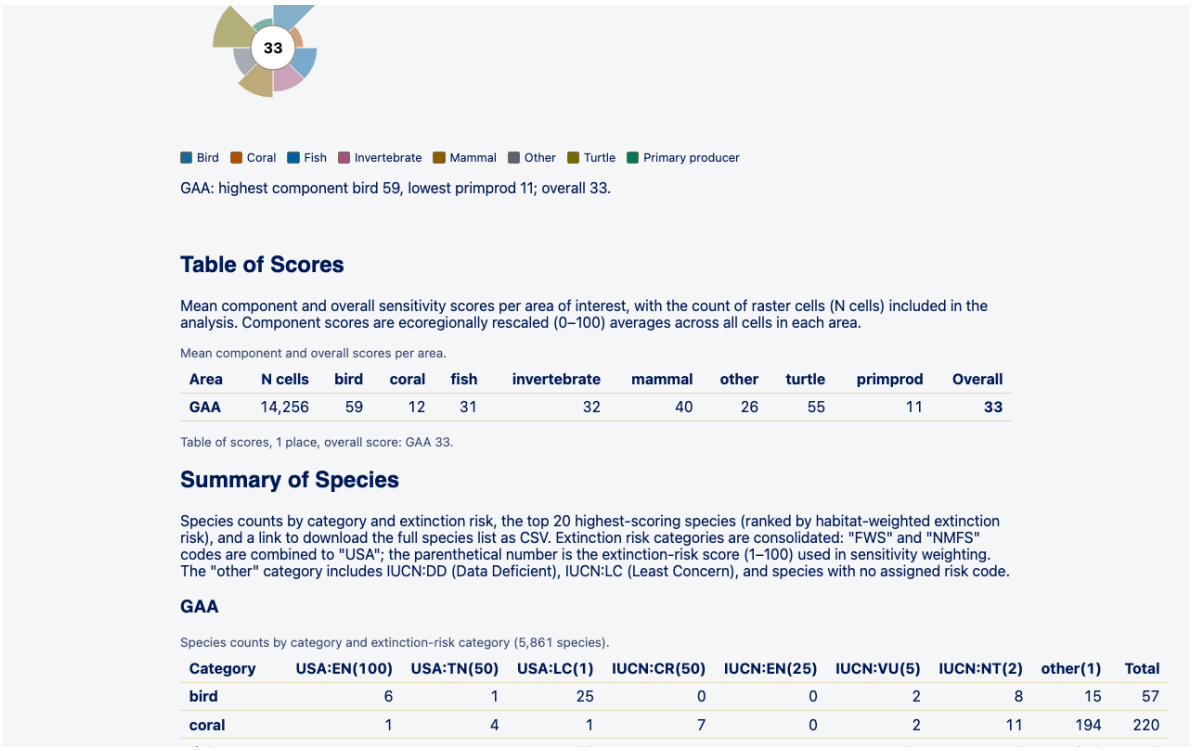


Figure B.13: The report's Table of Scores and the start of the Summary of Species, for the same Program Area.

the map is an embedded PNG); a **data package (ZIP)** — `README.md`, `CITATION.md`, `provenance.json`, `places.geojson` (drawn/uploaded places only, not zone places), a `scores_*.csv`/`species_*.csv` pair per place, and `query/*.sql`; and a **Word document (.docx)** — the same tables as coded Word styles, plus the map and a rasterized flower per place.

B.10 Accessibility and performance notes

The Atlas ships a written WCAG 2.1 conformance note and a performance budget report in its own repository. As of the note's own 0.10.24 (2026-09-23) evaluation the claim is **"partially supports,"** not full support, and it states plainly that **no screen reader was actually driven** in producing it — automated and manual keyboard testing stand in for that. Two skip links sit at the very top of the page: "Skip to the tools" and "Skip to the details panel." The critical-path JavaScript was measured (0.10.20, 2026-09-23) at ~**417 KB gzip** against a **450 KB** budget, plus ~**140 KB** for the MapLibre worker against **150 KB**; a separate gate asserted the species lens paints its first data pixel within 2.5 s on a laptop and 4 s on CI. Both budgets may have moved since; see the linked reports for current numbers.

B.11 When a service is down

Since 0.10.59, the Atlas checks two external services it does not control: **titiler-v8** (every score and species map tile) and the **S3 data origin** each release's own files are fetched from. Both are checked at boot and on a manual **Retry**; a map tile that actually fails to load for a real reason — a 5xx, a network error, or a timeout, never a normal "this release has no data here" 403/404 — re-checks only the tile server. It never polls in the background while everything is fine.

When either is down, a dismissible top banner names the failing host, the reason (timeout or HTTP status), and what still works — places, tables and reports already built from data already loaded, just no new tiles or new fetches from the down host; when it is specifically the **tile server** that is down, the legend also switches to a compact **"tiles unavailable"** state instead of a bare colour ramp. Built 2026-09-24 after titiler-v8 and the API were unreachable for an hour and the live app kept rendering a perfectly normal-looking map with no raster and no word to anyone.

What it cannot detect: a partial outage (some tiles or objects failing, others not — only a further real request surfaces it), a degraded-but-200 response (a tile that loads but paints wrong data), or a preview session's own restricted data prefix (the probe always checks the public bucket origin, not a signed-in override). No release the Atlas reads from calls a separate API today, so there is nothing of that kind to probe; DuckDB-WASM's own Parquet fetches,

PMTiles zone archives and the basemap are likewise outside this check's scope — a partial CDN problem affecting only one of those would not trip the banner.

B.12 Feedback and versions

Feedback opens a dialog: a kind (bug / idea / question / data), a title, a description, an optional email, an annotatable screenshot, and a privacy checkbox (off by default) for including the current view's link. **Send** is live: it posts to a Google Apps Script endpoint that writes a Sheet row, mails the team, and — for a **public** release — also files a public GitHub issue in `MarineSensitivity/atlas` with the screenshot attached (your email, if given, is never included in it); a **restricted** release never gets a public issue either way. An email, if given, is kept only in the team's Sheet row and mail (plus a copy to you), never in the issue (see `docs/feedback.md` in the atlas repo for the dialog's exact privacy wording). The original zero-backend fallback stays alongside Send, not replaced by it: **"Open as GitHub issue"** opens its own prefilled public issue, labelled by kind, with the screenshot copied to your clipboard to paste in — offered on every public release whether or not Send is configured, and the only option on a build with no endpoint set. Tell an analyst plainly: on a public release, both routes end in a public GitHub issue. A restricted release never offers the GitHub route; on a build with no endpoint configured it says plainly that "feedback cannot be sent from here right now."

GA4 receives interaction events as well as page views, and the app also logs those same events to the batched usage-log Sheet the Shiny `scores`/`species` apps already write (`app = "atlas"` distinguishes its rows) — never the URL fragment, which the app's own analytics code is source-scanned to never read.

Versions follow the same public/restricted split as the rest of the MST (see Chapter 2): a public release opens directly on the public host; a restricted pre-release's Atlas is reached only through the signed-in preview host, at its own `/{ver}/atlas/` path. Requesting a release the current host cannot show opens the version picker automatically: a restricted one offers "Continue on the preview host" (from 0.10.59, that link goes straight to the Atlas there), an unrecognized one reads "is not a version this host recognizes," and either way the promoted release loads underneath while the picker explains why. In the picker, every public release other than the current one offers **Switch**.

B.13 Known limitations (as of 0.10.59)

- For every release before v8 (v2–v7b), every input pill in the Species card still shows struck through, the leatherback's six included (Figure B.5) — as of this writing, the app bundles that would resolve the model-asset registry lookup (see Species lens above) for 10,247 of v7's 12,120 inputs have not yet been republished; once they land, only 1,873 of



Figure B.14: The release picker.

its 2,619 `rng_iucn` range inputs, which have no model-asset registry row (struck through in the Species Shiny app too), will still show this way.
- **GeoPackage** is refused (see the Places section above) whenever `ST_Read` fails to parse it, including a file with no vector layer — its declared-CRS and no-feature-table checks cannot actually run against an uploaded file, because DuckDB-WASM's `sqlite_scan()` cannot open a file registered only through `registerFileBuffer`, the way an uploaded GeoPackage is, so both checks silently fall through and `ST_Read`, or the normal out-of-range magnitude check, is left to catch a projected or empty file instead (still refused, just by a different, honest rule).
- A multi-Program-Area selection (several picks added as one place) sums area and means the composite for its list row, but its flower, component table and species table are not aggregated — the results panel asks you to select a single Program Area for those.
- **Show analysis cells** is not available for a Program Area place: the app bundle the Atlas reads carries each zone's count, area and scores but not its cells.
- The Atlas's report is a print-first document built and exported entirely client-side (HTML/ZIP/DOCX from the same source), not the Shiny apps' own report.

B.14 Generating Screenshots

The atlas repository's own `scripts/eyes-shots.mjs` captures the standard states most of this chapter's figures come from (18 numbered shots per viewport, about 11 states, phone 390×844 at 2x pixel density and desktop 1280×800, dark theme), run against a live or locally built app:

```
ATLAS_URL=https://marinesensitivity.org/atlas OUT=.tmp/eyes node scripts/eyes-shots.mjs
```

A few figures here (the Share dialog and its simplification ladder, the Coordinate dialog's refusal state, the version picker, and one Report scroll position) are not among that script's own states; they were captured by hand from the same live build, driving the app to the specific state each caption describes.

B.15 Source Code

`atlas` on its own `main` branch — one static build for every release, the same "one checkout, not one per version" convention the Scores and Species apps follow.

C Scores — mapper of aggregate sensitivity scores

D Scores

D.1 Overview

The **Scores** application (app.marinesensitivity.org/v3/scores/) is the primary interactive tool for exploring composite marine sensitivity scores. It renders **any published release** from the URL path — `/v3/scores/` shows v3 — defaulting to the promoted release when none is given; an older `?ver=v3` deep link still resolves, 301'd to the equivalent path. It enables users to visualize extinction risk weighted species presence at multiple spatial resolutions, with integrated flower plot and treemap summaries.

Figure D.1 shows the global view of cell-level sensitivity scores across all US waters. The sidebar provides controls for study area, spatial units (raster cells or program areas), and layer selection.

Figure D.2 shows the flower plot panel that appears when a program area is selected. Here, the Southern California program area displays a weighted mean score of 29, with petals representing species categories (bird, coral, fish, invertebrate, mammal, other, turtle) and primary productivity.

D.2 Features

D.2.1 Map Display

- **Cell-level view**: 0.05° grid cells colored by sensitivity score for the selected metric
- **Program area view**: aggregated scores by BOEM Program Area, with boundaries overlaid
- **Base maps**: multiple MapBox base layers (satellite, streets, light, dark)
- **Study area selection**: click a program area to focus the view and update summary panels

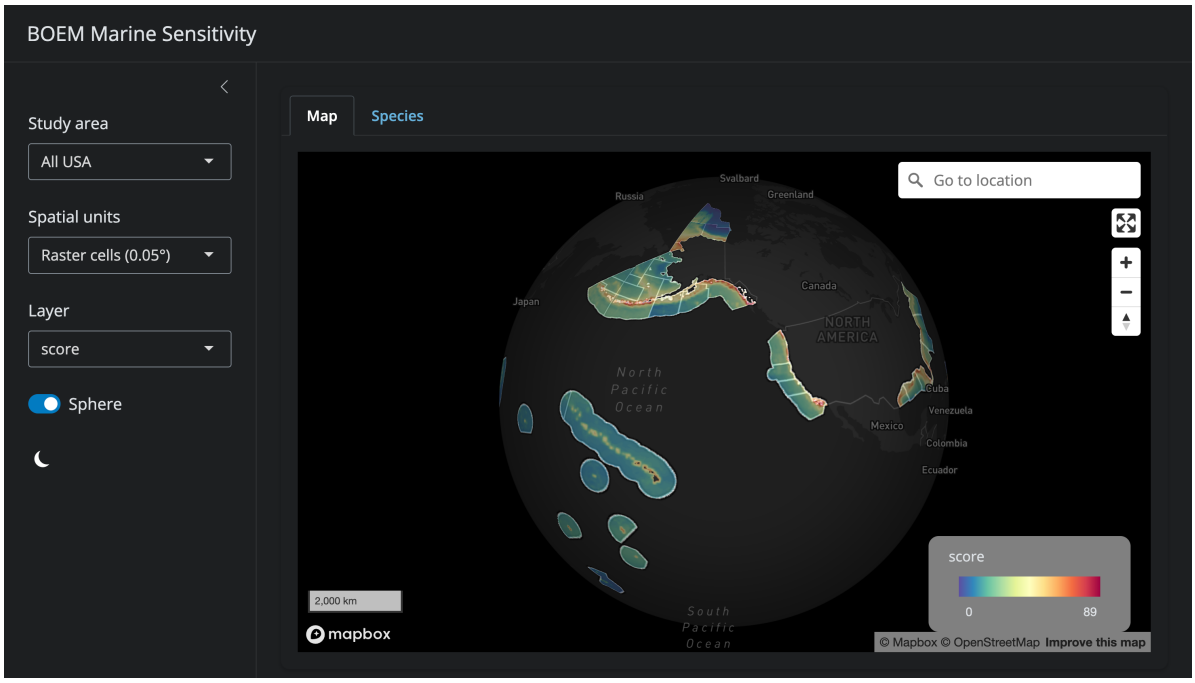


Figure D.1: The Scores application displaying cell-level sensitivity scores across US waters. The sidebar provides controls for study area selection, spatial resolution (raster cells vs. program areas), and layer choice. The globe view shows scores from 0 (blue) to 89 (red) with MapBox basemap.

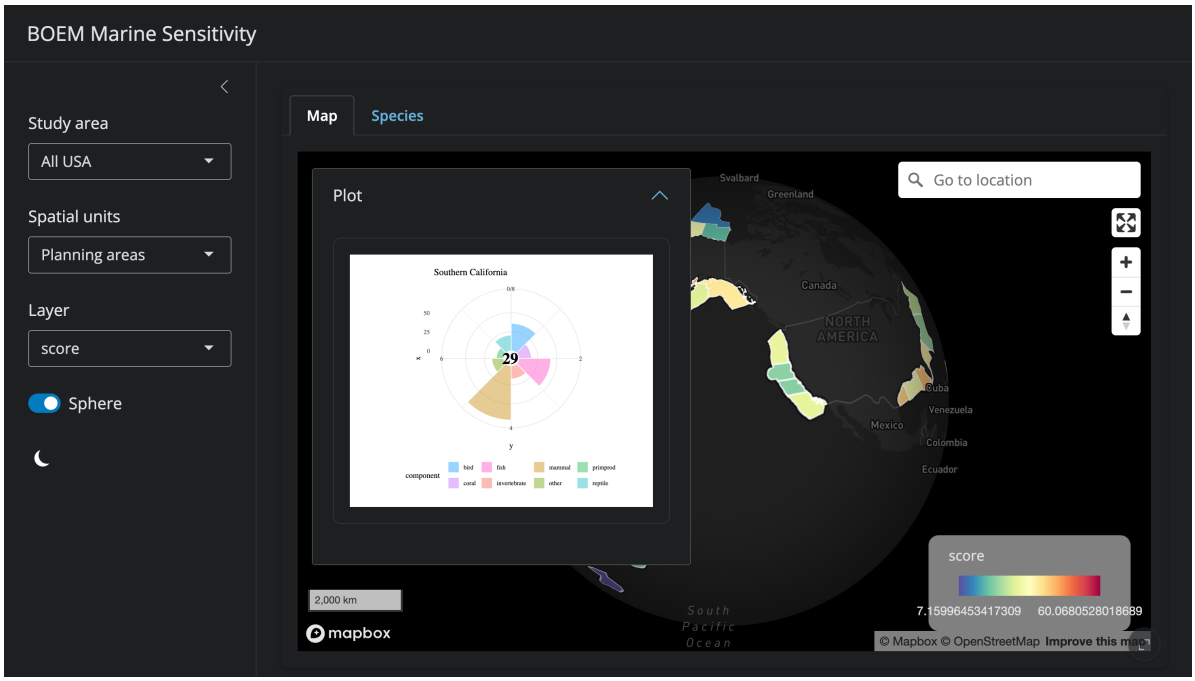


Figure D.2: The Scores application showing the flower plot panel for Southern California. Petal length represents each category's rescaled sensitivity score; petal width represents its weight. The center value (29) is the weighted mean across all categories.

D.2.2 Sensitivity Metrics

The map can display any of the following layers:

- Composite extinction risk (weighted sum across all species categories)
- Individual category scores, one per scored species category in the release being viewed
- Primary productivity

All scores are ecoregionally rescaled to [0–100] for meaningful cross-region comparison (see Chapter 10).

D.2.3 Flower Plot

Selecting a program area displays a flower plot summarizing the sensitivity profile:

- Each petal represents a species category
- Petal length = rescaled sensitivity score (0–100)
- Petal width = category weight
- Center value = weighted mean across categories

D.2.4 Species Table

A sortable table of all valid species present in the selected area, showing:

- Scientific and common name
- Species category
- ESA listing, IUCN Red List status
- MMPA/MBTA protection flags
- Extinction risk score

D.2.5 Data Export

Users can download CSV files of scores and species lists for the selected program area or custom area of interest.

D.3 Generating Screenshots

D.4 Source Code

`scores/app.R` on the `apps` repository's `main` branch. There is **one** checkout, not one per release: the version is data, resolved from the published manifest, so the per-version forks (`mapgl`, `mapgl_v1`…`v6`, `scores_v6`, `scores_v8`) are retired and their URLs 301 to the equivalent `/v{n}/scores/` path (an older `?ver=` deep link 301s the same way).

E Species — mapper of individual species distributions

F Species

F.1 Overview

The **Species** application (app.marinesensitivity.org/v3/species/) enables exploration of individual species distribution models. Like the Scores app it renders **any published release** from the URL path — `/v3/species/` shows v3; an older `?ver=v3` deep link still resolves, 301'd to the equivalent path. Users can view merged and source-level models, compare masked vs. unmasked distributions, and access detailed species regulatory status information.

Figure F.1 shows the merged distribution model for the blue whale (*Balaenoptera musculus*). The map displays cell values from 1 (low suitability, blue) to 100 (high suitability, red) across all US program areas where the species occurs.

F.2 Features

F.2.1 Species Selection

- **Search**: find species by common name or scientific name
- **Category filter**: narrow results by species category, as defined in the release being viewed
- **Alphabetical browsing**: scroll through species lists within each category

F.2.2 Model Display

- **Merged model**: the final merged distribution showing MAX values across all source datasets, with masking and spatial floors applied
- **Source datasets**: toggle individual source layers (AquaMaps, NMFS CH, FWS CH, FWS Range, BirdLife, IUCN Range) to see each dataset's contribution
- **IUCN masking toggle**: compare the unmasked AquaMaps SDM with the masked merged model to visualize the effect of range constraints

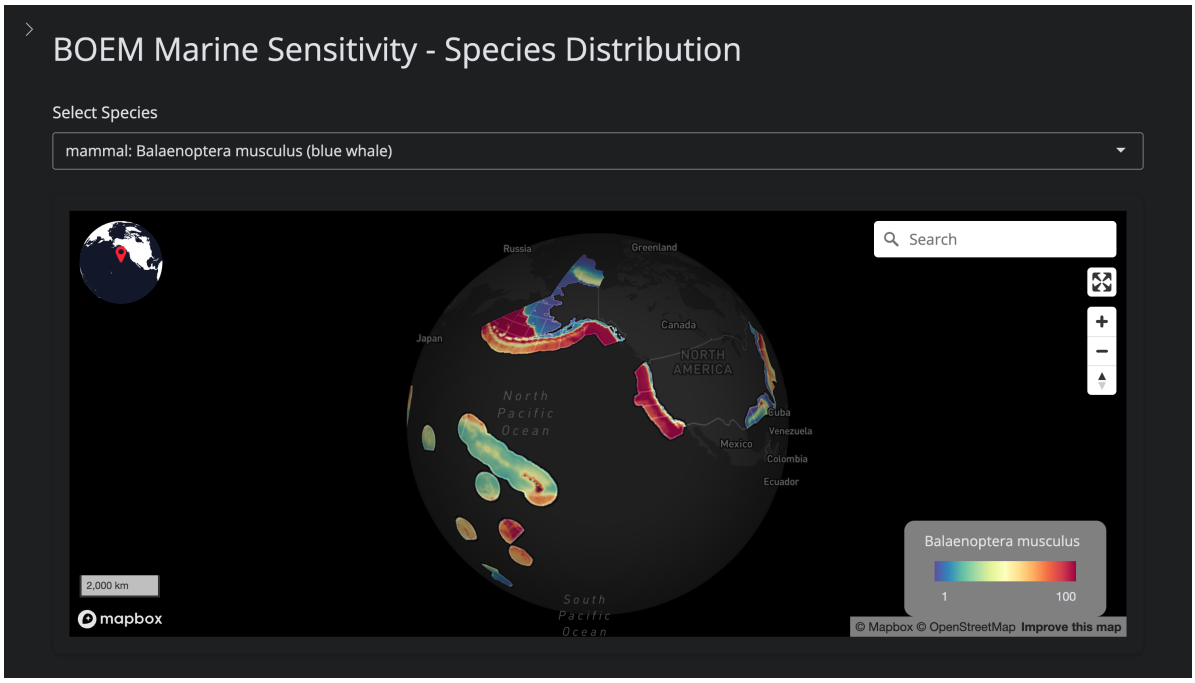


Figure F.1: The Species application showing the merged distribution model for blue whale (*Balaenoptera musculus*). Cell values range from 1 (low suitability) to 100 (high suitability). The species selector dropdown allows searching by common or scientific name with category prefix.

F.2.3 Species Info Panel

Detailed information for the selected species:

- Scientific name and common name
- Species category (bird, fish, mammal, etc.)
- ESA listing status (Endangered, Threatened, or not listed)
- IUCN Red List category
- MMPA protection flag
- MBTA protection flag
- Computed extinction risk score (1–100)
- Number of source datasets contributing to the merged model

F.2.4 Deep-Linking

The application supports URL parameters for direct linking to specific species, enabling bookmarking and sharing of specific views:

`https://shiny.marinesensitivity.org/species/?species=Balaenoptera+musculus`

F.3 Generating Screenshots

F.4 Source Code

`species/app.R` on the `apps` repository's `main` branch — one checkout serving every release, not one per version.

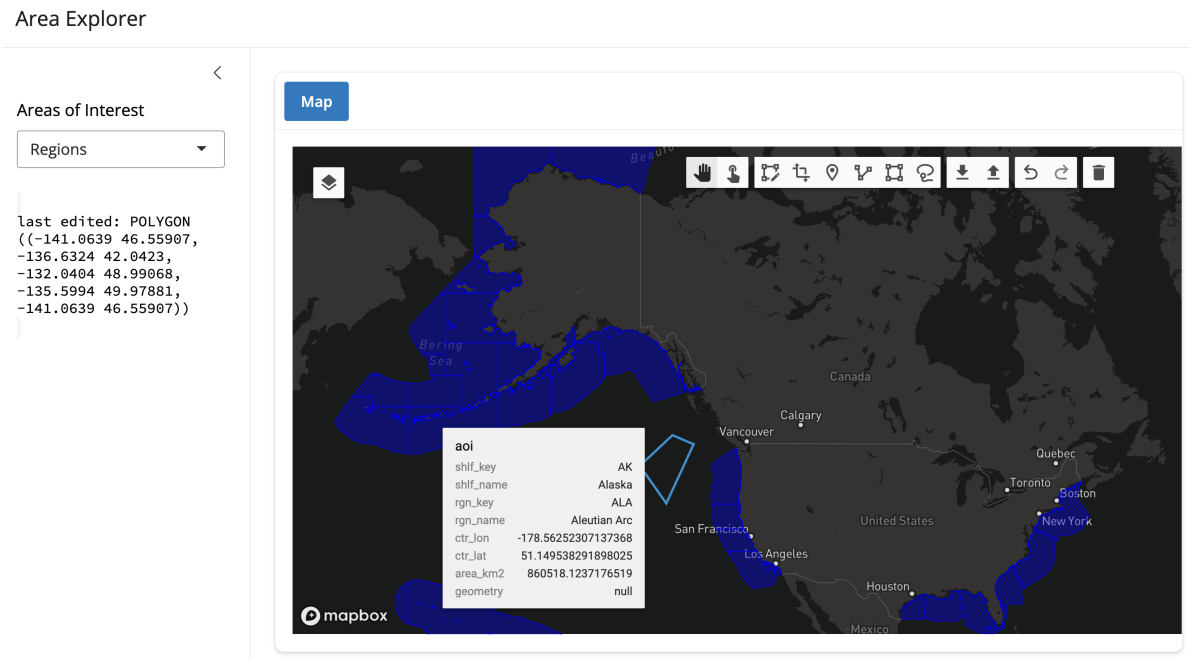
Part V

Applications (Legacy)

Areas of Interest

Map high resolution Areas of Interest (using vector tiles) for visualization (and later summarization).

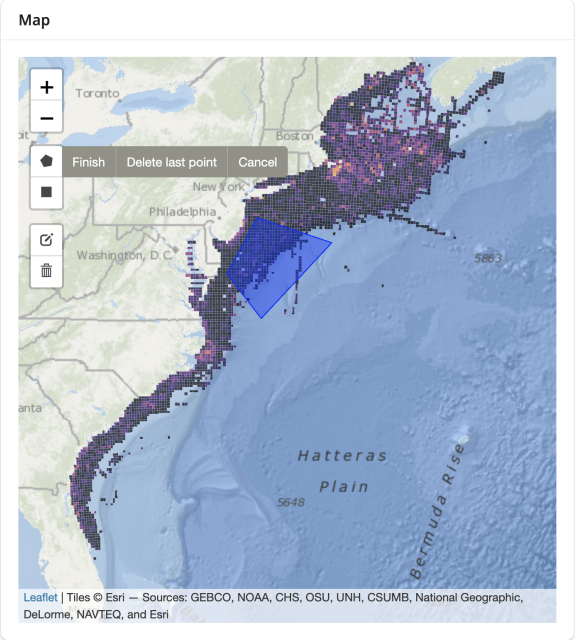
- website
- code

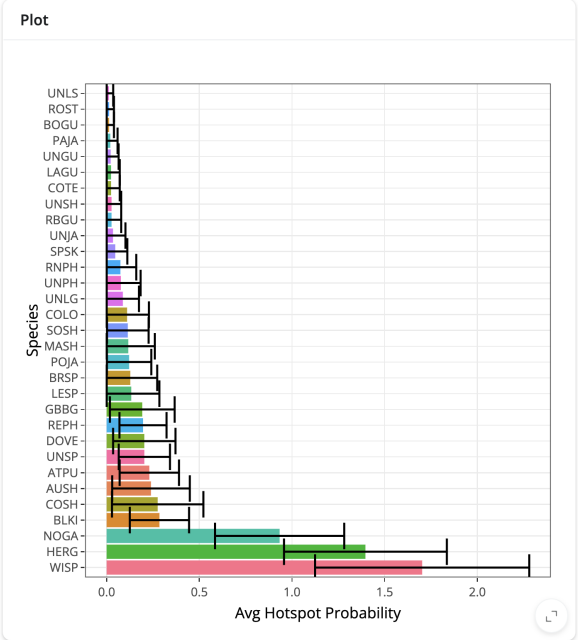


Bird Hotspots

Bird hotspots application showing hotspot probability for species present given drawn Area of Interest.

- website
- code

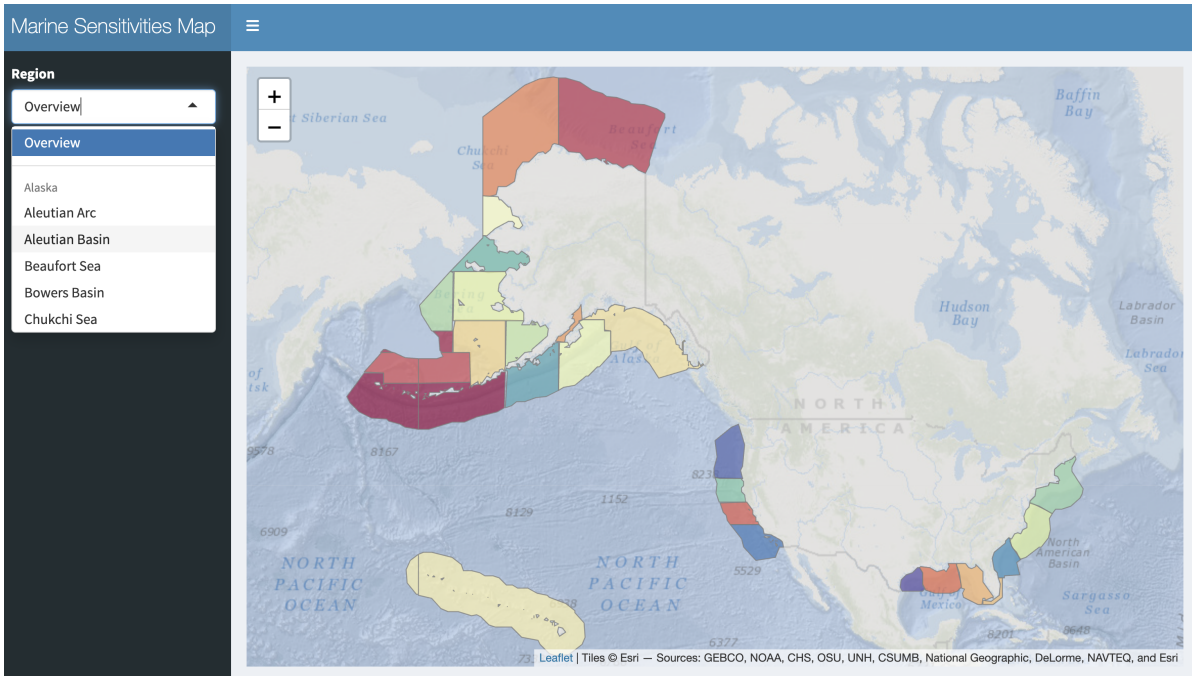




Regional Map

Basic interactive map of BOEM regions.

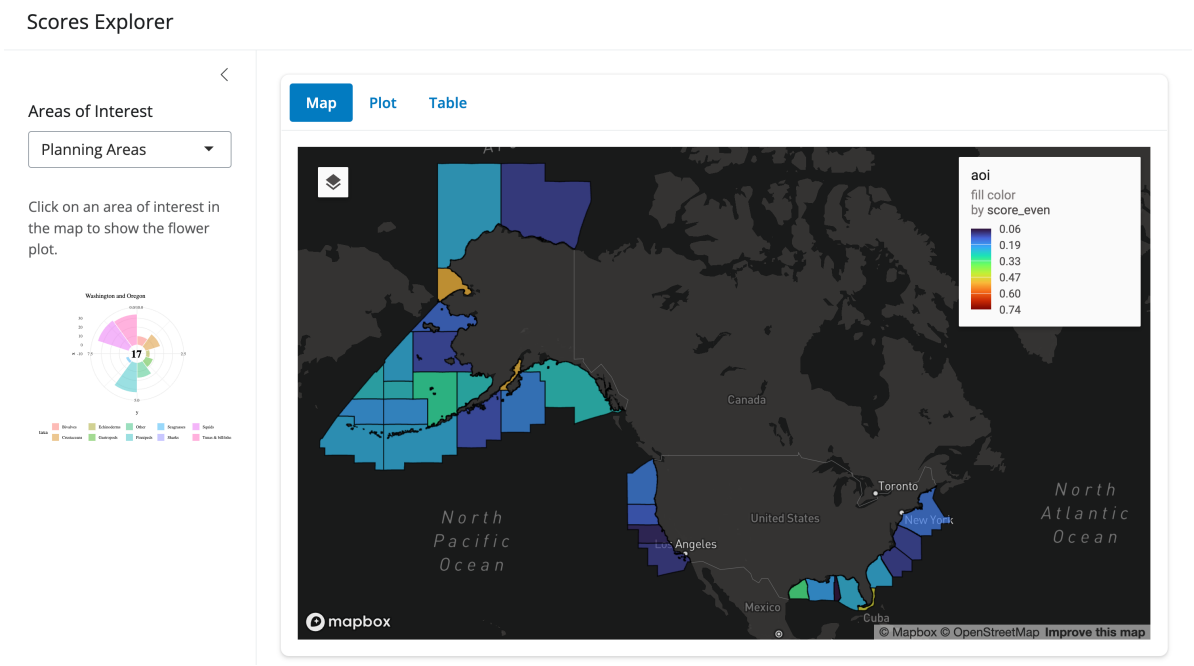
- website
- code



Scores Explorer

Calculate scores based on sensitivity metric – for now, extinction risk across taxonomic groups.

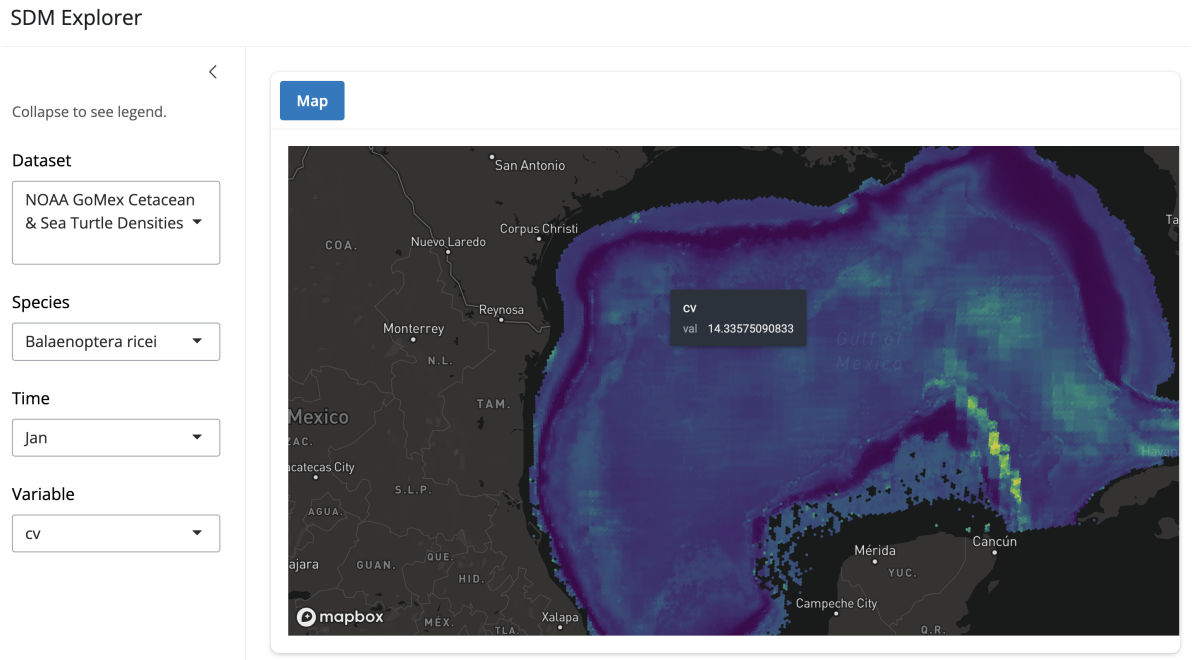
- website
- code



Distributions, Vector

Show species distributions with high resolution vectors (as vector tiles).

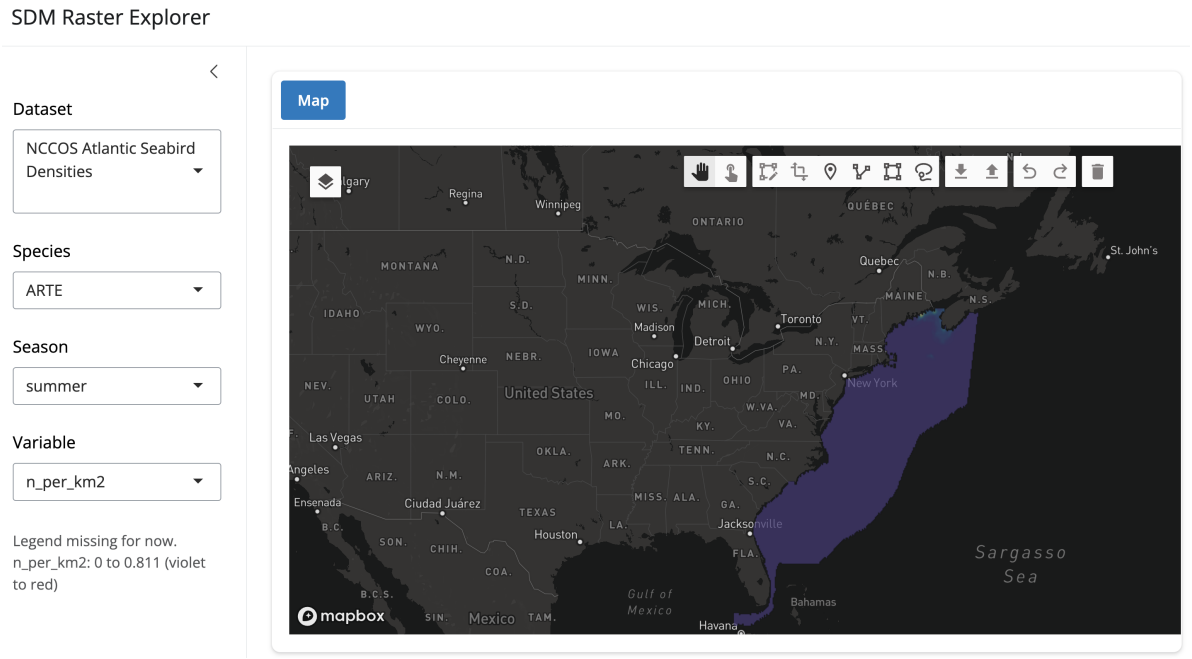
- website
- code



Distributions, Raster

Show species distributions with high resolution rasters as cloud-optimized GeoTIFFs (COGs).

- website
- code



Vulnerability Mapper

Combine species distribution models (raster) and vulnerability metrics (tables) to identify areas of high conservation concern.

- website
- code

