Building Open and Reproducible Geospatial Simulations with Python

DisSModel — Discrete Spatial Modeling Framework

Sérgio Souza Costa · LambdaGeo / UFMA

Cellular Automata | System Dynamics | LUCC · Mangroves | LUCC · CLUE

UFMA · INPE · Brazil Data Cube

AGENDA

What we will cover today

01	Contextualization	3 min
02	The Problem	6 min
03	DisSModel: A Discrete Spatial Modeling Framework for Python	5 min
04	The DisSModel Architecture	5 min
05	Laboratory Models	3 min
06	Empirical Models	4 min
07	What Do We Gain with DisSModel?	2 min
08	Next Steps	2 min

Career and Research Evolution



Focus on Spatial Modeling and Geographic Data (Open Science)

01 - CONTEXTUALIZATION

Geospatial Simulations

Land Use and Cover Change (LUCC)

LUCC models how and why the land surface changes over time. It is the mathematical language of human consequences on the territory.

Deforestation

Conversion of native forest to agricultural use.

Coastal Dynamics

Mangroves vs. urbanization and sea level rise.

Urban Expansion

Growth of cities over rural and natural areas.

Agricultural Frontier

Agribusiness advance over the Cerrado and the Amazon.

FOCUS: PROCESS-BASED GEOSPATIAL MODELING

Expectation Alignment

This presentation focuses exclusively on **Geospatial Models** instead of general-purpose Machine Learning models.

Key Brazilian Geospatial Tools

Dinamica EGO

UFMG · since 2002

Visual GUI platform for spatially explicit LUCC modeling.

STRENGTHS

- No-code visual interface
- Academic benchmark

LIMITATIONS

- Manual reproducibility
- Scalability issues

TerraME

INPE · since 2005

Lua simulation framework for cellular automata models.

STRENGTHS

- Native Agent/Space logic
- PostGIS integration

LIMITATIONS

- TerraLib is no longer supported
- TerraME last release (2020)
- No BDC/STAC support

LuccME

INPE · built on TerraME

Modular framework based on Demand/Allocation architecture.

STRENGTHS

- Interchangeable parts
- CLUE methodology

LIMITATIONS

- Inherits Lua constraints
- Weak Python link

02 — THE PROBLEM

The reproducibility crisis in computational science

Baker (2016), Nature · NAS (2019) · Stodden et al. (2016), Science · Krafczyk et al. (2021)

Baker, Nature 2016

70%

of researchers could not reproduce experiments from other groups.

NAS, 2019

National Academies, commissioned by the U.S. Congress. Formal report defining computational reproducibility as a pillar of modern science and establishing institutional recommendations.

Code doesn't run

Undocumented dependencies, incompatible versions, **implicit environments** — the experiment only works on the author's machine.

Inaccessible data

Input data not versioned or not published. **Without original data,** the result cannot be audited or replicated.

Opaque parameters

Critical model configurations are in ad hoc scripts or in the **researcher's memory**, not in formal auditable records.

Baker (2016) Nature 533:452 · NAS (2019) doi:10.17226/25303 · Stodden et al. (2016) Science 354:1240 · Krafczyk et al. (2021) Phil.Trans.R.Soc. 379:20200069

02 — THE PROBLEM

Challenges in Dynamic Geospatial Modeling

Spatial simulations have additional challenges beyond general computational reproducibility

01

Geospatial data is large and volatile

Rasters and vectors with specific CRS, varying resolutions and version history. A different GeoTIFF silently masks calibration errors.

02

Models couple science and infrastructure

The CA or LUCC logic gets mixed with file I/O, visualization and loop control — impossible to reuse without rewriting.

03

Experiments are not traceable

Which model version? Which parameters? Which input dataset? Without formal records, the result is irreproducible by design.

04

Scale breaks the scientific workflow

The researcher calibrates in the notebook, but running scenarios at scale requires rewriting the code — or abandoning production execution.

02 — THE PROBLEM

Why doesn’t LambdaGEO propose modernizing TerraME?



	TerraME / LuccME	DisSModel (The Proposal)
Language	Lua + TerraLib (no longer maintained)	**Pure Python (Geopandas, NumPy)**
Reproducibility	No formal record	**Git + SHA-256 per execution**
Cloud Integration	Isolated local execution	**Cloud Native (API, Minio, Docker)**

DisSModel preserves the TerraME modeling philosophy in a modern ecosystem.

DisSModel

The Researcher's Journey: End-to-End Workflow

Modules of the @dissmodel Organization

PHASE 1 DEVELOPMENT & PACKAGING

1. Creation

Model design using the base library (model, environments, geo models, rules).

2. Packaging

Logic is fully encapsulated in *executors* (validate, load, execute, save).

3. Tests

The model is validated locally in an incredibly agile way via CLI tool.

4. Publishing

Library source code made available in independent GitHub repositories.

PHASE 2 INTEGRATION & EXECUTION

5. Integration

Researcher creates a .toml manifest and submits a PR to the catalog.

6. Curation

Maintainers review compliance and security before official incorporation.

7. API Execution

Platform manages dependencies and exposes results via dynamic API.

"Science should not need to be rewritten to go into production." **— DisSModel Principle**

DisSModel

Core Pillars of DisSModel

Essential foundations for robust and scalable scientific modeling.

Reproducible Research

Sharing data and code for external verification.

IMPLEMENTATION:

- Executor + TOML: separation of code/params
- Docker: isolated execution
- SHA-256: auditable records

Claerbout (1992) · Stodden (2016)

FAIR Software

Findable, Accessible, Interoperable, and Reusable.

KEY PILLARS:

F: Zenodo DOI + reprod. package
A: REST API + MinIO/S3
I: GeoPandas + Xarray + GeoTIFF
R: Versioned TOML manifest

Wilkinson et al., 2016

Software Engineering

Bridging Scientists and Production Engineers.

LAYER SEPARATION:

- **Science:** logic remains pure
- **Platform:** APIs and infrastructure
- **SimOps:** Scientific Operations

Baxter et al. (2012) · Wilson (2017)

"Science should not need to be rewritten to go into production." **— DisSModel Principle**

DisSModel

Main Repositories

Modules of the @dissmodel Organization

Base

3 repositories

Provides the model's core engine, platform infrastructure, and official catalog.

Lab Models

2 repositories

Focused on cellular automata and dynamic systems for testing, teaching, and learning.

Empirical Models

2 repositories

High-fidelity models previously implemented in TerraME, now available as reproducible Python workflows.

Interfaces

1 repository

A QGIS plugin that runs the BR-MANGUE model natively — no API calls, no middleware, simulation inside the GIS environment.

Visit our documentation at: **https://dissmodel.github.io/**

DisSModel

Base Components

🧩 dissmodel

LIBRARY (Core Framework)

Central simulation engine for local scientific model development and execution.

`github.com/DisSModel/dissmodel`

```
pip install dissmodel
```

⚙️ dissmodel-platform

PLATFORM (Cloud-Native)

Open and scalable infrastructure for distributed cloud experiment execution.

`github.com/DisSModel/...-platform`

```
docker compose up --build
```

📁 dissmodel-configs

MODEL CATALOG

The single source of truth. Centralizes configuration files (manifests in .toml).

`github.com/DisSModel/...-configs`

```
pull requests
```

*"Science should not need to be rewritten to go into production" — **dissmodel** runs the same code locally and on the remote **platform**.*

DisSModel

Core Framework

core

Simulation Engine

Discrete clock + execution cycle via Environment & Model.

Environment, Event dispatch, step control.

geo

Spatial Substrate

Dual design: Vector (GeoDataFrame) + Raster (NumPy).

Queen/Rook and Moore/Von Neumann neighborhoods.

executor

Execution Layer

ModelExecutor ABC + SimulationRegistry.

Local CLI and Remote Workers — same code.

io

Storage I/O

Local file and MinIO/S3 read/write operations.

Checksums and provenance.

vis

Visual Observers

Chart, Map, RasterMap, display_inputs.

Decoupled Observer pattern.

"Science should not need to be rewritten to go into production" — **DisSModel Principle**

DisSModel

DisSmodel-platform - Server Side

FastAPI Gateway

API-Driven Execution

- Programmatic REST calls
- Legacy dashboard integration
- Validation via Pydantic

Docker Workers

Subprocess Isolation

- Jobs in isolated containers
- No dependency conflicts
- Redis Queue scalability

Jupyter Lab

Experiment Service

- Interactive experimentation
- Direct data visualization
- Rapid model prototyping

MinIO Storage

Agnostic Storage

- S3-compatible integration
- Unified spatial writing
- Zarr and GeoTIFF support

The infrastructure orchestrates **FastAPI, Redis, and Docker** *to ensure science does not need to be rewritten for production.*

Architecture deployable via **Docker Compose** in Cloud environments.

DisSModel

DisSModel-configs - The Central Catalog:

Single Source of Truth

To keep the ecosystem scalable and secure, we separate the model code from its publication.

The catalog stores only declarative manifests, ensuring full traceability and governance.

Manifestos .toml
Each model has a centralized configuration file.

Versioning
Points to the specific *release* or *tag* on GitHub.

Parameterization
Defines default limits consumed by the API.

configs/models/coastal_raster.toml

```toml
[model]
executor_module = "br-mangue.executor"
name = "br-mangue"
package = "git+https://github.com ... "

[model.parameters]
end_time = 88
taxa_elevacao = 0.5
altura_mare = 6.0
acrecao_ativa = false
resolution = 100.0
crs = "EPSG:31984"
```

Declarative configurations ensure exact simulation reproducibility.

DisSModel

Practical Examples & Implementations

Now that the base repositories and server-side architecture are established, let's explore the practical laboratory models built upon this foundation.

Spatial Models

Complex systems using Cellular Automata (CA)

- Localized spatial interactions
- Game of Life & Forest Fire
- Built on GeoDataFrames

Temporal Dynamics

System Dynamics (SysDyn) Framework

- Flows, stocks, and feedback loops
- SIR & Predator–Prey models
- Interactive Streamlit explorers

Cellular Automaton: Game of Life

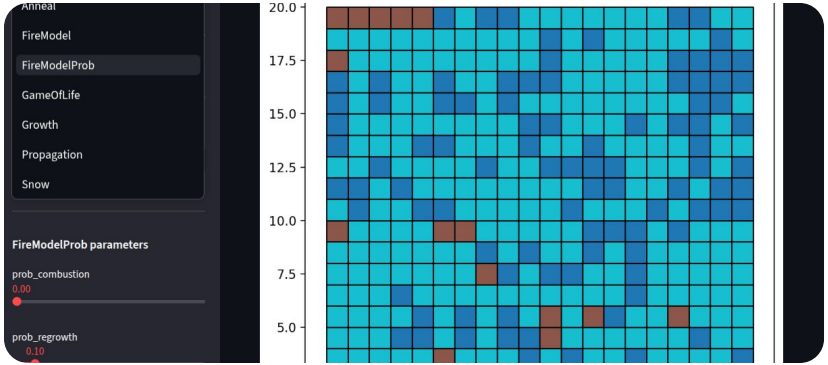
DisSModel

Model Definition

```python
class GameOfLife(CellularAutomaton):
    def setup(self) -> None:
        self.create_neighborhood(strategy=Queen)

    def rule(self, idx) -> int:
        state = self.gdf.loc[idx, 'state']
        n = self.neighbor_values(idx).sum()
        if state == 1:
            return 1 if 2 <= n <= 3 else 0
        return 1 if n == 3 else 0
```

Model Execution & Results




```
gdf = vector_grid((20, 20), attrs={"state": 0})
env = Environment()
GameOfLife(gdf).initialize()
Map(gdf, column="state")
env.run(steps=10)
```

Explore the full implementation at **github.com/DisSModel**

DisSModel

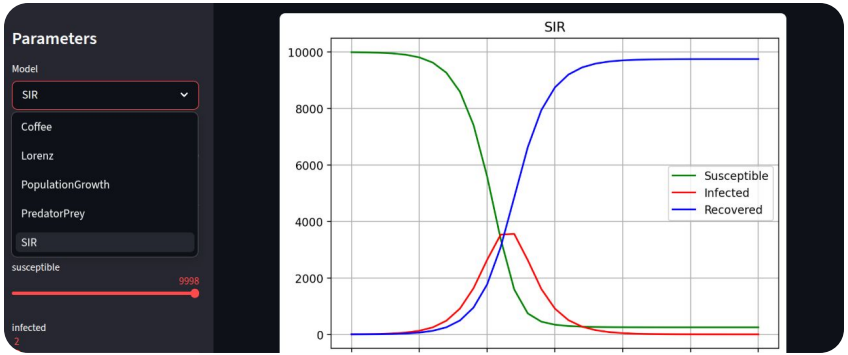
System Dynamics: SIR Model

Model Definition

```
class SIR:
  def execute(self):
    total = S + I + R
    alpha = contacts * prob
    new_inf = I * alpha * (S/total)
    new_rec = I / duration

    S -= new_inf
    I += new_inf - new_rec
    R += new_rec
```

Results & Implementation




```
env = Environment()
SIR(9998, 2, 0, 2, 6, 0.25)
Chart(show_legend=True, show_grid=True, title="SIR Model")
env.run(30)
```

Explore the full implementation at **github.com/DisSModel**

DisSModel

Real-World Empirical Models



From Theory to Application

Transitioning from laboratory simulations to complex, empirical scenarios. Explore how DisSModel addresses coastal dynamics and land use change in real-world environments.

Coastal Management

Modeling mangrove migration and flooding risks. Analysis of sea-level rise scenarios in the Maranhão Coast.

Land Cover Change

Simulating agricultural expansion and urban growth. Empirical validation with real-world land use datasets.

Coastal Case Study (brmangue)

📍
Study Area
Maranhão Coast, Brazil. Vector spatial grid.

📄
Scientific Basis
Bezerra et al. (2014).

🌊
FloodModel
Flooding due to sea level rise scenarios.

🌿
MangroveModel
Migration by tide and soil type dynamics.

```
env = Environment(start_time=2008, end_time=2030)

# Add specialized models
FloodModel(gdf=gdf, elevation_rate=0.5,
use_attr='usage')
MangroveModel(gdf=gdf, tide_height=6.0,
soil_attr='soil')

env.run()
```

Performance Benchmark

SUBSTRATE | MS/STEP | SPEEDUP

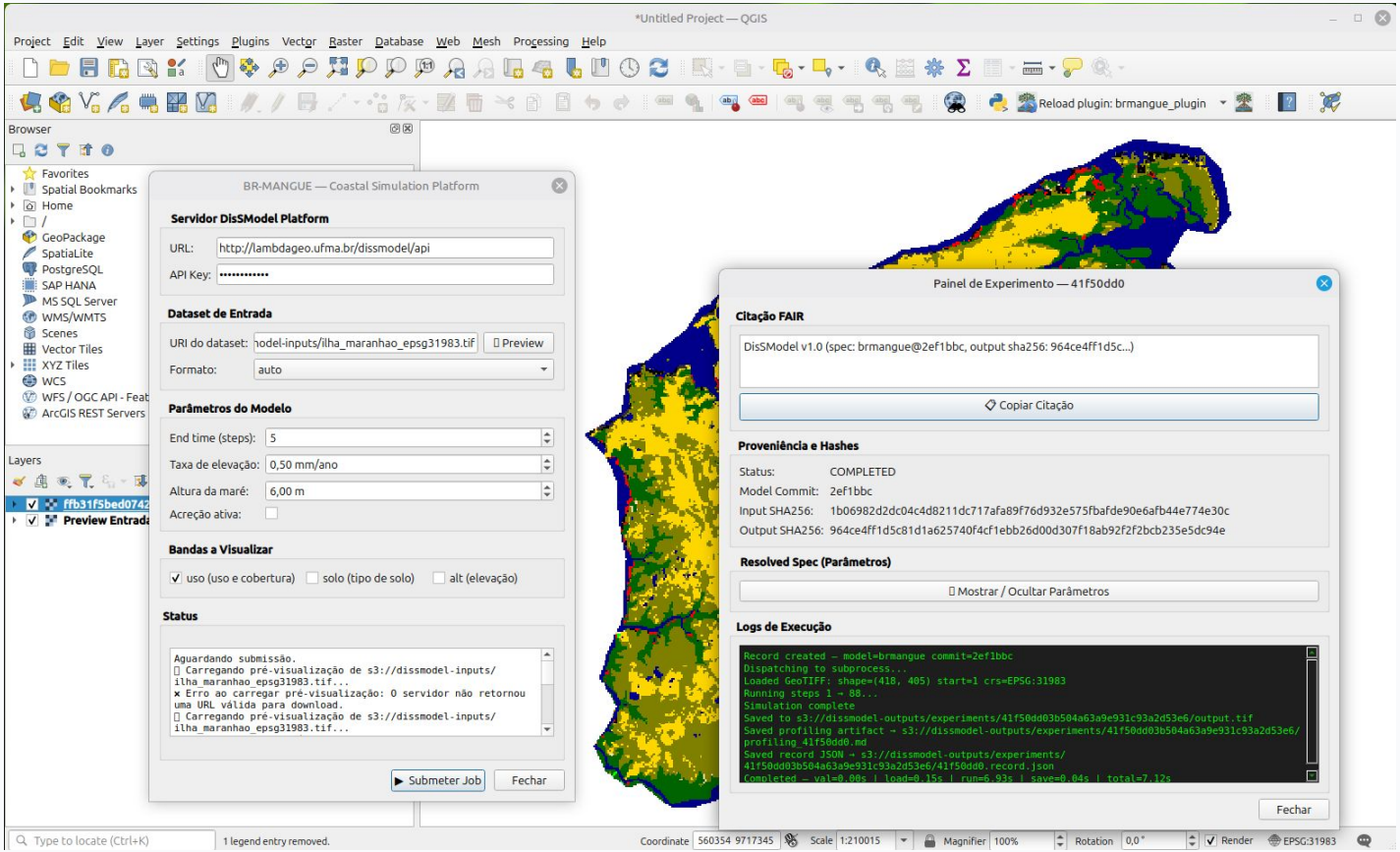
Vector: **1241.7 ms** (1.0x)

Raster: 31.5 ms (39.4x ⚡)

Scale: 97,309 cells

Accuracy vs TerraME: **99.8% Match**

Coastal Case Study (brmangue)



Plugin Feature

QGIS Integration

Dedicated plugin for the brmangue model, enabling streamlined simulation and visualization of coastal dynamics within the QGIS environment.

- Intuitive interface for parameter input
- Direct visualization of model outputs
- Seamless integration with vector grids

DisSLUCC-Continuous: Land Use & Cover

Continuous Modeling

The package implements the CLUE-type allocation algorithm (Veldkamp & Fresco, 1996). Cells receive *fractional proportions* instead of discrete classes.

Evaluation Metric

For continuous outputs, the **MAE (Mean Absolute Error)** was adopted to directly evaluate modelling error (Willmott, 2005).

1. Demand

Magnitude of change per timestep. Pre-computed via historical CSV.

```
DemandPreComputedValues(
 annual_demand=load_csv( ... )
)
```

2. Potential

Change suitability via linear regression with driver variables.

```
PotentialLinearRegression(
 gdf=gdf, drivers=[ ... ]
)
```

3. Allocation

CLUE-like distribution based on potential and constraints.

```
AllocationClueLike(
 demand, potential
)
```

Experimental Setup

DATASET	TIME STEPS	RESOLUTION
LuccME AC (2008-2014)	**6 Steps**	**6,574 cells**

DisSLUCC - Validation and Performance Results

Performance (6 steps · 6,574 cells)

3.9×

SPEEDUP RASTER VS VECTOR
(31.6 ms vs 122.5 ms)

Accuracy vs TerraME

0.002276

MAE VS TERRAME · BOTH SUBSTRATES
Vector = Raster · MAE 0.002276

Technical Highlight: Raster substrate achieves 3.9× speedup over Vector (31.6 ms vs 122.5 ms/step). Both substrates produce identical output (MAE = 0.000) compared to each other, confirming the raster optimisation preserves full algorithmic correctness.

Dataset: LuccME Package - AC - 2008–2014 · 6 steps · 6,574 cells · tolerance 0.01

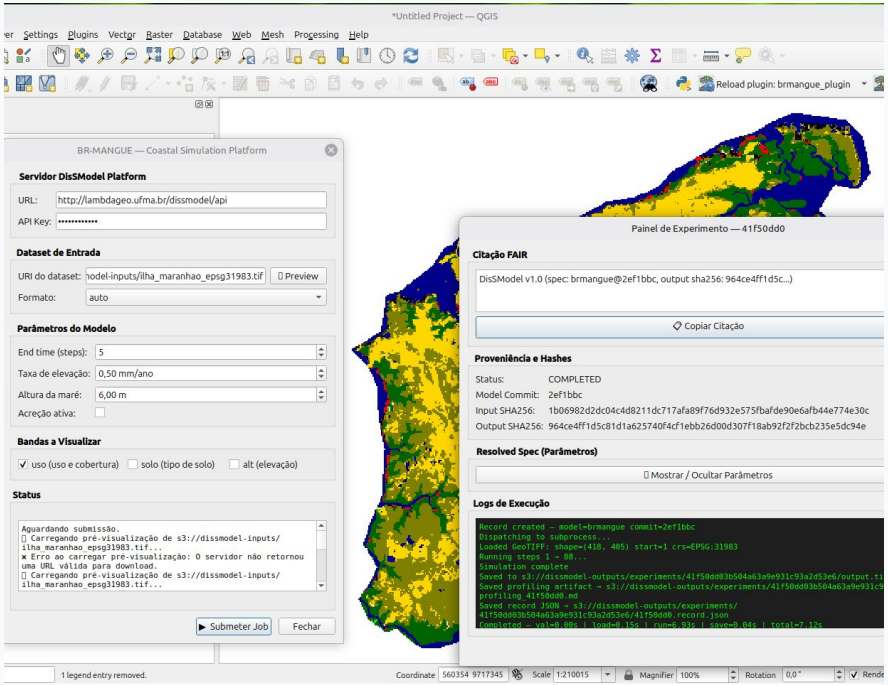
TRANSFORMING GEOSPATIAL SCIENCE

Hidden Engineering
The complex cycle of data orchestration runs seamlessly.

Full Traceability
Guaranteed data provenance for every simulation stage.

Direct Delivery
Dynamic results delivered directly within GIS tools.

End-to-End Focus
Researchers focus on science, users on decisions.



Transparency, reproducibility, and usability at scale

08 — NEXT STEPS

DisSModel Roadmap 2026–2028

2026 S1

JOSS Publication

Repo organisation & PyPI v0.5

Formalisation at UFMA

Research plan submissions

2026 S2

Project Kickoff

CLUE & CLUE-S (PIBIC)

DisSLUCC (PIBIC)

BRMangue JOSS

DisSModel v1.0

2027 S1

BDC Integration

STAC source reader

Xarray (time, y, x)

ExpRecord → MinIO

2027 S2

Dask / Ensemble

Lazy eval (Pangeo)

Parallel scenarios

Native Zarr output

2028

Book + Community

Geospatial DS book

PT-BR Edition

Workshop INPE/SBSR

** QGIS plugins (qgisparql-triple2layer / layer2triple): joint JOSS submission under evaluation.*

Beyond the Technology: Governance

The platform works. To deliver on its promises, it needs institutional backing.

Infrastructure

Who maintains the platform? Server, storage, updates, and uptime.

Data policy

Where do results live? For how long? Integration with Brazil Data Cube?

Model catalog

Who reviews Pull Requests? What are the acceptance criteria?

Community

If each group runs locally, reproducibility problems remain.

Technology solves the ***how***. Governance solves the ***who*** and the ***where***. The INPE partnership for LuccME++ transforms DisSModel from a lab tool into shared scientific infrastructure.

08 — NEXT STEPS

Strategic Partnership: DisSModel & LuccME++

2027/28 — DisSModel will serve as the innovation laboratory for the evolution of **LuccME** at CCST/INPE.



DisSModel

Partnership & Evolution



Future of LUCC

TerraME was the starting point — **DisSModel advances as the foundation for LuccME++**

Thank you!

DisSModel — open source, open science

GitHub
`dissmodel.github.io`

Docs
`dissmodel.github.io/dissmodel/`

PyPI
`pip install dissmodel`

Contact
`sergio.costa@ufma.br`

> *"Science should not need to be rewritten to go into production."*
> — DisSModel Principle

`dissmodel` `dissmodel-ca` `dissmodel-sysdyn` `brmangue-dissmodel` `DisSLUCC-Cont.`

PPGC&TAmb — Programa de Pós-Graduação em Ciência e Tecnologia Ambiental

DisSModel

LAMBDAGEO

SEMPRE+ Qualidade Inovação Inclusão — UFMA Universidade Federal do Maranhão

References

Modeling

Aguiar, A. P. D. et al. (2016). Modeling the spatial and temporal heterogeneity of LUCC: the LUCCME framework. *Env. Modelling & Software.*

Bezerra, D. S. et al. (2014). Sea level rise impacts on mangroves. *Ciência e Natura.*

Carneiro, T. G. S. et al. (2013). An extensible toolbox for modeling nature-society interactions. *Env. Modelling & Software.*

Pontius Jr., R. G. & Millones, M. (2011). Death to Kappa. *Int. Journal of Remote Sensing.*

Soares-Filho, B. S. et al. (2002). DinamicaEGO — a stochastic cellular automata model. *Ciência e Agrotecnologia.*

Veldkamp, A. & Fresco, L. O. (1996). CLUE: a conceptual model to study land use change. *Ecological Modelling.*

Verburg, P. H. et al. (2002). Modeling the spatial dynamics of regional land use. *Environmental Management.*

Zeigler, B. P. et al. (2018). Theory of Modeling and Simulation. *Academic Press.*

Geospatial Python & Data

Camara, G. et al. (2021). The Brazil Data Cube project. *ISPRS Annals of the Photogrammetry, Remote Sensing.*

Jordahl, K. et al. (2021). geopandas/geopandas. *Zenodo.*

Rey, S. J. & Anselin, L. (2010). PySAL: A Python library of spatial analytical methods. *Review of Regional Studies.*

Simoes, R. et al. (2023). Satellite Image Time Series Analysis. *Remote Sensing, 13(13), 2428.*

References

Open Science & Reproducibility

Baker (2016): 1,500 scientists lift the lid on reproducibility. Nature.

Baxter et al. (2012): The Research Software Engineer. Digital Research Conf.

Claerbout & Karrenbach (1992): Electronic docs give meaning to research. SEG.

Gentleman & Lang (2004): Statistical analyses and reprod. research. Bioconductor.

Krafczyk et al. (2021): Learning from reproducing results. Phil. Trans. Royal Soc. A.

Lamprecht et al. (2020): FAIR principles for research software. Data Science.

Scientific Software & FAIR Principles

Marwick et al. (2018): Packaging data analytical work. The Am. Statistician.

Moreau & Missier (2013): PROV-DM: The PROV Data Model. W3C.

National Academies (2019): Reproducibility and Replicability in Science. NAP.

Stodden et al. (2016): Enhancing reproducibility. Science.

Wilkinson et al. (2016): The FAIR Guiding Principles. Scientific Data.

Wilson et al. (2017): Good enough practices in computing. PLOS Comp. Bio.