EAA BUDAPEST
2022 31 AUG - 3 SEP
[RE]INTEGRATION

Developing best practices for an open library of archaeological ABM modules

Lessons learned from other initiatives

A. Angourakis, F. Riede, I. Romanowska, P. Verhagen, M. Saqalli, D. Taelman, M. Vlach, K. Sikk, J.M. Galán, T. Brughmans

NA
S²A
NETWORK FOR
AGENT-BASED MODELLING OF
SOCIO-ECOLOGICAL SYSTEMS
IN ARCHAEOLOGY

Outline

1. **Other experiences**
    1. Code libraries
    2. Simulation platforms
    3. Other open science/source/data initiatives
2. **Best practices principles**
    1. Collaboration
    2. Workflow
    3. Impact and sustainability
3. **Our choices (so far)**
    1. On code
    2. On metadata
    3. On documentation
    4. On version control and submission management
4. **Open questions**

1.
Other experiences

1.1. Code libraries

CoMSES Model library - https://www.comses.net/

NetLogo User Community Models - https://ccl.northwestern.edu/netlogo/models/community/

R CRAN - https://cran.r-project.org/

NASA Open Source Software - https://code.nasa.gov/

Domain-specific

Agriculture:

- Decision Support System for Agrotechnology Transfer (DSSAT) - https://dssat.net/
- SWAT - https://swat.tamu.edu/

Physics:

- MechSys - https://mechsys.nongnu.org/index.html

1.2. Simulation platforms (for or including ABM)

Libraries/packages (no stand alone, no GUI):

- SimPy (Python) - https://simpy.readthedocs.io/en/latest/contents.html
- Mesa (Python) - https://mesa.readthedocs.io/
- Agents.jl (Julia) - https://juliadynamics.github.io/
- LSD (C++) - https://www.labsimdev.org/wp/

Free & Open-source platforms:

- NetLogo - https://ccl.northwestern.edu/netlogo/
- Repast - https://repast.github.io/
- MASON - https://cs.gmu.edu/~eclab/projects/mason/
- GAMA - https://gama-platform.org/
- TurtleSpaces - https://turtlespaces.org/
- PhET Interactive Simulations - https://phet.colorado.edu/

Commercial platforms:

- AnyLogic - https://www.anylogic.com/
- MathWorks' Simulink - https://www.mathworks.com/solutions/system-design-simulation.html?s_tid=hp_solutions_system

1.3.
Other open science/source/data initiatives

Archaeological data:

- Journal of Open Archaeological Data - https://openarchaeologydata.metajnl.com/
- The Digital Archaeological Record (tDAR) - https://core.tdar.org/
- Archaeological Data Service (ADS) - https://archaeologydataservice.ac.uk/
- ARIADNE EU - https://www.ariadne-eu.org/

Generic data repositories:

- Zenodo - https://zenodo.org/
- Open Science Foundation (OSF) - https://osf.io/
- Humanities Commons - https://hcommons.org/
- Wikipedia (Wikimedia Foundation) - https://wikimediafoundation.org/

Git hosts:

- GitHub - https://github.com/
- GitLab - https://about.gitlab.com/
- Bitbucket - https://bitbucket.org/

Code Ocean - https://codeocean.com/

2.
Best practices principles

2.1. Collaboration

Avoid *overplan*, *overdo* or *oversell*

Communication with “core” group

- Single channel
- Bureaucracy matching scale
- Clarity of objective and TO-DOs
- Clarity of roles
- “Speak your truth”

Communication with wider community (users/authors)

- Multiple channels
- Inviting attitude
- Clarity of achievements
- Clarity of open roles

2.2. Workflow

Progressive immediatism

Every closeable outcome must be packed and made available in the most open and publicly way possible.

Modularity of outcomes

Outcomes should be kept bite-sized and relatively separable from the other outcomes.

Redundancy of outcomes

Do not discourage or correct the production of outcomes whose content overlap. It can help both the promotion and long-term sustainability of the redundant content.

Minimise external dependency

Make so that the outcomes remain usable and useful despite potential changes in the larger infrastructure context. Detect fragilities and prepare contingencies.

Multilateral revision

Outcomes should be subjected to a peer-review process.

2.3.
Impact and sustainability

FAIR principles

All data stored (code or else) should strive to be findable, accessible, interoperable, and reusable, as far as possible.

Free open distribution

No financial transaction is involved in the submission and use of the library.

Target users

Define and aim for a target audience first, then expand if opportunities are encountered.

Indicators of use

Measure use/access to the library. Get to know your actual users.

Feedback

Query the community of users, especially long-timers, to set pathways for further improvements.

3.
Our choices (so far)

3.1.
On code

Naming

Single-responsibility principle

No magic numbers

Exposed inputs and outputs

Keep it basic (minimise dependencies)

Commentary in code

3.2.
On metadata

The **NASSA.yml** file:

- A central configuration file that defines the module as a unit
- Contains all relevant meta-information in a well-specified format
- Allows for both minimal requirements and progressive editing
- Machine- and human-readable in the YAML language
- Can be used to auto-generate a module page (Rmarkdown-HTML)

Semantic tagging (archaeological & modelling)

Modular licencing

```yaml
id: 2021-GALAN-001
nassaVersion: 0.1.0
moduleType: Algorithm
title: Random Walk in R
moduleVersion: 0.1.0
contributors:
  - name: Galan, Jose M.
    roles: [ "Author", "Copyright Holder", "Creator" ]
    email: my.stable@email.com
    orcid: 1234-1234-1234-1234
lastUpdateDate: 2021-11-15
description: ...
keywords: ...
implementations:
  - language: R
    codeDir: r_implementation/
softwareDependencies:
  - R >= v4.0
  - tidyverse (R package) >= 1.3.0
  - ggrepel (R package) >= 0.9.1
inputs:
  - name: initialX, initialY
    type: float (numeric)
    unit: no specific unit
    description: ...
outputs:
  - name: trajectory
    type: data frame (tibble)
    description: ...
readmeFile: README.md
docsDir: documentation/
license: MIT
```

3.3.
On documentation

The README.md file:

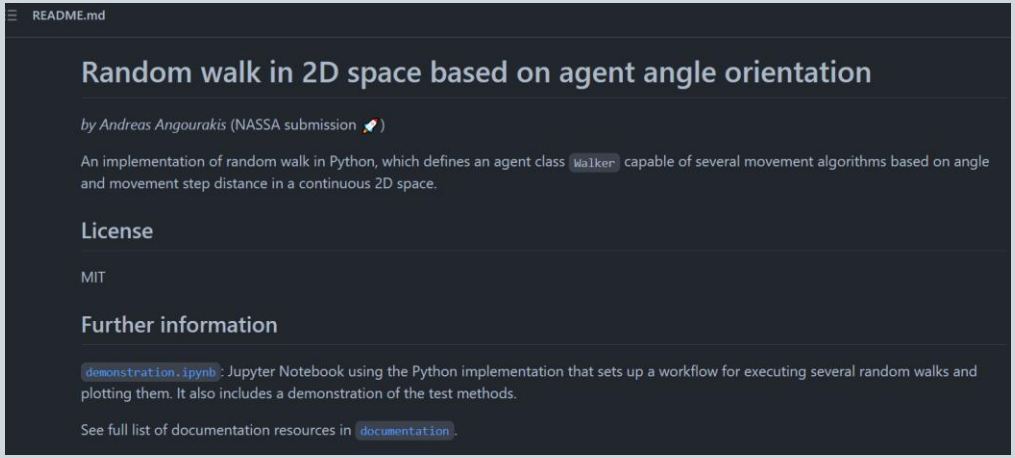
- Human readable free text presenting the module
- Includes title, author, version, etc.
- Hyperlinks to documentation and demonstration materials
- Can be used in the generation of a module page (HTML)

Variable lists, diagrams, pseudocode

Screenshots

Demonstrations, tutorials

Additional documents (ODD?)


README.md

Random walk in 2D space based on agent angle orientation

by Andreas Angourakis (NASSA submission 🚀)

An implementation of random walk in Python, which defines an agent class `Walker` capable of several movement algorithms based on angle and movement step distance in a continuous 2D space.

License

MIT

Further information

`demonstration.ipynb` : Jupyter Notebook using the Python implementation that sets up a workflow for executing several random walks and plotting them. It also includes a demonstration of the test methods.

See full list of documentation resources in `documentation`.


Module name or title

General documentation

Table of contents

- `designDetails.md` : the description of the methods and the class (`Walker`) for random walking.
- `variablesAndParameters.xcf` and `variablesAndParameters.png` : source file (GIMP image editor) and PNG image of a infographic explaining the module variables and parameters.
- Python implementation : documentation specific to the Python implementation.

3.3.
On documentation

The README.md file:

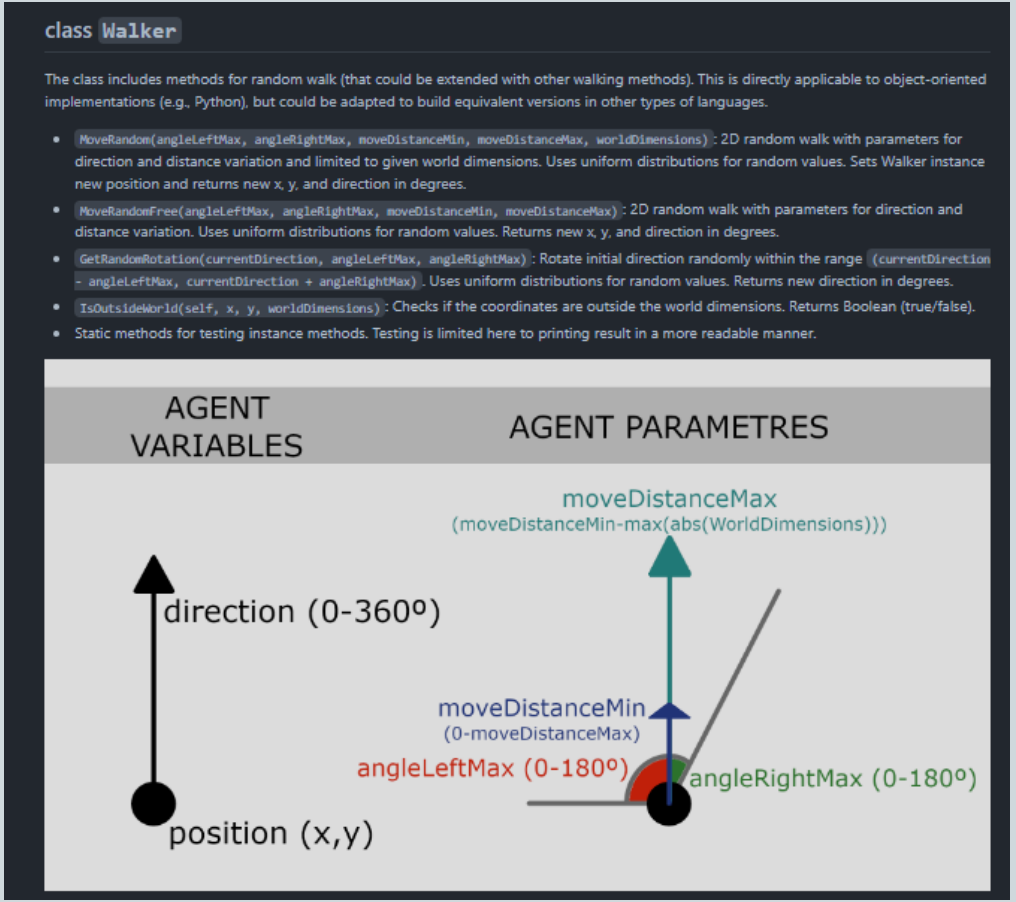
- Human readable free text presenting the module
- Includes title, author, version, etc.
- Hyperlinks to documentation and demonstration materials
- Can be used in the generation of a module page (HTML)

Variable lists, diagrams, pseudocode

Screenshots

Demonstrations, tutorials

Additional documents (ODD?)



3.3.
On documentation

The README.md file:

- Human readable free text presenting the module
- Includes title, author, version, etc.
- Hyperlinks to documentation and demonstration materials
- Can be used in the generation of a module page (HTML)

Variable lists, diagrams, pseudocode

Screenshots

Demonstrations, tutorials

Additional documents (ODD?)

3.4.
On version control and submission management

Nassa module submitted as (part of) GitHub repository:

https://github.com/Archaeology-ABM/NASSA-modules

```
YYYY-SURNAME-001 (module root)
|   CHANGELOG.md
|   LICENSE
|   NASSA.yml
|   README.md
|   references.bib
|
└──documentation
|   |   tableOfContents.md
|   |
|   └──<IMPLEMENTATION LANGUAGE>
|       |   tableOfContents.md
|       |   ... (minimum depending on language)
|
└──<IMPLEMENTATION LANGUAGE
    |   moduleShortTitle.<LANGUAGE EXTENSION>
```

A clearly defined, versioned definition, of the NASSA module structure(currently v0.1.0)

https://github.com/Archaeology-ABM/NASSA-schema

```
$ nassa validate -d .
nassa v0.6.0 for the NASSA standard v0.1.0

Searching NASSA.yml files...
6 found
Checking NASSA versions...
Loading NASSA.yml files...
***
6 loaded
Validation passed: OK
```

https://github.com/Archaeology-ABM/nassa-hs

Software to automatically validate structure and metadata of submitted modules (Haskell)

Really not possible without Clemens Schmid!

Open questions

More examples/models to follow?

More/alternative best practice principles?

Practical suggestions towards a better NASSA library?

NA S²A

NETWORK FOR AGENT-BASED MODELLING OF SOCIO-ECOLOGICAL SYSTEMS IN ARCHAEOLOGY

EAA BUDAPEST 2022 31 AUG - 3 SEP

[RE]INTEGRATION

Thank you

and...

site: https://archaeology-abm.github.io/NASA

repository: https://github.com/Archaeology-ABM