SARRA-Py: A Python-based geospatial simulation framework for agroclimatic modeling

Jeremy Lavarenne, Asse Mbengue

SoftwareX 30 (2025) 102145

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original Software Publication

SARRA-Py: A Python-based geospatial simulation framework for agroclimatic modeling

Jérémy Lavarenne [a,b,*], Asse Mbengue [c,d]

[a] CIRAD, UMR TETIS, 34000 Montpellier, France
[b] TETIS, Univ Montpellier, AgroParisTech, CIRAD, CNRS, INRAE, Montpellier, France
[c] Meteorological Exploitation Directorate, ANACIM, Dakar, Senegal
[d] University of Montpellier, ESPACE-DEV, IRD, France

ARTICLE INFO

Keywords:
Agroclimatology
Crop modelling
Rainfed cereal crops
Data-scarce environments
Water-balance

ABSTRACT

SARRA-Py is an open-source, Python-based adaptation of the long-standing SARRA crop model family–specifically building upon SARRA-H to enable spatially explicit agroclimatic simulations in tropical and data-limited environments. By leveraging Python's geospatial libraries (e.g., Xarray), SARRA-Py extends SARRA-H's proven crop physiology routines to large-scale, raster-based analyses, streamlines ingestion of diverse climate inputs with minimal preprocessing, and eases model customization via a modular code structure. Users interact with SARRA-Py primarily through Jupyter notebooks that provide guided workflows for data preparation, parameter configuration, and visualization of results. This design closes the gap between point-based crop models and broader geospatial frameworks, offering a practical tool for agricultural risk management, climate adaptation studies, and yield forecasting. Consequently, SARRA-Py fosters reproducible, scenario-based analyses and informs decision-making in vulnerable regions where water deficits, sparse ground observations, and climate variability threatens food security.

Metadata

Nr	Code metadata description	Metadata
C1	Current code version	0.1.3
C2	Permanent link to code/repository used for this code version	https://github.com/SARRA-cropmodels/SARRA-Py / https://doi.org/10.5281/zenodo.14641803
C3	Permanent link to reproducible capsule	none
C4	Legal code license	GPL-3.0
C5	Code versioning system used	git
C6	Software code languages, tools and services used	Python 3
C7	Compilation requirements, operating environments and dependencies	numpy, pandas, xarray, astral, matplotlib, tqdm, rasterio, rioxarray
C8	If available, link to developer documentation/manual	https://sarra-h.teledetection.fr / https://sarra-cropmodels.github.io/SARRA-Py/
C9	Support email for questions	sarra-h@cirad.fr

1. Motivation and significance

Assessing the impact of climate variability and climate change on agricultural production is a critical challenge, particularly in tropical and subtropical regions where rainfed systems dominate. The increasing rainfall variability and frequency of extreme weather events associated with climate change may cause reduced yields and heightened risks of food shortages [1]. Hence, crop simulation models are useful tools for assessing long term impacts, but also for operational planning and responding to potential food crises. Understanding and mitigating agroclimatic risks taking into consideration spatial variability observed at scale of territories requires tools that can integrate geospatial data to assess crop performance under varying environmental conditions. Traditional point-based crop models can inform long-term risk assessments and operational decision-making but often lack the ability to scale up to larger territories or natively handle complex geospatial data. In areas like West Africa and the Sahel—where meteorological observations are sparse—decision-makers require tools that integrate multiple climate datasets to reliably simulate crop growth and water stress dynamics at broader scales. To meet these needs, SARRA-Py (Système

* Correspondence: Jérémy Lavarenne, TETIS, Univ Montpellier, AgroParisTech, CIRAD, CNRS, INRAE, Montpellier, France
E-mail addresses: jeremy.lavarenne@cirad.fr (J. Lavarenne), asse.mbengue@anacim.sn (A. Mbengue).

https://doi.org/10.1016/j.softx.2025.102145
Received 6 February 2025; Received in revised form 21 March 2025; Accepted 24 March 2025
Available online 29 March 2025

d'Analyse Régional des Risques Agroclimatiques–Python) was developed as an open-source evolution of the long-standing SARRA model family (which history is reviewed in [2]), specifically developed and validated for cereal crops in tropical environments where climate constraints, particularly water deficits, significantly impact productivity [3,4].

Unlike many existing spatialized models that demand extensive data preprocessing (e.g., CRAFT [5]) or more generic geospatial frameworks (e.g., GeoPandas, Xarray) that are far from being natively tailored to perform crop simulations, SARRA-Py bridges both worlds. It accepts a wide range of raster data formats—including GeoTIFF for "traditional" geospatial layers and NetCDF for reanalysis or forecast datasets—enabling users to run simulations using satellite-derived rainfall (e.g., CHIRPS [6], TAMSAT [7]), climate model outputs (e.g. AgERA5 reanalysis products [8], CORDEX projections [9]), and work alongside vegetation dynamics data (e.g. MODIS NDVI) or in situ observations with minimal transformation. Building on well-established physiological formalisms (e.g., thermal time, photoperiod sensitivity, and basic biomass accumulation rules, described in the latest SARRA-H foundational paper from Sultan et al., 2005 [10]), SARRA-Py simulates crop phenology, water balance, and carbon-related processes. Although model precision ultimately depends on context-specific validation, these core calculations offer a relatively robust representation of rainfed cereal dynamics. Compared to widely used models such as DSSAT [11] or APSIM [12], SARRA-Py requires fewer parameters—making it particularly suitable for data-limited environments—while still capturing key water- and temperature-driven stresses (Table 1).

The software, as a Python package, is primarily script-based and does not feature a graphical user interface; however, it includes ready-to-run Jupyter notebooks to illustrate standard workflows. These notebooks help researchers and practitioners navigate data preparation, parameter assignment, and simulation execution more smoothly.

By handling spatially explicit data and focusing on simpler calibration requirements, SARRA-Py can be adapted to explore crop performance under various climate scenarios, assess risk management strategies, and inform policy or development projects in vulnerable regions. While still relatively new, its modular, extensible design positions SARRA-Py as a valuable addition to the toolbox for agroclimatic studies and potential decision support, particularly in the Sahelian and Sudanian zones where rainfed farming is dominant.

Although SARRA-Py's development has primarily focused on tropical and Sahelian environments, members of the SARRA model family have also been applied in other contexts—including subtropical regions of Brazil—demonstrating adaptability under diverse climatic conditions. These efforts, detailed in Lavarenne et al. (in press) [2], show that while the model's simplified water-balance approach is especially valuable in data-scarce settings, partial re-calibration or additional parameter adjustments (e.g., for photoperiod sensitivity or thermal time requirements) may be necessary outside the Sahel. Consequently, users interested in deploying SARRA-Py beyond dry tropical zones should validate key phenological and agronomic parameters against local datasets to ensure accurate performance. For a more complete overview of the model family's evolution and applications, readers are referred to Lavarenne et al. (in press) [2].

2. Software description

SARRA-Py is an open-source Python package for spatially explicit modeling of rainfed cereal crops in tropical and subtropical regions. It integrates daily climate data (e.g., temperature, global radiation, rainfall) alongside invariant inputs such as soil water-holding capacity, and crop-specific parameters, to simulate daily crop phenology, water balance, and biomass production. Although SARRA-Py has no graphical user interface, a set of Jupyter notebooks provides guided workflows and lowers the entry barrier for new users.

2.1. Software architecture

The software is built around a set of example Jupyter notebooks and a small number of Python modules. Users typically start in a notebook by loading daily rainfall, and climate (temperature, reference evapotranspiration, global radiation) datasets (in GeoTIFF or NetCDF format). Helper functions from the `data_preparation` module then combine these inputs into an initial `xarray.Dataset` object, with spatial resolution and extent inherited from the rainfall data. A user-defined start date and duration define the time dimension, while YAML files supply crop and management parameters, making it straightforward to reconfigure simulations without editing source code.

At the core, a main function (`run_model`, located in the `models` module) orchestrates the simulation by iterating over each day (i.e., each slice of the `xarray.Dataset`) within the chosen time range. It sequentially updates phenological stages, soil water content, and biomass allocation by calling specialized routines in `bilan_pheno`, `bilan_hydro`, and `bilan_carbo` modules. Inputs, intermediate states, and final outputs are stored in the same xarray.Dataset, so any computed variable can be inspected at any point in time. The example notebooks illustrate this workflow: after data loading and parameter assignment, run_model is invoked to loop through the dataset, recording each daily update. Fig. 1 below summarizes the data flow of the software.

Because SARRA-Py can handle large spatial domains at a daily time step, the memory footprint may be substantial. For reference, a one-year simulation over Senegal at 0.0375° (~4 km) resolution can consume 8–10 GB of RAM. In practice, users can run chunked or parallelized simulations (e.g., with Python's joblib or Dask) to distribute the workload across available hardware, reducing peak memory usage. Those with limited resources are also advised to increase swap space on Unix-based systems, as detailed in the notebooks, to ensure that longer and/or

Table 1
Key comparisons among SARRA-Py, DSSAT, and APSIM.

	SARRA-Py	DSSAT	APSIM
Focus	Rainfed cereals (millet, sorghum, maize); data-limited, tropical or semi-arid contexts	Many crops (cereals, legumes, vegetables); widely used worldwide	Multiple crops, rotations, and system-level analyses; diverse agricultural contexts
Parameter Complexity	Fewer parameters needed; simplified calibration	Moderate to high; detailed phenology and soil modules	Moderate to high; includes soil nutrient and multi-year rotations
Spatial Handling	Natively supports raster data (Python + xarray), enabling large-scale geospatial simulations	Primarily point-based; spatial analysis requires external tools	Generally point-based; spatial runs typically scripted in R/Python
Data Requirements	Minimal daily weather + soil info; fits contexts with few observations	Daily weather, soil profiles, management data; more extensive parameter sets	Similar to DSSAT; can incorporate additional details like soil N or multi-crop sequences
Key Strengths	Straightforward calibration, open-source, adapted for tropical cereals	Well-validated, large user base, extensive documentation	Comprehensive system approach (soil-water-nutrients), strong for multi-crop studies
Limitations	Currently focused on cereal crops; requires Python skills; high-res runs can be memory-intensive	Complex to calibrate; lacks native spatial module	Higher data needs; can be time-consuming to configure
Typical Uses	Yield forecasting, agricultural risk assessment in dry tropics/semi-arid zones; scenario testing	Crop breeding, management optimization, and climate adaptation studies	Long-term rotation modeling, detailed soil-water-nutrient interaction analysis

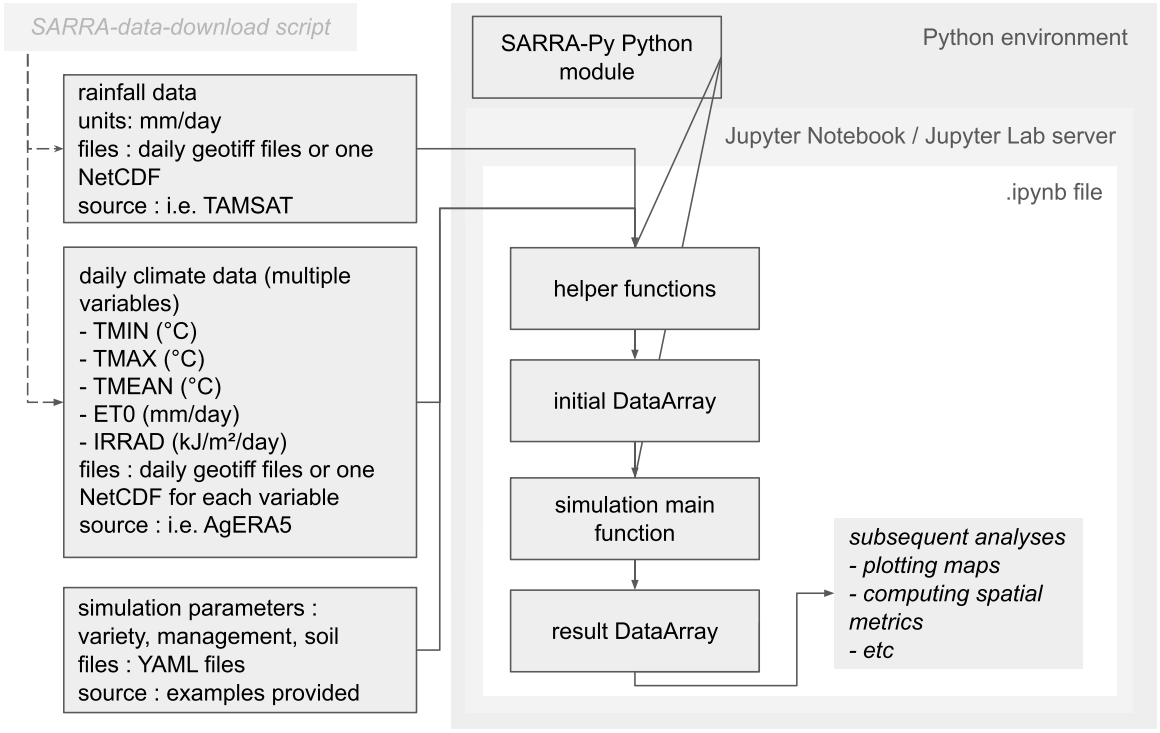

Fig. 1. High-level data flow for SARRA-Py simulation.

high-resolution simulations can be completed without memory issues.

2.2. Software functionalities

SARRA-Py's primary objective is to simulate how water availability and climate conditions affect rainfed cereal production. Each day, the software computes soil water dynamics, crop growth stages, and biomass accumulation, using a tipping-bucket approach for water balance and a simplified allometric scheme for biomass allocation. Crop emergence is triggered by a soil moisture threshold, and subsequent phenological stages (e.g., flowering, grain filling) are driven by cumulative thermal time, optionally adjusted for photoperiod sensitivity.

The software can ingest various climate data (rainfall, temperature, potential evapotranspiration, solar radiation) from satellite products or model outputs, enabling users to run simulations under historical conditions or future climate scenarios. Soil properties (e.g., water holding capacity, runoff thresholds) are equally flexible, allowing for different profiles across the simulation grid. At the end of a run, outputs such as final yield, LAI, and drought stress indicators remain in the `xarray.Dataset`, readily visualized via time-series plots or spatial maps within the same notebook.

This modular design helps researchers adapt SARRA-Py to a broad set of questions. For instance, one can alter the simulation start date and sowing rules to study the effect of shifting sowing windows, or adjust crop parameters to simulate different varieties. The open-source architecture also allows advanced users to integrate additional processes—like nutrient limitations or pest impacts—by modifying the respective Python modules. Through these features, SARRA-Py aims to support agroclimatic research and scenario analyses in data-scarce environments, particularly across the Sahelian and Sudanian zones where water stress frequently constrains production.

2.3. Testing and validation note

During the translation from SARRA-H codebase to SARRA-Py, we compared outputs under identical input and parameter settings, ensuring that daily water-balance, phenology, and yield results matched those of SARRA-H. Since SARRA-Py retains the same biophysical formalisms, previously published evaluations of SARRA-H against field data remain directly relevant. Consequently, further details on empirical validation and calibration can be found in those studies.

3. Illustrative examples

SARRA-Py comes with several example Jupyter notebooks[1] that demonstrate common use cases, such as simulating millet yields in Niger, exploring optimal sowing dates in Senegal, or testing different climate inputs and soil parameters. This section describes one such example—simulating millet yields in Niger for a single year—highlighting the typical workflow and core features.

Running a millet yield simulation in Niger (2017): Niger spans a large portion of the Sahel, presenting both data challenges (limited in situ observations) and significant climate variability. The example notebook focuses on a single growing season in 2017, illustrating how to prepare input data, set up crop parameters, run the daily simulation, and visualize outputs. Because simulations at the country scale can demand considerable memory (on the order of ~8–16 GB), the notebook also guides users on how to increase virtual memory (swap space) in Unix-based systems if needed.

3.1. Defining the simulation scope

Users first specify a start date and a simulation duration (in days). It is generally recommended to begin the simulation about a month prior to expected sowing so that the soil water balance can stabilize:

```
import datetime
# Example start date and total duration in days
```

(continued on next page)

[1] see https://github.com/SARRA-cropmodels/SARRA-Py/tree/main/notebooks

(*continued*)

```
date_start = datetime.date(2017, 5, 1)
duration = 220 # ~7 months, covering pre-sowing plus the growing
  season
```

3.2. Loading and preparing data

Next, the notebook loads spatial layers for daily climate variables (rainfall, temperature, solar radiation, reference evapotranspiration) and soil properties (e.g., water-holding capacity). For this example, as well as the five other example notebooks, all data are provided as prepared and automatically downloaded from a data repository. If not, these data can be semi-automatically downloaded and prepared in the right format by the SARRA-data-download helper tool[2] [13]. In our example, rainfall data come from TAMSAT v3.1 and climate variables from AgERA5; both are provided as daily GeoTIFF files:

```
import xarray as xr
from sarra_py import (load_TAMSAT_data, load_AgERA5_data,
  load_iSDA_soil_data, calc_day_length_raster_fast)
# Create an empty dataset
base_data = xr.Dataset()
# Load daily rainfall and weather data into base_data
base_data = load_TAMSAT_data(base_data, rainfall_data_path,
  date_start, duration)
base_data = load_AgERA5_data(base_data, climate_data_path,
  date_start, duration)
# Incorporate soil parameters and compute daily day-length values
base_data = load_iSDA_soil_data(base_data, grid_width,
  grid_height)
base_data = calc_day_length_raster_fast(base_data, date_start,
  duration)
```

These functions automatically clip or reproject the rasters so they share the same spatial grid, defined by the resolution of the rainfall dataset. The user can inspect the resulting `xarray.Dataset` to confirm that dimensions *(time, y, x)* and variables (e.g., rain, tpMoy, ET0, rg) are correct.

3.3. Initializing crop and management parameters

To configure the simulation, SARRA-Py relies on parameter files in YAML format, stored in the `./data/params/` folder, that specify crop variety traits, soil characteristics, and management practices (e.g., sowing rules). Loading these is straightforward:

```
file_paramVariete = "millet_variety.yaml"
file_paramITK = "millet_niger_2017.yaml"
file_paramTypeSol = "USA_iowa_V42.yaml"
paramVariete, paramITK, paramTypeSol = load_YAML_parameters(
file_paramVariete, file_paramITK, file_paramTypeSol
)
```

A new data dataset is then created by copying `base_data` and injecting these parameters. Key variables—such as soil moisture initialization or sowing triggers—are set up using helper functions:

```
data = base_data.copy()
data = initialize_simulation(data, grid_width, grid_height,
  duration,
  paramVariete, paramITK, date_start)
data = initialize_default_irrigation(data)
data = calculate_once_daily_thermal_time(data, paramVariete)
```

[2] see https://github.com/SARRA-cropmodels/SARRA-data-download

3.4. Running the model

After preparing the dataset, users call `run_model` from the SARRA-Py package to execute daily updates over the simulation period. This routine iterates through each time step and each grid cell, applying water-balance, phenology, and biomass-allocation processes:

```
data = run_model(paramVariete, paramITK, paramTypeSol, data,
  duration)
```

Progress is displayed via a simple progress bar, indicating how many days have been processed.

3.5. Visualizing results

SARRA-Py stores all intermediate and final outputs (e.g., yield, leaf area index, soil moisture) directly in the same `xarray.Dataset`. Users can generate spatial maps or time-series plots for any variable of interest. Fig. 2 illustrates the estimated yield distribution (in kg/ha) across Niger at the final simulation date (Day 365 of 2017).

Similarly, time-series plots of soil water content or phenological stage can be generated for specific locations by slicing [y, x] coordinates:

```
# plot stockSurf variable, which is the soil water content
data["stockSurf"].sel(x = 2.1, y = 13.5).plot.line()
plt.show()
# alternatively, we can plot other variables just by changing the
  name, for instance to plot the time series of the phenological
  phases at the same location:
data["numPhase"].sel(x = 2.1, y = 13.5).plot.line()
plt.show()
```

3.6. Key insights

By analyzing endpoint maps and/or time-series plots for all variables (which listing and definition can be accessed on the functions docstrings of the `bilan_carbo` module), users can identify regions that experienced drought stress or poor soil conditions. This example illustrates how SARRA-Py supports scenario testing in data-scarce contexts, offering a workflow that can be readily adapted to different crops (e.g., maize, sorghum) or management strategies (e.g., altered sowing windows, irrigation).

4. Impact

SARRA-Py extends the scope of agroclimatic research by enabling spatially explicit simulations across large territories and data-limited environments. Its integration of geospatial datasets and daily crop processes supports investigations into how soil variability influences drought resilience and how photoperiod sensitivity shapes yield outcomes in intertropical regions. Researchers can now consider additional modules, such as nitrogen balance or pest and disease impacts, without needing to rebuild the software's core structure. This flexibility allows for advanced questions about genotype-by-environment interactions, climate adaptation strategies, and the performance of crops under increasingly frequent extreme weather events, while still maintaining a relatively simple calibration process suitable for data-scarce contexts.

In practical terms, SARRA-Py improves everyday research workflows by reducing the time spent on data preprocessing (thanks to the SARRA-data-download companion tool), simulation runs, and result visualization, thanks to a unified Python and Jupyter-based approach. Its modular design and open-source distribution encourage reproducible science, as shown in multi-institutional collaborations documented in Lavarenne et al. [2]. This family of models has already reached diverse user groups—academic researchers, government agencies, and NGOs in Africa, Brazil, and South Asia—evidenced by over 2000 downloads since

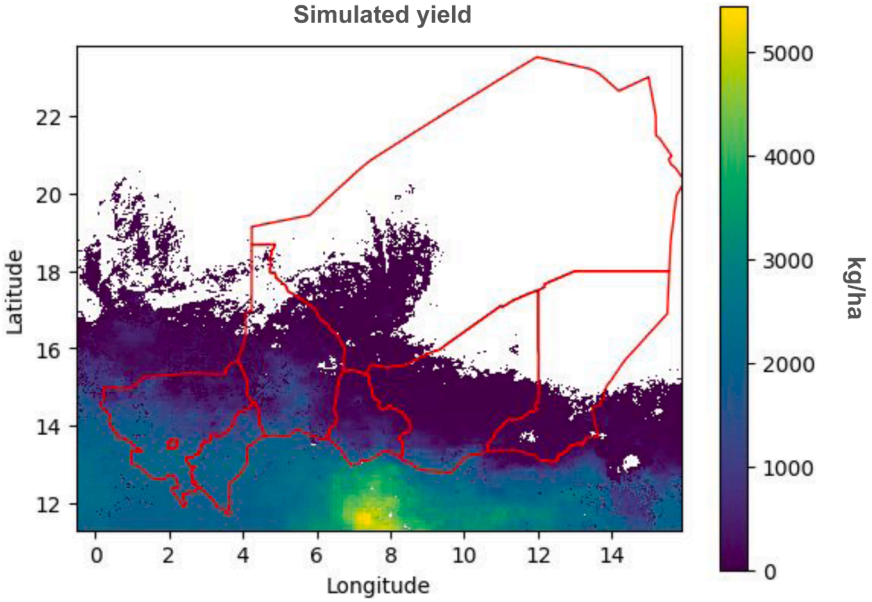

Fig. 2. example of simulated yield output at the last simulated day in Niger for year 2017; red lines indicate level 0 and 1 administrative boundaries.

the release of SARRA-H, the direct predecessor of SARRA-Py[3], as well as a list of publications using results produced by these models.[4]

By offering region-wide simulations and scenario testing, SARRA-Py provides more spatially refined insights that can improve drought mapping, early warning systems, and resource allocation decisions. Beyond research, SARRA-Py holds potential for commercial applications in agri-tech and consulting, where precision agriculture services or climate risk assessments could leverage its crop modeling capabilities at scale. In the future, it is poised to contribute to broader climate adaptation agendas by enhancing yield forecasts under IPCC-class climate scenarios, informing breeding programs on traits needed in water-limited conditions, and linking crop water demand with hydrological models. By addressing pressing data and modeling gaps in vulnerable regions, SARRA-Py is likely to remain a valuable tool for decision support and scientific advancement in tropical and subtropical agroecosystems.

5. Conclusions

SARRA-Py offers an open-source, Python-based framework for spatially explicit agroclimatic simulations in tropical and subtropical regions. By combining crop phenology, water balance, and simplified biomass allocations with geospatial data handling, it addresses critical gaps in existing models—namely, the need for fewer parameter requirements, adaptability to data-scarce conditions, and integration with high-resolution climate and soil datasets. Its modular design supports straightforward customization for diverse environments, while example Jupyter notebooks facilitate rapid adoption and reproducibility.

Going beyond yield forecasting, SARRA-Py's capacity to incorporate photoperiod sensitivity, drought stress, and flexible input datasets encourages new lines of inquiry in climate adaptation research. It also lends itself to policy and operational decision-making, particularly in regions where food security is highly dependent on rainfed farming systems. Future development plans include extending the model to additional crops, refining parameterization strategies for local contexts, and incorporating further modules (e.g., nitrogen dynamics, pest impacts). As SARRA-Py continues to evolve, it has the potential to drive innovation in climate-resilient agriculture and serve as a practical tool for researchers, policymakers, and practitioners seeking to understand and mitigate agroclimatic risks.

[3] see https://sarra-h.teledetection.fr

[4] see https://www.zotero.org/groups/5369680/prj_sarra_cropmodels/library

CRediT authorship contribution statement

Jérémy Lavarenne: Writing – original draft, Visualization, Validation, Supervision, Software, Project administration, Methodology, Data curation, Conceptualization. **Asse Mbengue:** Writing – original draft, Visualization, Validation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to thank all the colleagues at the Maison de la Télédétection lab for beta-testing of the software.

References

[1] Kotir JH. Environ. Dev. Sustain. 2011;13:587–605. https://doi.org/10.1007/s10668-010-9278-0.
[2] J. Lavarenne, M. Vaksmann, F. Affholder, M. Ferré, M. Dingkuhn, F.A. Macena Da Silva, et al., SARRA, histoire d'un modèle de simulation des cultures pour les zones intertropicales, Cah. Agric. (in press).
[3] Sultan B, Roudier P, Quirion P, Alhassane A, Muller B, Dingkuhn M, Ciais P, Guimberteau M, Traore S, Baron C. Environ. Res. Lett. 2013;8. https://doi.org/10.1088/1748-9326/8/1/014040.
[4] Traore SB, Ali A, Tinni SH, Samake M, Garba I, Maigari I, Alhassane A, Samba A, Diao MB, Atta S, Dieye PO, Nacro HB, Bouafou KGM. Weather Clim. Extrem. 2014;3:22–30. https://doi.org/10.1016/j.wace.2014.03.008.
[5] Shelia V, Sharda V, Hansen J, Porter C, Zhang M, Aggarwal P, Hoogenboom G. Am. Soc. Agric. Biol. Eng. Annu. Int. Meet. 2015;2:1094–116. https://doi.org/10.13031/aim.20152182505. 2015.
[6] Funk C, Peterson P, Landsfeld M, Pedreros D, Verdin J, Shukla S, Husak G, Rowland J, Harrison L, Hoell A. J. Michaelsen, Sci. Data 2015;2:150066. https://doi.org/10.1038/sdata.2015.66.
[7] Maidment RI, Grimes D, Black E, Tarnavsky E, Young M, Greatrex H, Allan RP, Stein T, Nkonde E, Senkunda S, Alcántara EMU. Sci. Data 2017;4:170063. https://doi.org/10.1038/sdata.2017.63.
[8] Copernicus Climate Change Service, (2019), doi:10.24381/cds.6c68c9bb.

[9] Coppola E, Raffaele F, Giorgi F, Giuliani G, Xuejie G, Ciarlo JM, et al. Clim. Dyn. 2021;57:1293–383. https://doi.org/10.1007/s00382-021-05640-z.
[10] Sultan B, Baron C, Dingkuhn M, Sarr B, Janicot S. Agric. For. Meteorol. 2005;128:93–110.
[11] Hoogenboom G, Porter CH, Boote KJ, Shelia V, Wilkens PW, Singh U, et al. Burleigh dodds ser. agric. sci. Burleigh Dodds Science Publishing; 2019. p. 173–216. https://doi.org/10.19103/AS.2019.0061.10.
[12] Holzworth DP, Huth NI, deVoil PG, Zurcher EJ, Herrmann NI, McLean G, et al. Environ. Model. Softw. 2014;62:327–50. https://doi.org/10.1016/j.envsoft.2014.07.009.
[13] J. Lavarenne, S. Madec, (2024), doi:10.5281/zenodo.11091599.