167. Farmingpy: Python package for developing digital twins and precision farming data processing

A. Hartikainen*, J. Backman and M. Pastell
[1] *Natural Resources Institute Finland (Luke), Latokartanonkaari 9, Helsinki, Finland;*
* annimari.hartikainen@luke.fi

Abstract

Digital twins of cropping systems are under development aiming towards better forecasting and decision-making on farms. A Python package called farmingpy was developed to support the development of digital twins combining machinery data and remote sensing data with the APSIM simulation model. The package can also be used for other precision farming data processing tasks. Shared open-source software packages can provide generic modules and assist the digital twin development processes by reducing duplication of work. The aim of this paper is to present the farmingpy Python package including two use case examples.

Keywords: APSIM, data processing, digital twin, ISOBUS, python package

Introduction

Digital twins of cropping systems are under development aiming towards better forecasting and decision-making at farms. The term digital twin describes a trend where farm objects are virtualized and increasingly controlled remotely based on object-specific analyses (Verdouw *et al*., 2021). A few applications of digital twins already exist in agriculture but further development would be required to reach the level and benefits of digital twins in other disciplines (Pylianidis *et al*., 2021).
Digital twins consist at least of monitoring, interface and analytics components (Pylianidis *et al*., 2021). Shared open-source software packages could provide generic modules and assist the digital twin development processes by reducing duplication of work.
Development of digital twin software which combines machinery data with a simulation model (Bloch *et al*., 2023) initiated the development of farmingpy open-source Python package (https://github.com/TwinYields/farmingpy). The package contains functionality which can also be used for other precision farming data processing tasks. Python is a popular language in scientific computing (Virtanen *et al*., 2020) and several Python packages for modeling agricultural systems exist such as PCSE (Berghuijs *et al*., 2024) and AquaCrop-OSPy (Foster *et al*., 2017) for crop modelling and pyfao56 for irrigation decision support (Thorp, 2022). Farmingpy aims to complement existing Python packages and to be interoperable with Python geospatial tools.
The aim of this paper is to present farmingpy Python package with a use case on combining machinery data with simulation model APSIM (Agricultural Production Systems sIMulator) (Holzworth *et al*., 2014).

Materials and methods

A Python package called farmingpy was developed to support research in digital twins for decision making in precision farming. The aim of the development was to cover the whole cropping process including processing of input data including variable rate application of input from ISOBUS task data, real-time simulation of crop status using a simulation model, monitoring of the crop with remote sensing and analysis of the outcomes from yield maps. Figure 1 represents the architecture of the digital twin for precision farming with modules of the farmingpy package. The functionality

makes the package also useful for the analysis of precision farming experiments. Python geospatial data library formats are used when possible i.e. GeoPandas (Van den Bossche *et al*., 2024) for vector data and xarray (Hoyer and Hamman, 2017) for raster data. This makes it easy to process the data further using standard Python geospatial libraries.

The farmingpy package contains the following modules, folder names in parentheses:
- interface to APSIM, including ensemble modelling interface (apsim),
- ISOXML task data parser for planned and implemented tasks (data),
- earth observation (EO) download and analysis functions (eo), and
- pedotransfer functions (soil).

APSIM
Simulation models are often running inside digital twins. For the fields, crop growth models such as APSIM are essential. The farmingpy module apsim provides an interface for setting up and running high resolution spatial simulations with APSIM next generation (Holzworth *et al*., 2014). The decision to use APSIM as the simulation model was made due to its ability to model nitrogen limitation to crop yield and the modularity of APSIM code. APSIM is implemented in C# and python.net package is used as a bridge to call APSIM function directly from Python.
This integration enables running APSIM models from Python with daily step and assimilating EO and sensor data with APSIM by model ensembles and optimizing model parameters. At the time of the writing a special build of APSIM including C# extension classes is required for the ensemble modelling functionality, and the possibility of contributing these changes back to APSIM is being investigated. The model can be used to forecast the effect of management decisions during growing season to make fertilization and irrigation decisions.

ISOXML
Crop management dates, yield data and inputs such as fertilization are commonly recorded as ISOBUS data. There are limited open-source options for reading the data. In farmingpy package, the module data reads ISOBUS task data generated by farming machinery such as tractors and combine harvesters. It provides Python interface to the underlying ISOBUS task data parsing package written in C#.

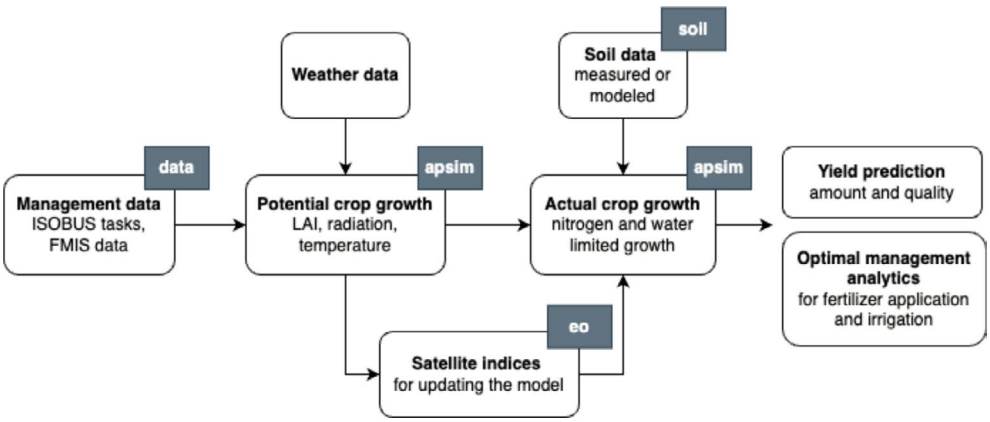

Figure 1. Architecture of a digital twin for precision farming (white rectangles and black arrows) and the parts where farmingpy package modules (grey rectangles) have been used.

Parsing package assumes the data complies with ISO 11783-10 standard and extracts the information relevant to the digital twin. This module makes parsing of ISOBUS task data easier, as manual parsing or parser code writing is not required. The module has been tested using sowing and fertilization tasks and silage and cereal harvester yield maps from several farms, with several combines from different manufacturers.

Earth observation
EO data can be retrieved from SentinelHub and crop biophysical parameters (Leaf area index, Canopy chlorophyl content and Canopy water content) can be predicted using neural network models from Sentinel2 and PlanetScope images. The neural network models are a Python implementation of the Sentinel 2 Toolbox Biophysical Processor (Weiss *et al.*, 2020) models and use the same weights. These interfaces are found in the module eo.

Soil
Soil water holding capacity is a required input for digital twin models which want to estimate the crop water demand. This data is seldomly available from commercial farms. However, it can be estimated from soil sample data, more specifically from the sand-silt-clay distribution and organic matter content, using USDA Rosetta or euptfv2 (Szabó *et al.*, 2021) pedotransfer functions. Module *soil* provides a simple unified interface for estimating the soil water capacity with soil sample data available from farms in a format which can be used for crop models. The EUPTF2 R-package is called from Python using rpy2 and the package needs to be installed in order to use the interface.

Use cases

Two farmingpy package use cases with apsim and data modules are presented. Note that the use cases cover only parts of the digital twin components as shown in Figure 1.

Decision support for nitrogen application using APSIM
This use case demonstrates a hypothetical example of using the simulation of crop growth with an ensemble of APSIM models. The ensemble is initialized to account for uncertainty in initial soil nitrogen concentration and information about soil water holding capacity. The model is run with daily time step and example on the use of LAI from remote sensing to choose the most likely model from the ensemble is given. The full use case code is available as a Jupyter notebook file on Zenodo (https://doi.org/10.5281/zenodo.14245025). The initial values for the ensemble and remote sensing data are from real field data from spring wheat (cultivar Jaarli) from a field parcel in Jokioinen, Finland described in Bloch *et al.* (2023). However, the decision support aspect of demonstrated in the example was not implemented. The use of calibrated APSIM model for similar decision support use case has been shown by Jin *et al.* (2017).
The farmingpy module apsim was used as an interface for running APSIM and implementing data-assimilation. The notebook demonstrates the use of farmingpy APSIM interface with the following steps:
1. Setting up the base model
2. Creating model ensembles
3. Implementing data-assimilation
4. Providing decision support

Setting up the base model includes setting the input values and updating internal model variables. The base model is then used to initialize model ensembles. Ensemble describes uncertainty of one or more model parameters. Uncertainty distribution of parameters was set and each model in the ensemble is initialized with random values drawn from these distributions. In this example, there

were three ensembles with different uncertain soil parameters. Ensembles were initialized and uncertain values set through the interface. All ensemble models were run and the results plotted.
Basic data-assimilation was implemented to choose the most likely model according to remotely sensed LAI. First, leaf area index (LAI) was fetched from remote sensing data from Terrascope. An example in spatial simulation and decision support was developed. It includes using four separate model ensembles with varying management options to simulate yield and grain protein outcomes of different management choices.
Figure 2 shows model ensemble outputs (lines) and remotely sensed LAI values (dots). Simulated crop was spring wheat with sowing and fertilizer application done on 23 May 2022.

ISOXML map use case
The data module is used to import real measurements produced by agricultural machinery from operations that have been made during the growing season. In precision farming, ISOBUS compatibility has become common. Most of the machines can utilize ISOBUS Task controller that uses and produces ISOBUS Task files. The Task files are in standardized format, but not necessarily easy to use. The data module simplifies the usage of the data that the machines produce.
The underlying C# implementation parses the original XML data, that has many cross references. The output of the parser is table format data, that has geographical location, date and time in each row in addition to the measurement data that is bounded to that particular location and time. The table has also a header which describes the content of the data in each column.
Originally in the Task file, the measurement data is a binary file (for example TLG00001.bin). There is a separate TimeLog XML file (TLG00001.xml) which describes the content of the binary file. The TimeLog XML file contains a list of DataLogValue (DLV) elements which references to DeviceElements (DET) and describes the type of the measurement (ProcessDataDDI). To transform this information into a plain text, the corresponding DeviceElement has to be found from the device description that is under the Device (DVC) element in the original XML Task file (TASKDATA.XML). The device description can also include geometric information about the implement and the tractor. The parser is able to utilize this information and calculate the actual geographical location of the

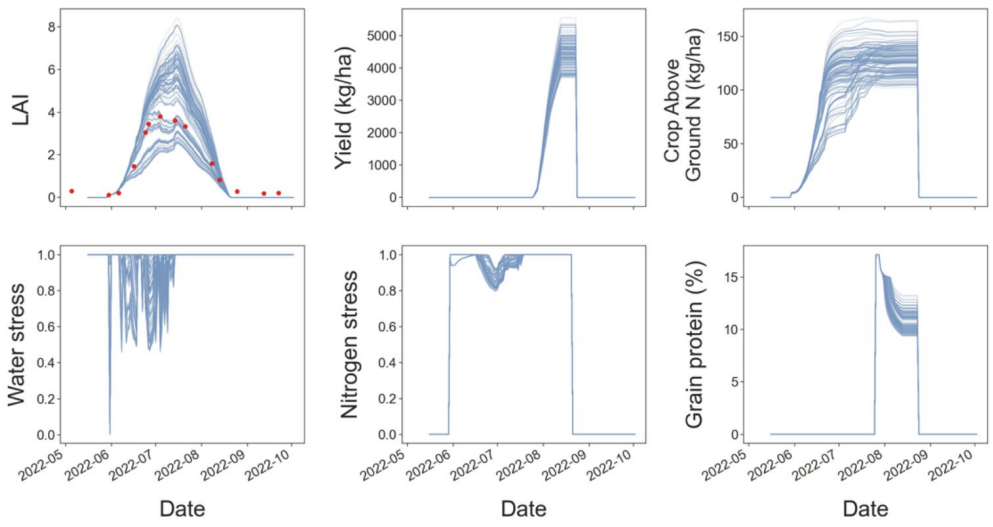

Figure 2. Simulated leaf area index (LAI), yield, crop above ground nitrogen, water stress, nitrogen stress and grain protein values from one APSIM model ensemble are presented with blue lines. Remotely sensed LAI values are plotted with red dots.

measurement based on the kinematic structure and behavior of the tractor-implement system. However, in the data module, geometric information is not used for simplicity and the geographical location reported by the GNNS receiver is used directly.
Figure 3 shows an example of output of the data module from variable rate nitrogen application task with a centrifugal spreader. The original task file was recorded in June in Jokioinen Finland. The planned and recorded data were parsed using farmingpy classes TimeLogData and TaskReader. The package can return the data either as pandas DataFrame or GeoPandas format. The GeoDataFrame can be plotted on the map as shown in Figure 3 with a single command.

Discussion

The farmingpy open-source package modules apsim, data, eo and soil can be used for recurring purposes, which are typical for digital twin development and precision farming data processing tasks. Sharing these modules as an open-source package can accelerate the development process of cropping system digital twins. The main idea is to provide existing modules as building blocks for other use cases, hence reducing the duplication of work. Open-source package allows the user not only to reuse the code, but also to understand and improve it by examining and modifying the code.
The functionality of the package is intended to be broadly useful. ISOBUS tasks are a standard format for recording work tasks and to the authors' knowledge farmingpy is the only Python package which supports reading the format. The package has been tested and works with several sowing and fertilization tasks as well as cereal and grass harvester data.
The main limitation of the ISOBUS functionality is that it uses C# for reading the timelog data from implemented tasks, this means that the user also needs to install dotnet in order to use the package. In the future it could be beneficial to implement the same functionality in Python and remove the dotnet dependency.
The package interfaces both European and USDA pedotransfer functions making it simple to extract parameters for crop modelling in situations where limited information about soil properties is available.

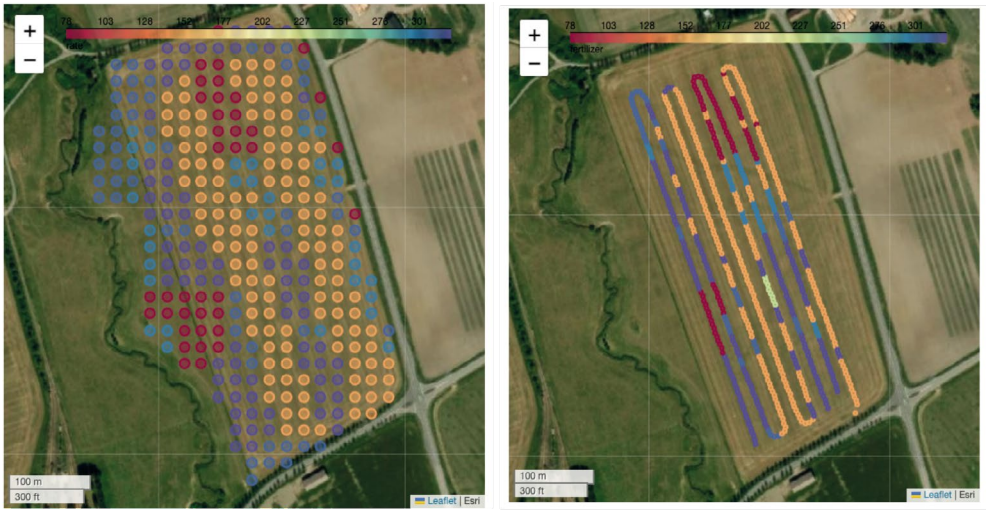

Figure 3. Map of planned (left) and implemented (right) fertilizer application map. Both are read from a task file recorded during application using farmingpy.

Requirements of cropping system digital twins can vary and many different simulation models or machine learning models could be used for decision support instead of APSIM. For instance, it would be easy to integrate the WOFOST model which has recently gained support for simulating nitrogen limited production (Berghuis *et al*. 2024) and is implemented in Python in the PCSE package. There is no coupling between the different modules in the package and other provided modules can be useful on their own for e.g. analyzing data from precision farming experiments. The package is open-source and open for collaborative development and the modules can be improved.

Conclusion

Farmingpy open-source package was created to support digital twins of cropping systems and other precision farming processing tasks. The open-source package is published on GitHub and available under the MIT license. The package reduces duplication of work by providing open-source modules for cropping system digital twins.
For further collaborative development of cropping system digital twins, testing of farmingpy package in other use cases is recommended. Farmingpy package usability could be improved based on requirements raised from new use cases. The Natural Resources Institute Finland (Luke) will continue updating the package to support recurring processing tasks related to digital twin and precision farming data processing.

Acknowledgements

The development of the farmingpy package has been funded by the Natural Resources Institute Finland (Luke) in the TwinYields project, European Commission in the ScaleAgData project (Grant agreement ID: 101086355) and by the Finnish Ministry of Agriculture and Forestry in the INTEGRITY ERA-NET project.

References

Bloch, V., Palosuo, T., Huitu, H., Ronkainen, A., Backman, J., Pussi, K., Suokannas, A., & Pastell, M. (2023). Towards a digital twin for optimal field management, in: In J.V. Stafford (ed.), Precision Agriculture '23, Proceedings of the 14th European Conference on Precision Agriculture. Wageningen Academic Publishers, Wageningen, pp. 377–383. https://doi.org/10.3920/978-90-8686-947-3_46

Berghuijs, H.N.C., Silva, J.V., Reidsma, P., & de Wit, A.J.W. (2024). Expanding the WOFOST crop model to explore options for sustainable nitrogen management: a study for winter wheat in the Netherlands. European Journal of Agronomy, 154, 127099. https://doi.org/10.1016/j.eja.2024.127099

Foster, T., Brozović, N., Butler, A.P., Neale, C.M.U., Raes, D., Steduto, P., Fereres, E., & Hsiao, T.C. (2017). AquaCrop-OS: An open source version of FAO's crop water productivity model. Agricultural Water Management, 181, 18–22. https://doi.org/10.1016/j.agwat.2016.11.015

Hoyer, S., & Hamman, J. (2017). xarray: N-D labeled arrays and datasets in Python. Journal of Open Research Software, 5(1), 10. https://doi.org/10.5334/jors.148

Holzworth, D.P., Huth, N.I., deVoil, P.G., Zurcher, E.J., Herrmann, N.I., McLean, G., …, & Keating, B.A. (2014). APSIM – evolution towards a new generation of agricultural systems simulation. Environmental Modelling & Software 62, 327–350. https://doi.org/10.1016/j.envsoft.2014.07.009

Jin, Z., Prasad, R., Shriver, J., & Zhuang, Q. (2017). Crop model- and satellite imagery-based recommendation tool for variable rate N fertilizer application for the US Corn system. Precision Agriculture, 18, 779–800. https://doi.org/10.1007/s11119-016-9488-z

Szabó, B., Weynants, M., & Weber, T.K.D. (2021). Updated European hydraulic pedotransfer functions with communicated uncertainties in the predicted variables (euptfv2). Geoscientific Model Development, 14, 151–175. https://doi.org/10.5194/gmd-14-151-2021

Thorp, K.R. (2022). pyfao56: FAO-56 evapotranspiration in Python. SoftwareX, 19, 101208. https://doi.org/10.1016/j.softx.2022.101208

Pylianidis, C., Snow, V., Overweg, H., Osinga, S., Kean, J., & Athanasiadis, I.N. (2021). Simulation-assisted machine learning for operational digital twins. Environmental Modelling & Software, 148, 105274. https://doi.org/10.1016/j.envsoft.2021.105274

Van den Bossche, J., Jordahl, K., Fleischmann, M., Richards, M., McBride, J., Wasserman, J., …, & Gardiner, J. (2024). geopandas/geopandas: v1.0.1 (v1.0.1). Zenodo. https://doi.org/10.5281/zenodo.12625316

Verdouw, C., Tekinerdogan, B., Beulens, A., & Wolfert, S. (2021). Digital twins in smart farming. Agricultural Systems, 189, 103046. https://doi.org/10.1016/j.agsy.2020.103046

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., …, & SciPy 1.0 Contributors (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods 17, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Weiss, M., Baret, F., & Jay, S. (2020). S2ToolBox level 2 products LAI, FAPAR, FCOVER. EMMAH-CAPTE, INRAe Avignon. Available online at https://step.esa.int/docs/extra/ATBD_S2ToolBox_V2.0.pdf (accessed November 2024).