

Low-cost GUI-based deep learning deployment solutions

# Getting started with napari in phenotyping

Herearii METUAREA, Engineer INRAe France

27th June 2023

Institut Agro Rennes-Angers, B202

# Table of contents

Context

Napari

Plugins

Widget

# IPPN : Napari as a tool for phenotyping

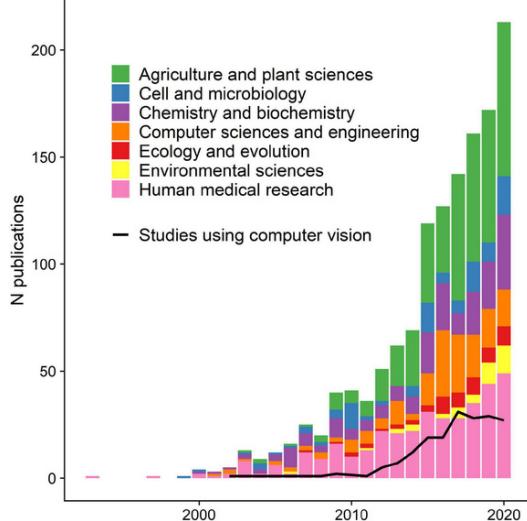
## Context

Napari

## Plugins

# Widget

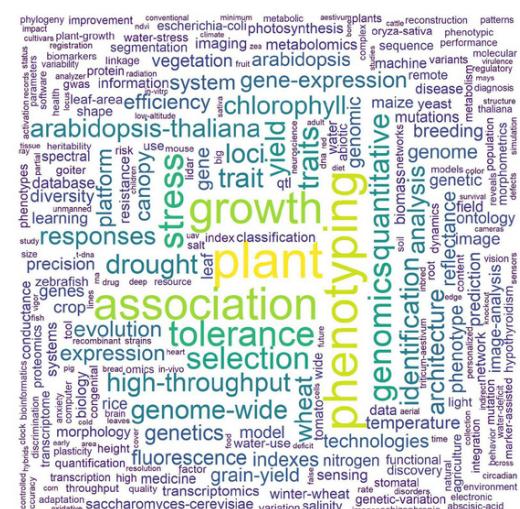
# Deep learning: increasingly used



Current state of phenomics research (left) and 500 most used keywords from the papers presented in the left panel (right)

Lürig, Moritz D., et al, 2021

# Two platforms for deep learning models





**deep**ImageJ

# Google Colaboratory

Google's right to read and use personal data

# IPPN : Napari as a tool for phenotyping

Context

Napari

Plugins

Widget



napari

n-dimensional data viewer in Python

The screenshot displays the Napari application window. On the left, a large 3D volume rendering shows green and magenta structures. Below it, a layer list panel contains 'nuclei' and 'membrane'. On the right, there are four smaller panels illustrating different features:

- Annotation:** Shows a grayscale image with a black mask overlaid, along with a controls panel.
- Segmentation:** Displays a 3D reconstruction of plant roots with colored segments and bounding boxes.
- Process heavy data:** Shows a 3D point cloud with many small colored dots.
- Tracking:** Displays a 3D reconstruction of plant roots with colored segments and bounding boxes, similar to the Segmentation panel but with more complex tracking lines.

At the top of the main window, a toolbar includes icons for file operations like Open, Save, and Import, as well as zoom and navigation controls. The title bar reads 'napari'.

open-source, community developed

Annotation

Segmentation

Process heavy data

Tracking

# IPPN : Napari as a tool for phenotyping

Context

Napari

Plugins

Widget



napari

Combining interactive annotation and segmentation algorithms

```
In [43]: from skimage import data
...: from skimage import filters
...: from skimage import segmentation
...: from skimage import morphology
...:
...: import napari
...:
...: coins = data.coins()
...:
...: viewer = napari.view_image(coins, name='coins')
...:
...: edges = filters.sobel(coins)
...:
...: edges_layer = viewer.add_image(edges, name='edges', colormap='magenta',
...:                                blending='additive')
...:
...: pts_layer = viewer.add_points(name='seeds', size=5)
...: pts_layer.mode = 'add'
...: # annotate the background and all the coins, in that order
...:
...:
In [44]: coordinates = pts_layer.data
coordinates_int = np.round(coordinates).astype(int)
...:
markers_raw = np.zeros_like(coins)
markers_raw[tuple(coordinates_int.T)] = 1 + np.arange(len(coordinates))
...:
# raw markers might be in a little watershed "well".
markers = morphology.dilation(markers_raw, morphology.disk(5))
...:
segments = segmentation.watershed(edges, markers=markers)
...:
labels_layer = viewer.add_labels(segments - 1) # make background 0
...:
...:
```

In [45]:



enter paint or fill mode to edit labels

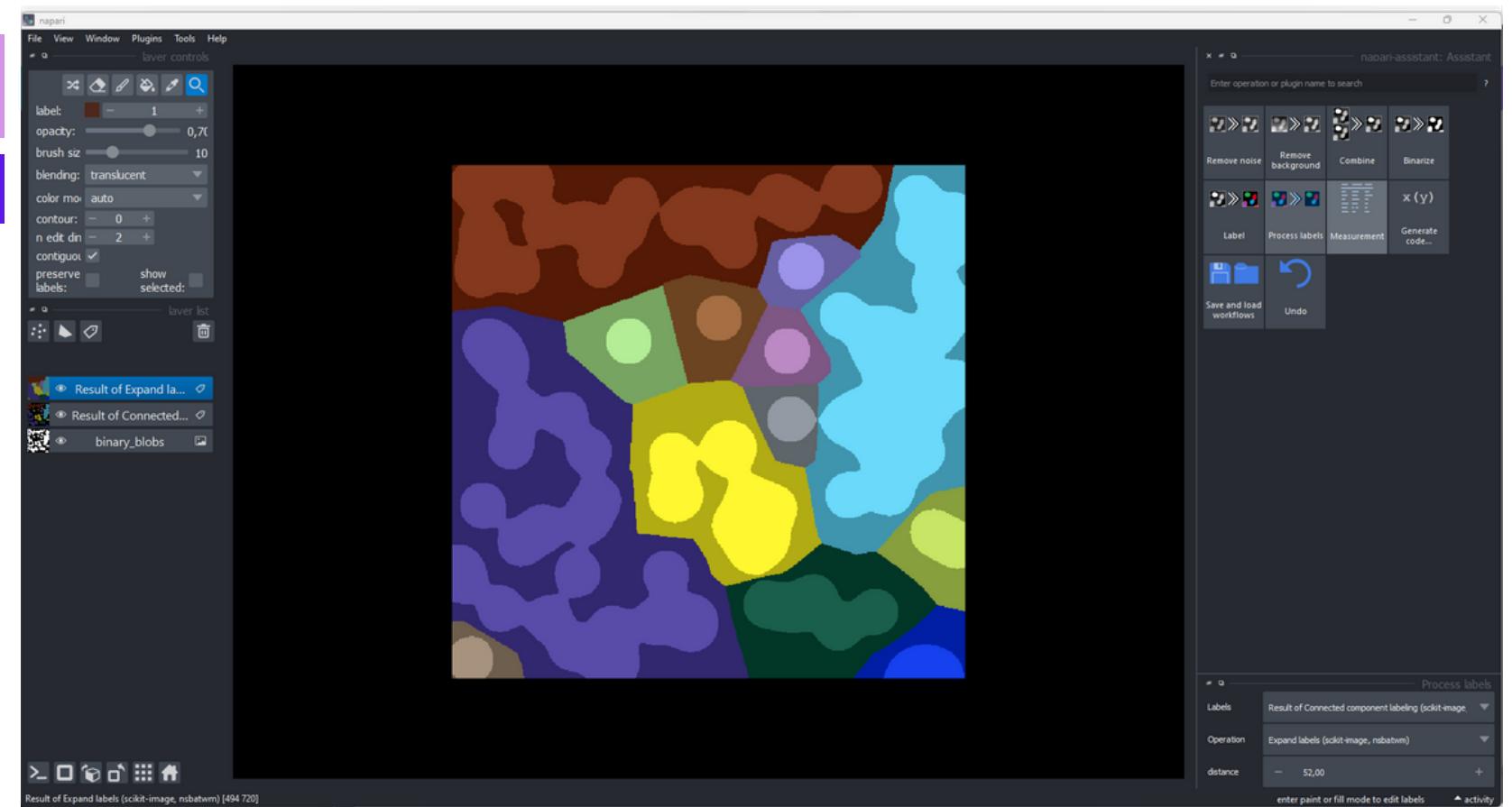
# IPPN : Napari as a tool for phenotyping

[Context](#)
[Napari](#)
[Plugins](#)
[Widget](#)


napari

Combining interactive annotation and segmentation algorithms

Annotate label and process labels



# IPPN : Napari as a tool for phenotyping

[Context](#)
[Napari](#)
[Plugins](#)
[Widget](#)


napari

Combining interactive annotation and segmentation algorithms

Annotate label and process labels

Deep-Learning (denoising, cell+nuclei segmentation)



# IPPN : Napari as a tool for phenotyping

[Context](#)
[Napari](#)
[Plugins](#)
[Widget](#)

Plugin: napari-process-points-and-surfaces

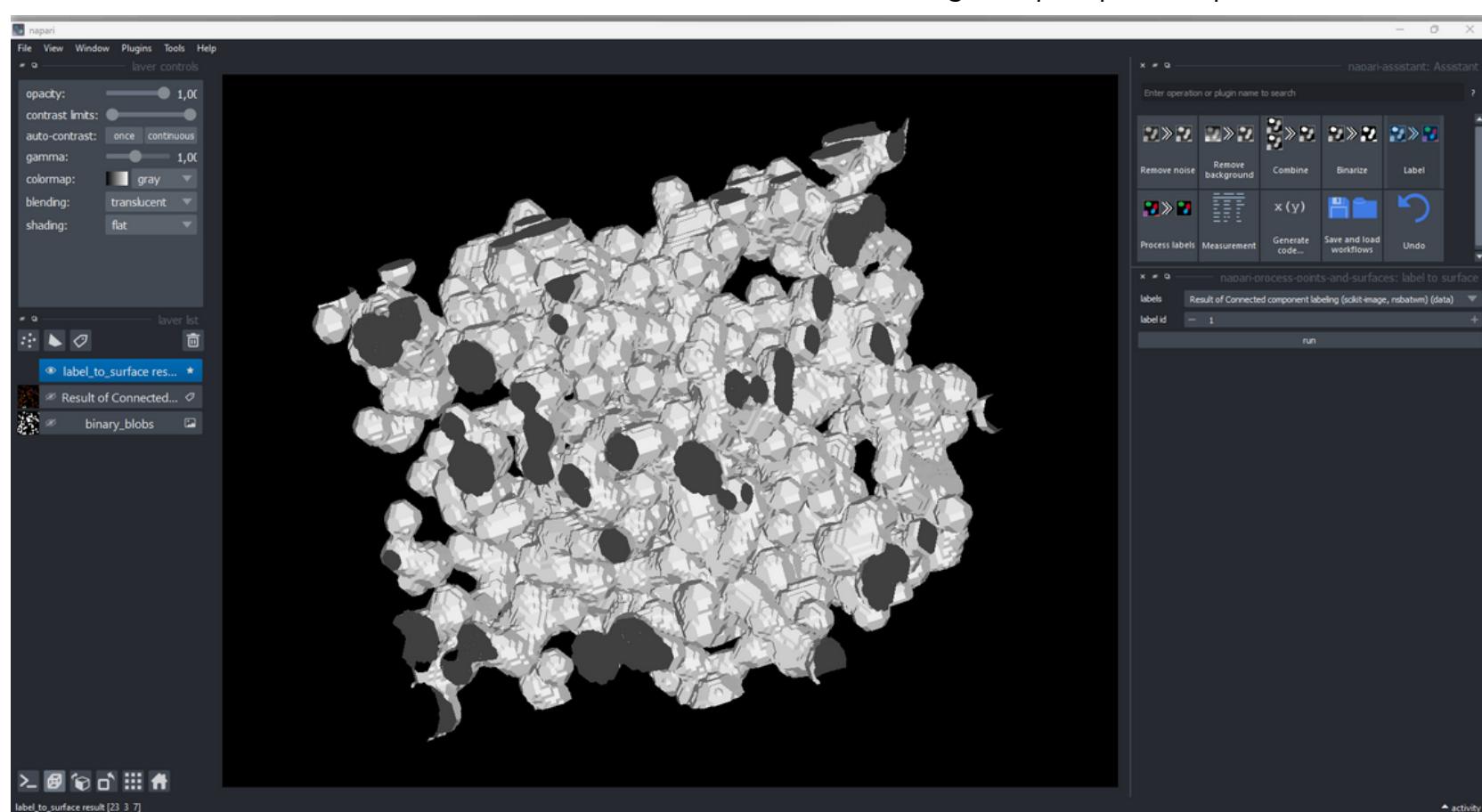

**napari**

Combining interactive annotation and segmentation algorithms

Annotate label and process labels

Deep-Learning (denoising, cell+nuclei segmentation)

Surface extraction &amp; analysis



# IPPN : Napari as a tool for phenotyping

Context

Napari

Plugins

Widget



napari

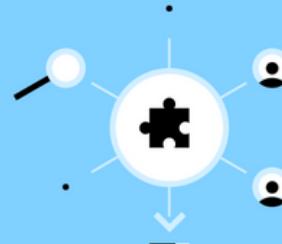
326 plugins

napari hub

Plugins

Collections

## Discover, install, and share napari plugins



- Discover plugins that solve your image analysis challenges
- Learn how to install into napari
- Share your image analysis tools with napari's growing community

Search for a plugin by keyword or author



# IPPN : Napari as a tool for phenotyping

Context

Napari

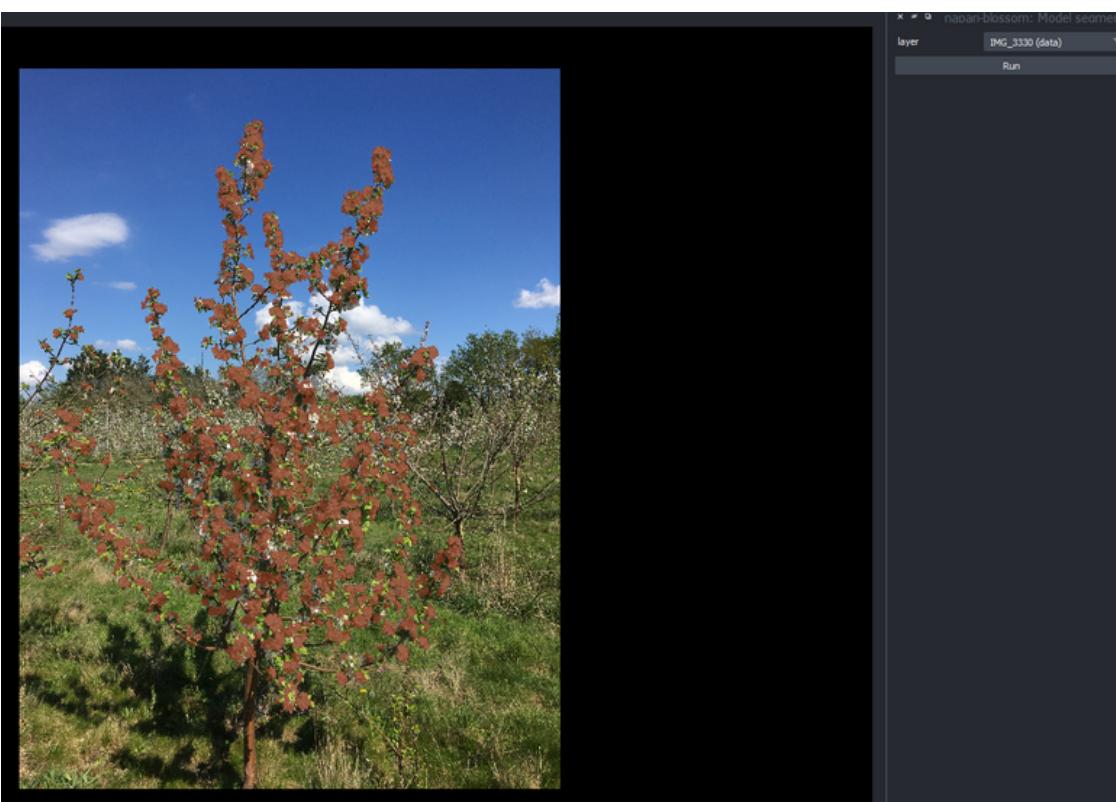
Plugins

Widget

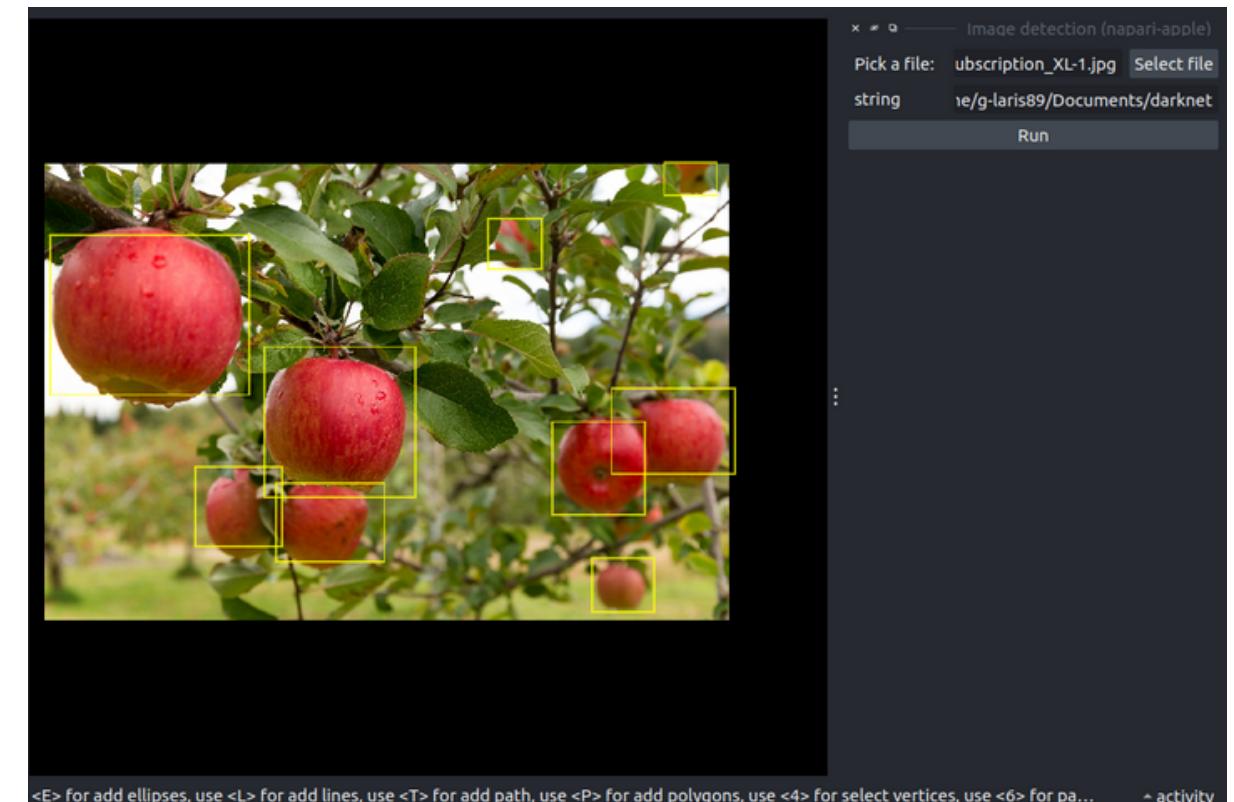


napari

Plugin: napari-blossom



Detection of apple flowering



Detection of apple

# IPPN : Napari as a tool for phenotyping

Context

Napari

Plugins

Widget



napari

## Which tools can be included in a plugin ?

Reader

Widget

Writer

Add instructions for special input data

Add instructions for processing data with a user interface

Add instructions for special output data

---

Match a set of incomplete ground truths to an image sequence

Applying a deep learning model on RGB image sequence

Save image sequence in compressed file

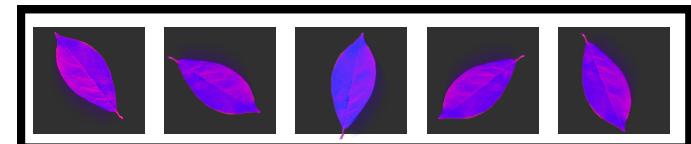
RGB image sequence:



RGB image sequence:



Mask image sequence:



Ground truth image sequence:



Mask image sequence:



.zip

Plugin: workshop-demo

Context

Napari

Plugins

Widget



napari

## Which tools can be included in a plugin ?

Reader

Add instructions for special input data

Widget

Add instructions for processing data with a user interface

Writer

Add instructions for special output data

Match a set of incomplete ground truths to an image sequence

RGB image sequence:



Ground truth image sequence:



Applying a deep learning model on RGB image sequence

RGB image sequence:

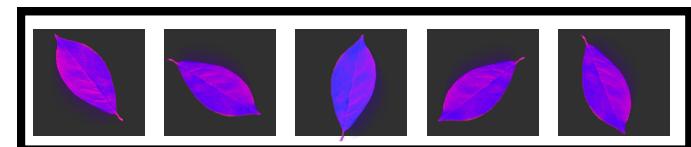


Mask image sequence:



Save image sequence in compressed file

Mask image sequence:



.zip

Plugin: workshop-demo

# IPPN : Napari as a tool for phenotyping

[Context](#)
[Napari](#)
[Plugins](#)
[Widget](#)


napari

[Create your plugin package](#)
[Import the codes into the widget.py file](#)
[Connect widget code to napari](#)
[Add dependencies in metadata](#)
[Add some test](#)
[Deploy](#)

## How to design a plugin and a widget ?

Generate minimal napari plugin repository

cookiecutter <https://github.com/napari/cookiecutter-napari-plugin>

```
full_name [Napari Developer]: guest-0000
email [yourname@example.com]: guest-0000@gmail.com
github_username_or_organization [githubuser]: guest-0000_pizalliol
plugin_name [napari-foobar]: napari-thresholds
Select github_repository_url:
1 - https://github.com/guest-0000_pizalliol/napari-thresholds
2 - provide later
Choose from 1, 2 [1]:
module_name [napari_thresholds]: napari_thresholds
display_name [napari FooBar]: Thresholds
short_description [A simple plugin to use with napari]: Several thresholds available
include_reader_plugin [y]: n
include_writer_plugin [y]: n
include_sample_data_plugin [y]: n
include_dock_widget_plugin [y]: y
use_git_tags_for_versioning [n]: n
install_precommit [n]: n
Select license:
1 - BSD-3
2 - MIT
3 - Mozilla Public License 2.0
4 - Apache Software License 2.0
5 - GNU LGPL v3.0
6 - GNU GPL v3.0
Choose from 1, 2, 3, 4, 5, 6 (1, 2, 3, 4, 5, 6) [1]: 1
```

# IPPN : Napari as a tool for phenotyping

Context

Napari

Plugins

Widget



napari

Create your plugin package

Import the codes into the  
widget.py file

Connect widget code to  
napari

Add dependencies in  
metadata

Add some test

Deploy

## How to design a plugin and a widget ?

Write your code into a function  
and adapt to napari convention

napari.types

ImageData

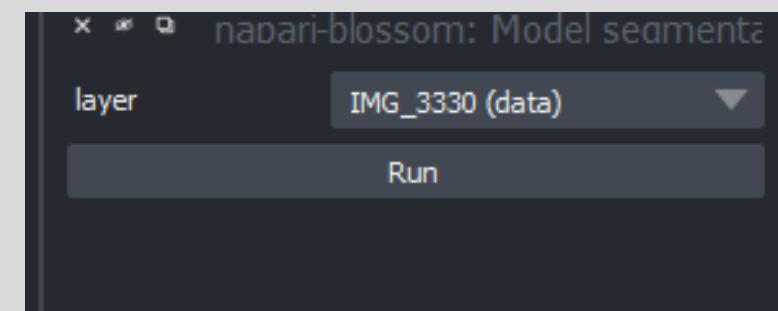
code

napari.types

LabelsData



Use magicgui library to create  
user interface



widget.py

# IPPN : Napari as a tool for phenotyping

[Context](#)
[Napari](#)
[Plugins](#)
[Widget](#)


napari

## How to design a plugin and a widget ?

Create your plugin package

Import the codes into the  
widget.py file

Connect widget code to  
napari

Add dependencies in  
metadata

Add some test

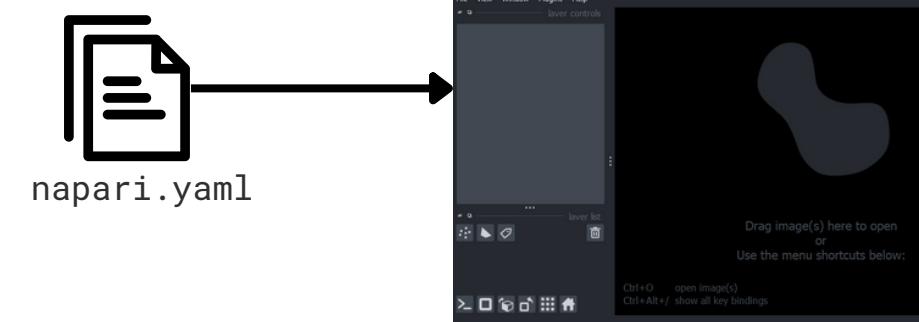
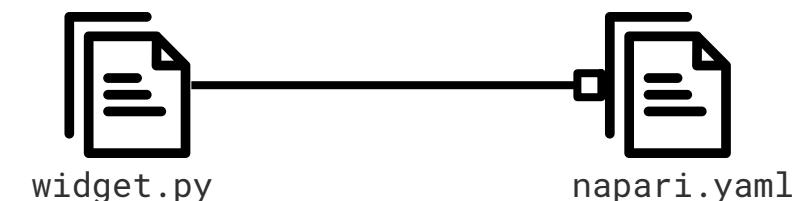
Deploy

```

name: napari-thresholds
display_name: Thresholds
contributions:
    commands:
        - id: napari-thresholds.my_widget #must be unique !
          python_name: napari_thresholds._widget:threshold_f
          title: Thresholds
    widgets:
        - command: napari-thresholds.my_widget #identity backend
          display_name: Thresholds
  
```



napari.yaml



napari.yaml

# IPPN : Napari as a tool for phenotyping

Context

Napari

Plugins

Widget



napari

## How to design a plugin and a widget ?

Create your plugin package

Import the codes into the widget.py file

Connect widget code to napari

Add dependencies in metadata

Add some test

Deploy



setup.cfg

```
[options]
packages = find:
install_requires =
    numpy
    magicgui
    qtpy
    scikit-image
    napari

python_requires = >=3.8
include_package_data = True
package_dir =
    =src

# add your package requirements here

[options.packages.find]
where = src

[options.entry_points]
napari.manifest =
    napari-thresholds = napari_thresholds:napari.yaml
```

Determine dependencies

Determine the repository containing codes

Determine napari-thresholds is napari plugin

# IPPN : Napari as a tool for phenotyping

Context

Napari

Plugins

Widget



napari

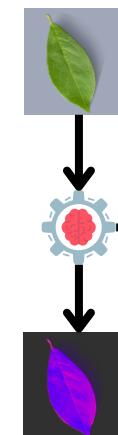
## How to design a plugin and a widget ?

Create your plugin package

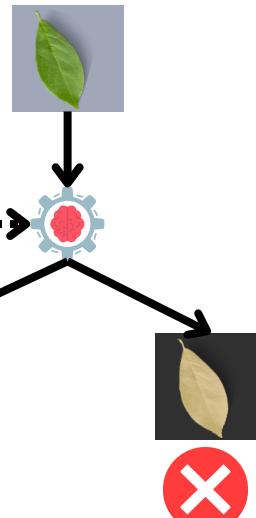
Instruction to be sure the widget works well whatever the change made

Import the codes into the  
widget.py file

code in widget

Connect widget code to  
napari

Test: check if output is violet leaf

Add dependencies in  
metadata

Add some test

check the output

Deploy



test\_widget.py

# IPPN : Napari as a tool for phenotyping

[Context](#)
[Napari](#)
[Plugins](#)
[Widget](#)


napari

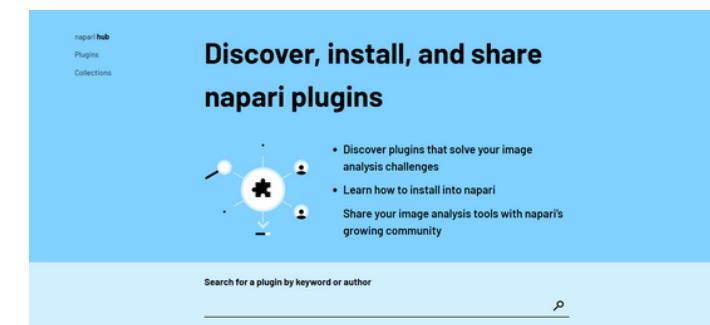
[Create your plugin package](#)
[Import the codes into the widget.py file](#)
[Connect widget code to napari](#)
[Add dependencies in metadata](#)
[Add some test](#)
[Deploy](#)

## How to design a plugin and a widget ?

Requirements:



1. Add napari project in GitHub (public access)
2. Generate API token
3. Add API token in GitHub as secret key
4. Create a build in napari-thresholds folder
5. Upload package to the PyPI



# Exercise

Context

Napari

Plugins

Widget



napari

Go to this page: [https://github.com/hereariim/IPPN\\_napari](https://github.com/hereariim/IPPN_napari)

**hereariim / IPPN\_napari**

Type  to search

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

**IPPN\_napari** Public

main 1 branch 0 tags

Go to file Add file > Code About

This is a training tutorial to learn how to integrate deep learning model into napari plugin from scratch

Readme Activity 0 stars 2 watching 0 forks

Releases No releases published Create a new release

Packages No packages published Publish your first package

Languages Jupyter Notebook 90.4% Python 9.6%

Suggested Workflows Actions Importer Set up Automatically convert CI/CD files to YAML for GitHub Actions

hereariim Presentation

- Exercise-1\_Getting started
- Exercise-2\_Getting started plugin
- Exercise-3\_My first plugin
- Exercise-4\_My first widget
- Exercise-5\_Do it yourself
- Extra-Deploy\_in\_builtin
- images-credit
- Introduction.pdf
- README.md

readme 1 2 days ago Update README.md yesterday Update README.md yesterday Update README.md yesterday Update README.md yesterday Update README.md 18 minutes ago logos 2 days ago Presentation 4 minutes ago Update README.md yesterday

IPPN : Napari as a tool for phenotyping

Institut Agro Rennes Angers université d'angers INRAE PHENOME EMPHASIS FRANCE

IPPN Phymea systems Sony CSL napari

This document is part of the Hands on session of the IPPN 2023. The aim of this document is to show the value of napari in plant phenotyping.

It is not intended to be a training session. That would require a lot of time to fully explore napari. The Hands on session for napari is a three-part discovery activity: