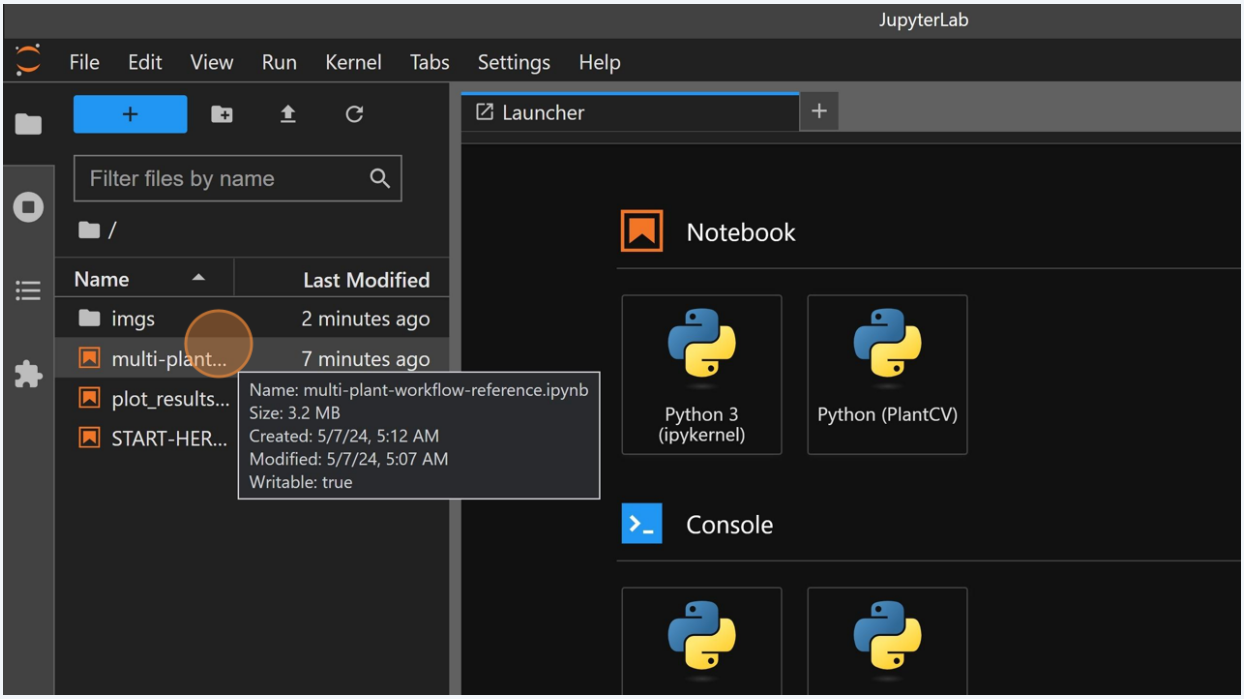
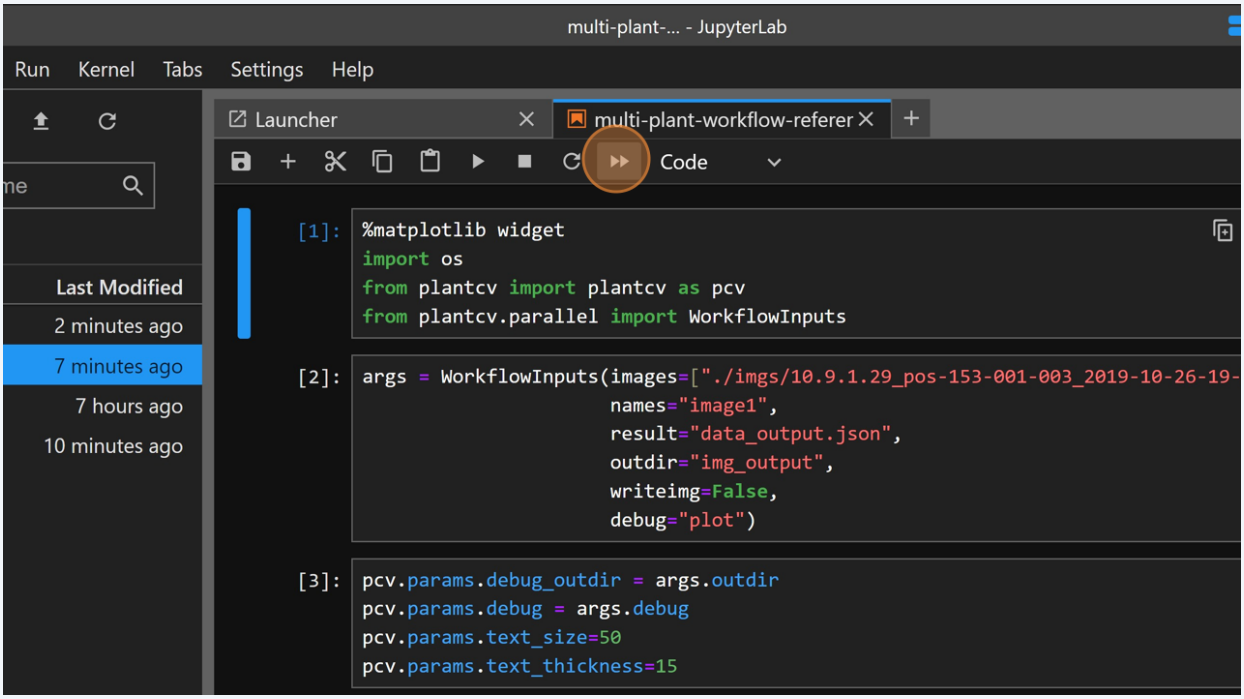
How to Parallelize Your Workflows Using PlantCV

This guide provides step-by-step instructions on how to parallelize workflows using PlantCV. By following these steps, users can optimize their workflow efficiency and increase productivity.

1 Begin with a prebuilt workflow. Ideally you have trained your workflow using a step-up image training protocol (i.e. training your workflow on **1 image > 2 images > 4 images > 10 images > 20 images**)

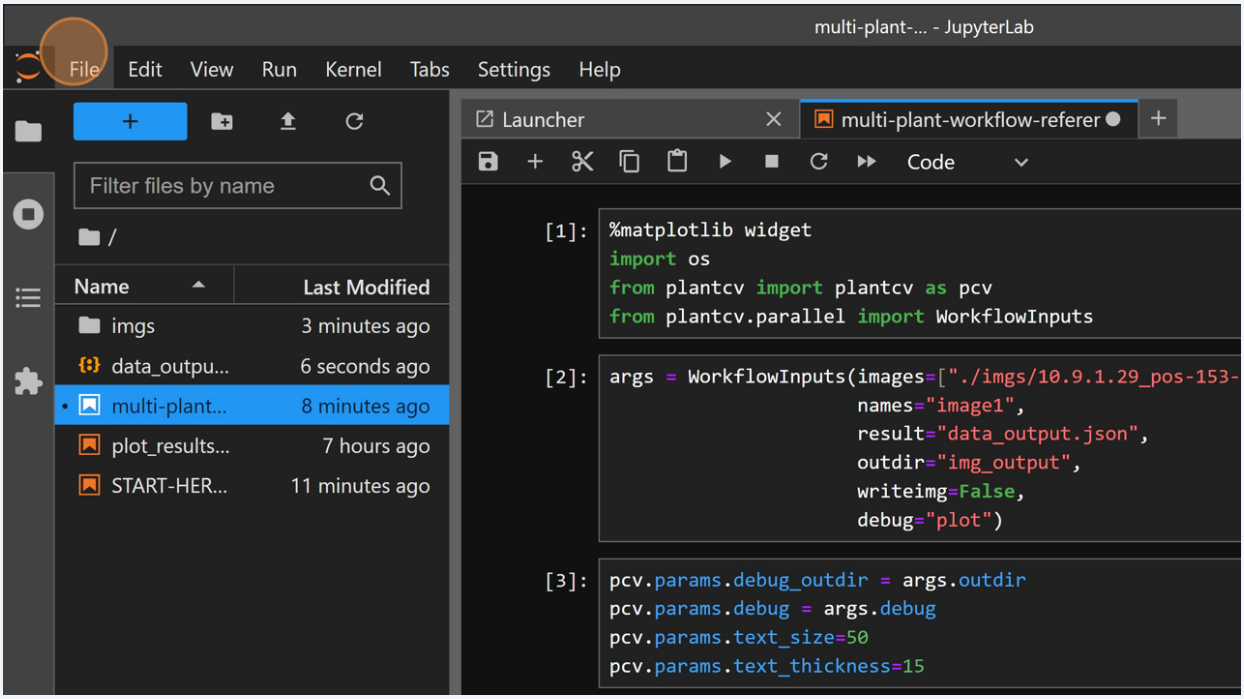
NOTE: Make sure you have ran your workflow and have produced a JSON output file.

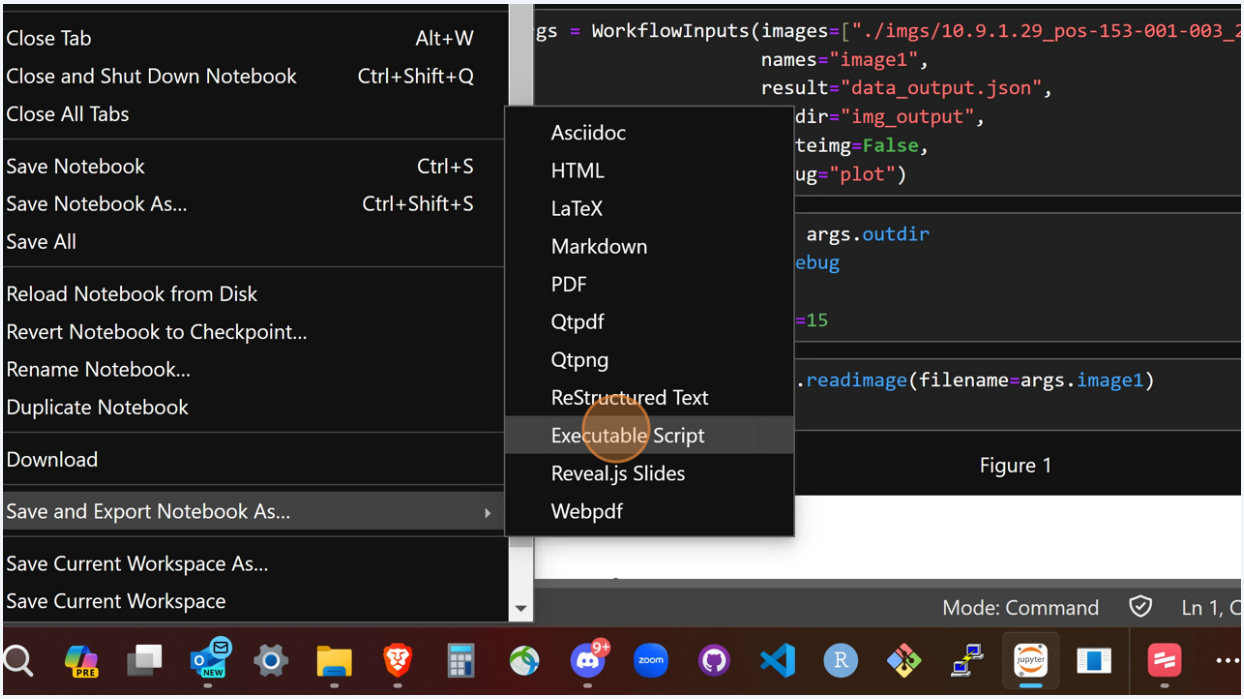


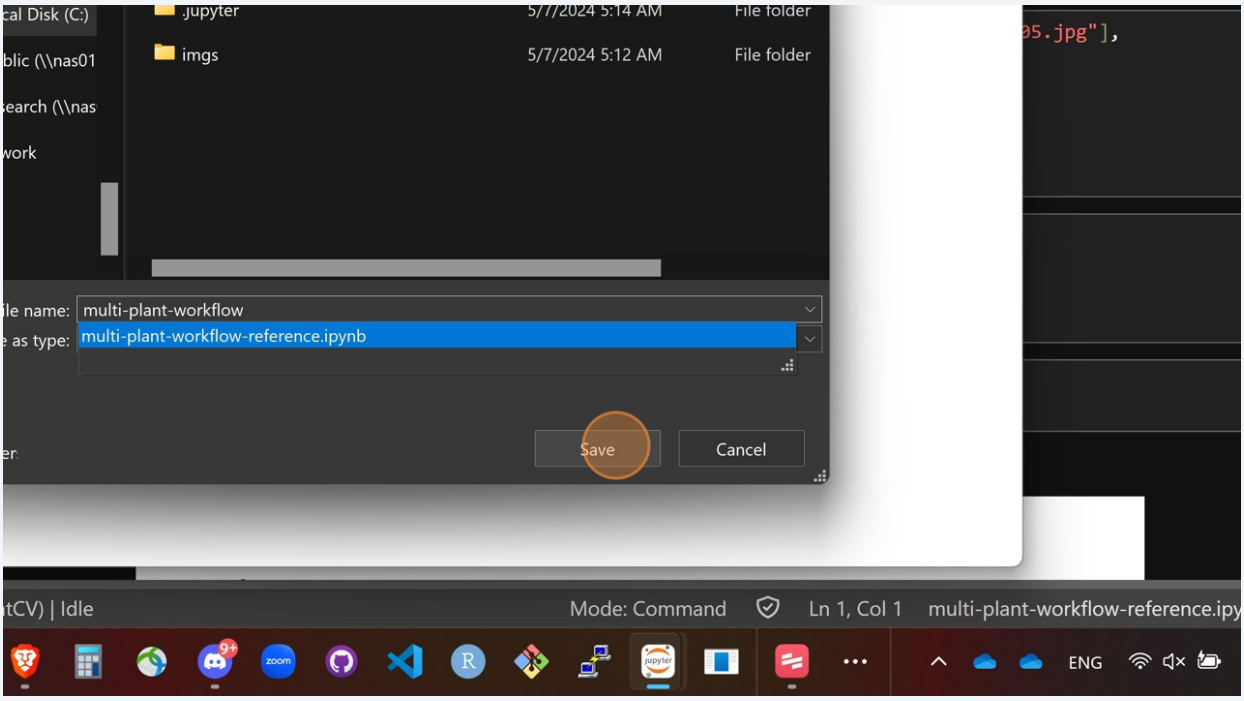


2 Convert your Jupyter Notebook (**.ipynb**) to an executable script (a Python script **.py**) selecting **File > Save and Export Notebook As... > Executable Script**

Save your Python script in the same directory as your Jupyter notebook for easy access. You may need to adjust the title of your script so it is concise (in the example, I am using a reference document and do not want **"reference"** in my final script.



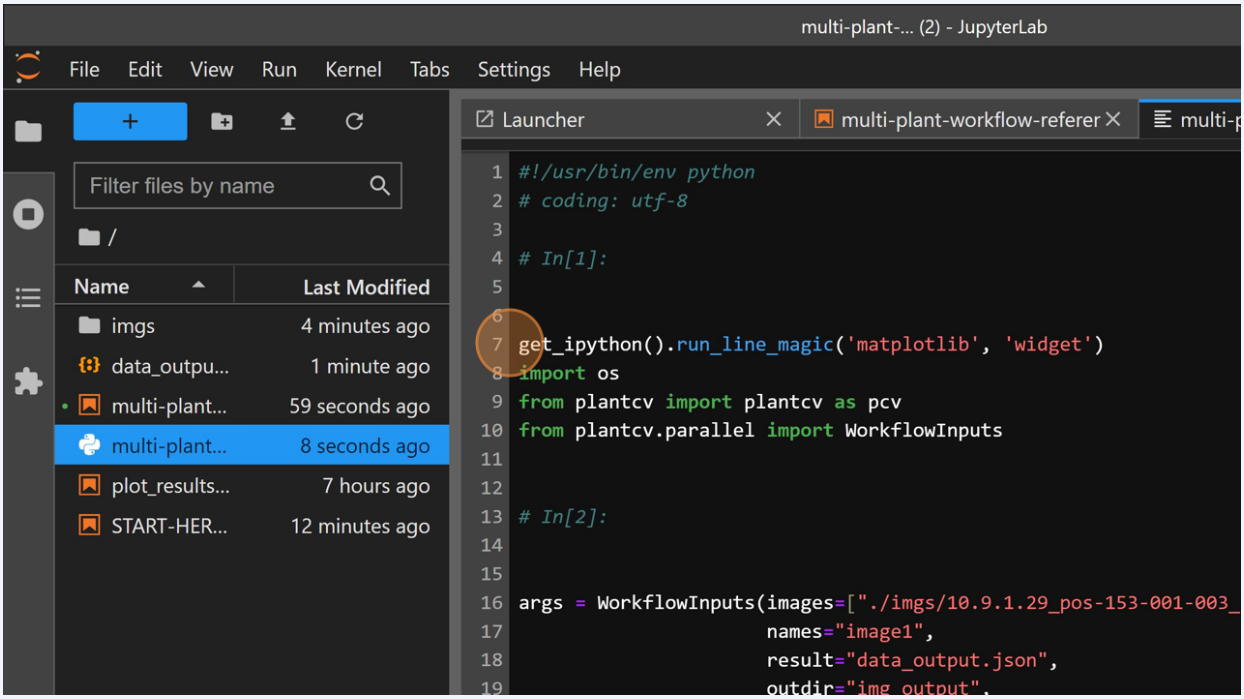

Close Tab Alt+W
Close and Shut Down Notebook Ctrl+Shift+Q
Close All Tabs
Save Notebook Ctrl+S
Save Notebook As... Ctrl+Shift+S
Save All
Reload Notebook from Disk
Revert Notebook to Checkpoint...
Rename Notebook...
Duplicate Notebook
Download
Save and Export Notebook As...
Save Current Workspace As...
Save Current Workspace
Asciidoc
HTML
LaTeX
Markdown
PDF
Qtpdf
Qtpng
ReStructured Text
Executable Script
Reveal.js Slides
Webpdf
gs = WorkflowInputs(images=["./imgs/10.9.1.29_pos-153-001-003_2
names="image1",
result="data_output.json",
dir="img_output",
teimg=False,
ug="plot")
args.outdir
ebug
=15
.readimage(filename=args.image1)
Figure 1
Mode: Command
Ln 1, C


.jupyter 5/7/2024 5:14 AM File folder
imgs 5/7/2024 5:12 AM File folder
multi-plant-workflow
multi-plant-workflow-reference.ipynb
Save
Cancel
05.jpg"],
Mode: Command
Ln 1, Col 1
multi-plant-workflow-reference.ipy
ENG

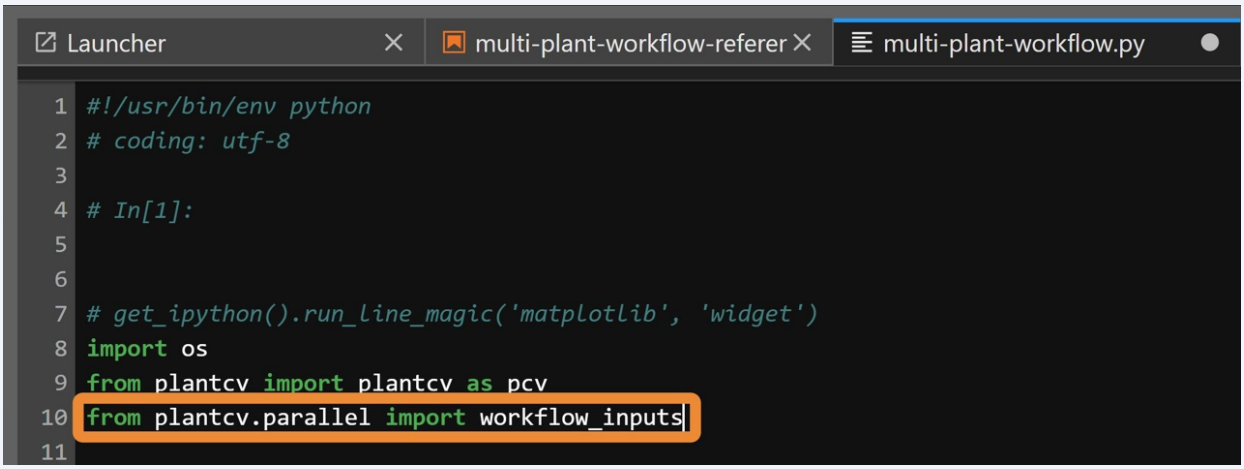
3 Open your new Python script. We will have to make some edits to make it a functioning workflow script.

The first thing we will want to do is "comment out" (adding an **octothorpe**, commonly referred to as a *hash tag* or *number sign*). the text on line 7: **get_python().run_line_magic('matplotlib', 'widget')**

You can also comment out an entire line of code using **CTRL + /**.

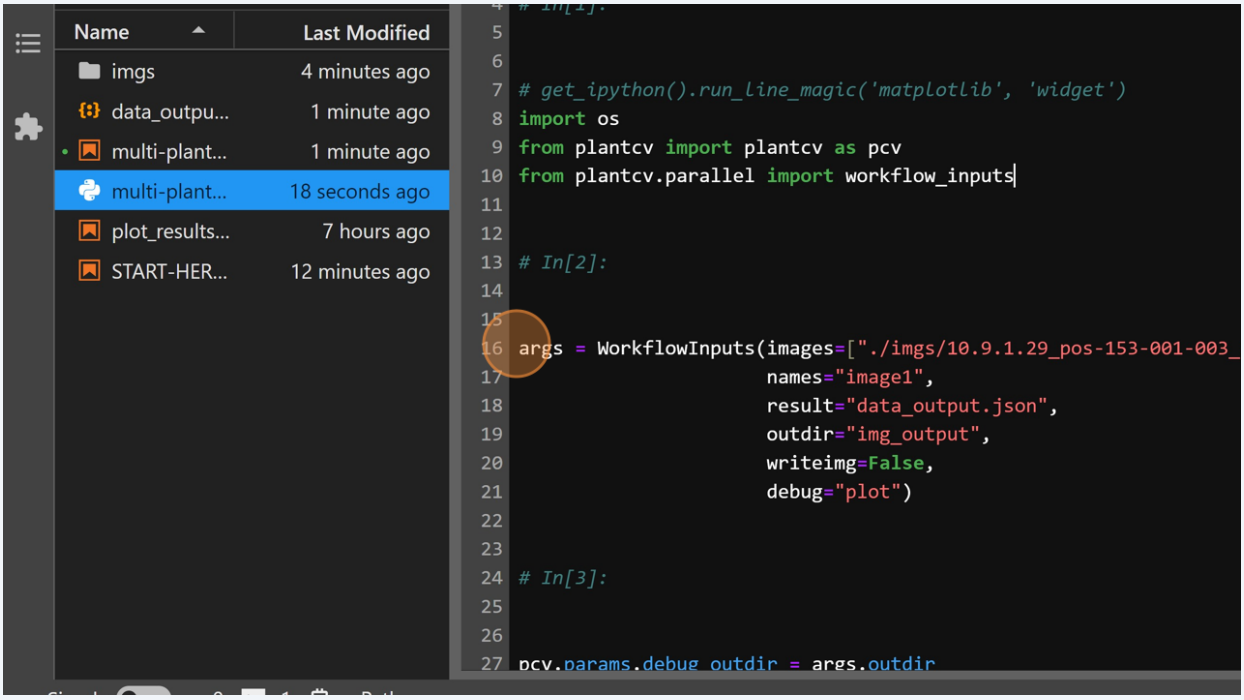


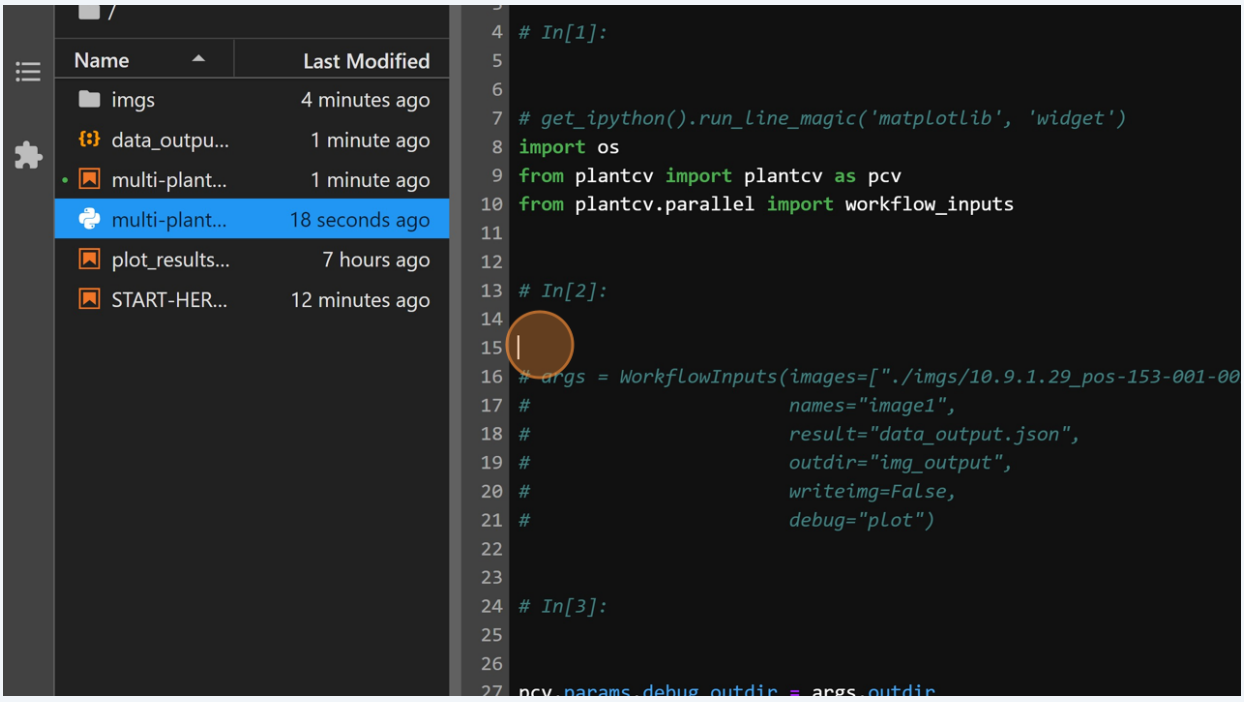
4 We need to make a few changes to our import statements. Namely, we need to change the step **from plantcv.parallel import WorkflowInputs** to **from plantcv.parallel import workflow_inputs**.



5 Next, we will change our argument definitions for this workflow by "commenting out" the our **WorkflowInputs** parameters. Remember that you can comment out an entire line by using **CTRL + /**.

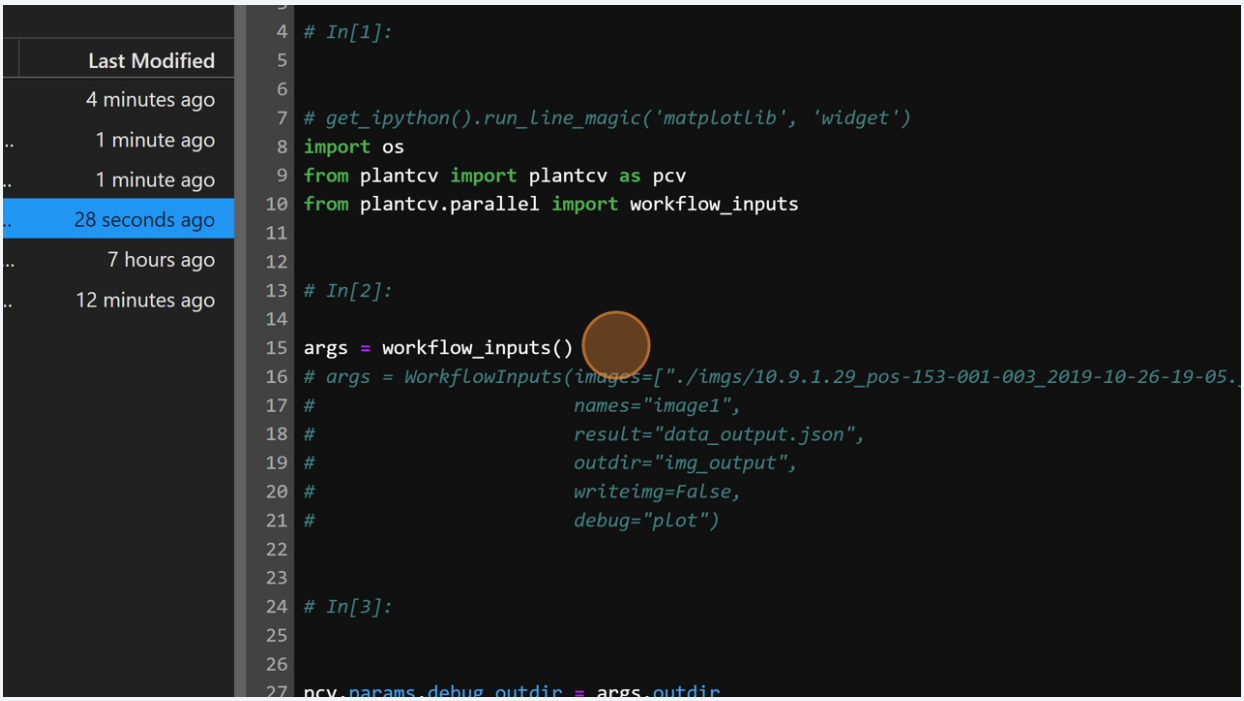
See the image below:



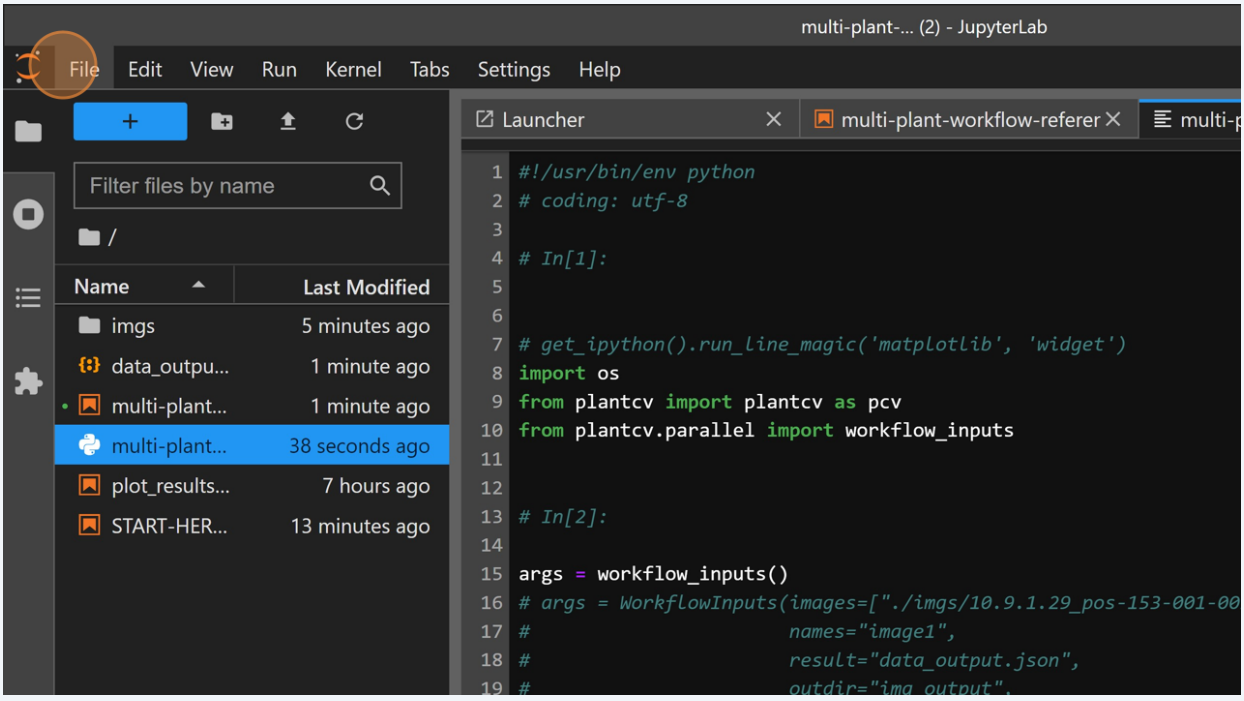


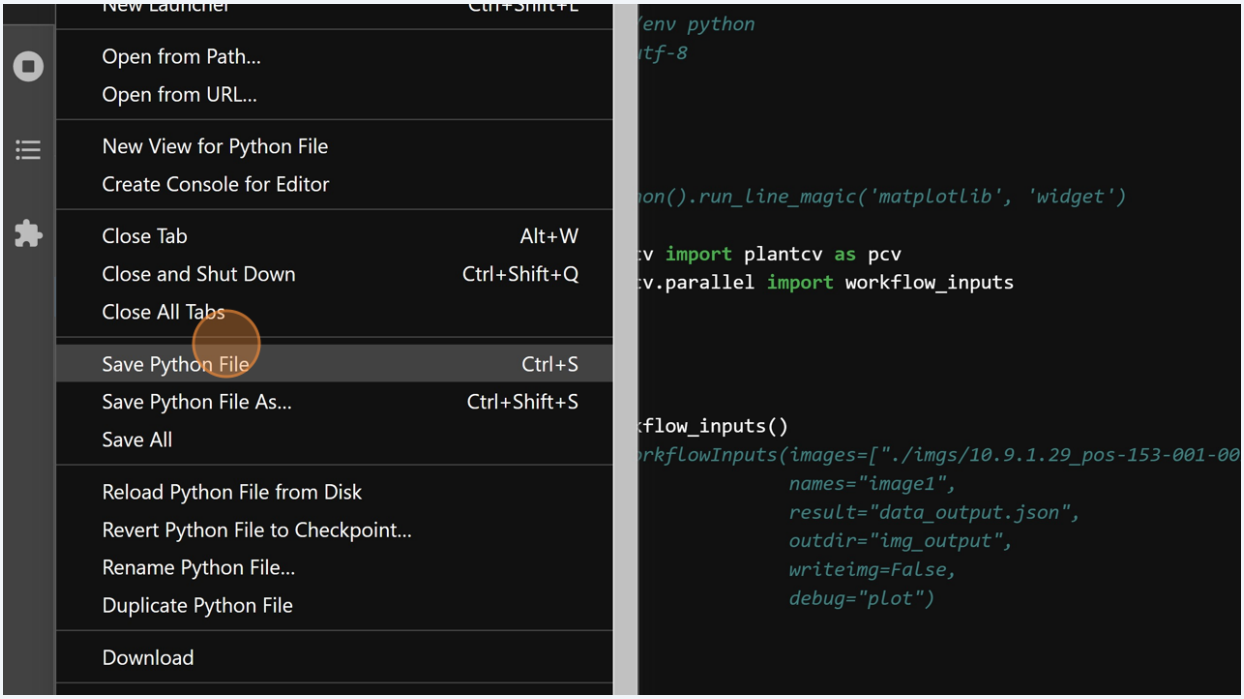
6 Above the commented out **WorkflowInputs**, we are going to add a new line that will store workflow inputs into **args** to support parallel workflow execution:

args = workflow_inputs()



7 Save the changes you made to your Python script.





In the next step we will have to change the working directory in terminal to the location where our scripts are. There are a couple of ways to obtain the file path for your working directory:

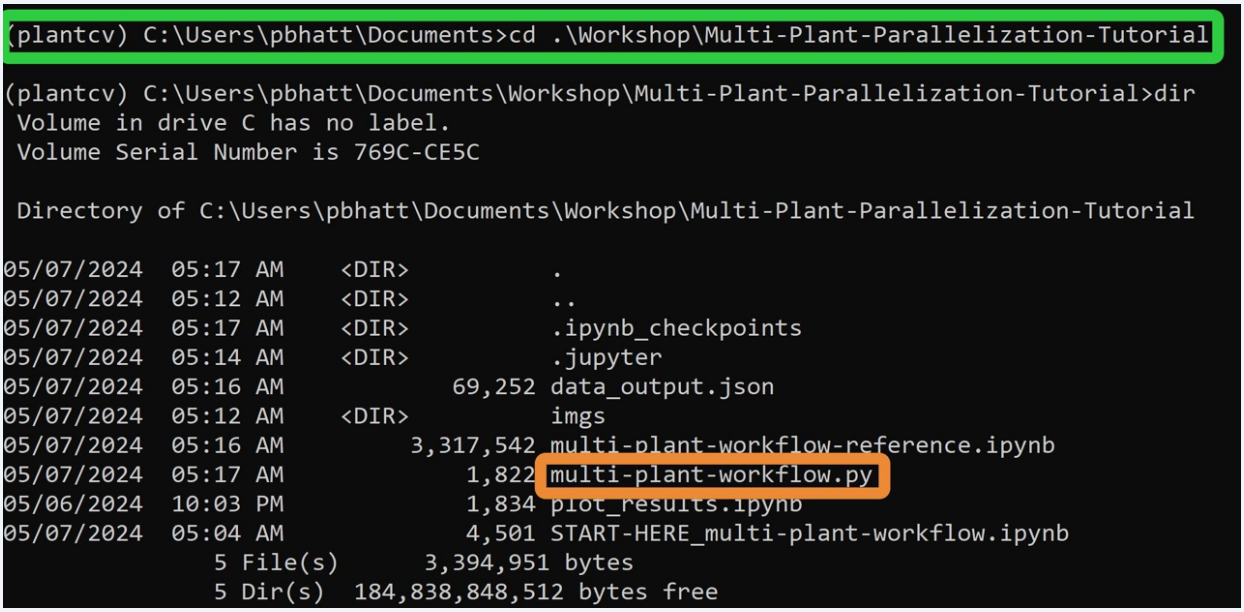
1. If you are working on your local device and have access to your system's complete root directory, then you can right-click on **multi-plant-workflow.py** and select **Copy Path**. This will copy the relative path of that file. We will use this information to change our working directory.

2. If you do not have access to the full file path, then open your file explorer and copy the address at the top.

NOTE: You may want to copy the file path to a document so you can see how much of the file path is copied (the full file path will not be copied).

8 In this guide, the current working directory is local to the root directory, so only a portion of the path was needed to point our conda environment to the current working directory. Remember that to change directories, you need to use the operator **cd** (See image below, outlined in green).

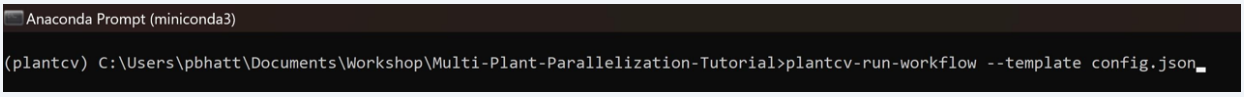
Once you have changed your working directory in conda, type **dir** to view the contents of the directory. Make sure that your executable Python script is in the correct location. (See image below, outlined in orange).



9 Now we need to create our parallel configuration file by typing:

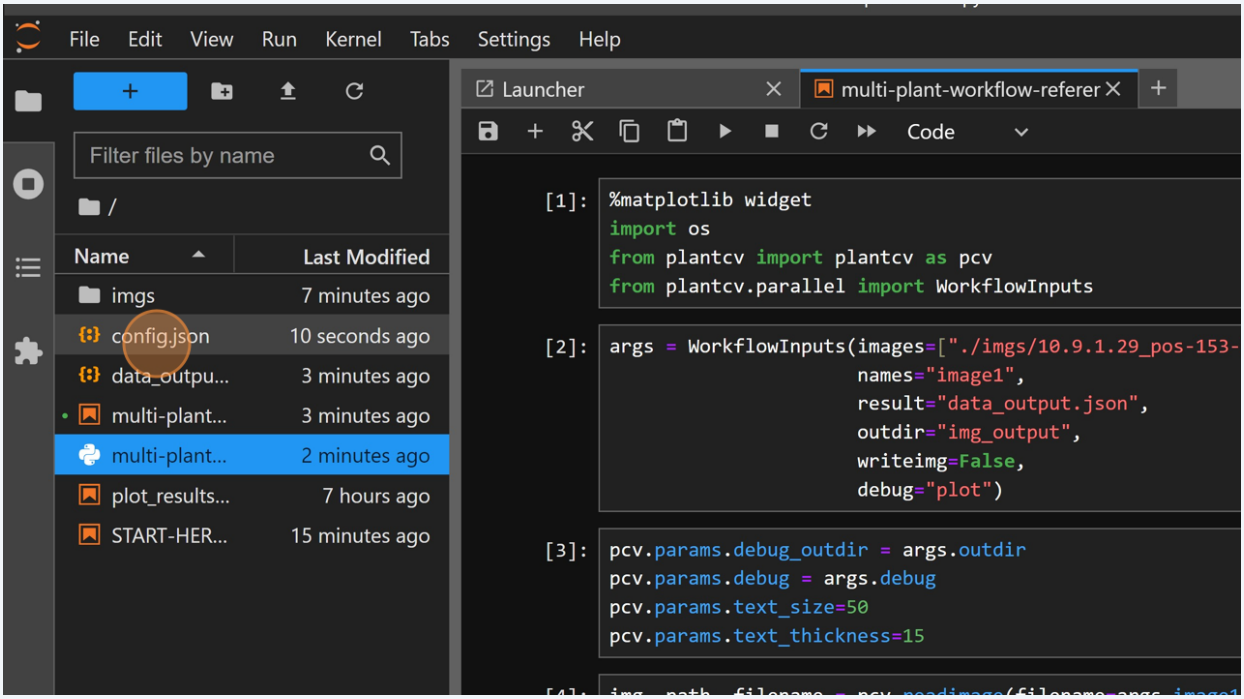
plantcv-run-workflow --template config.json

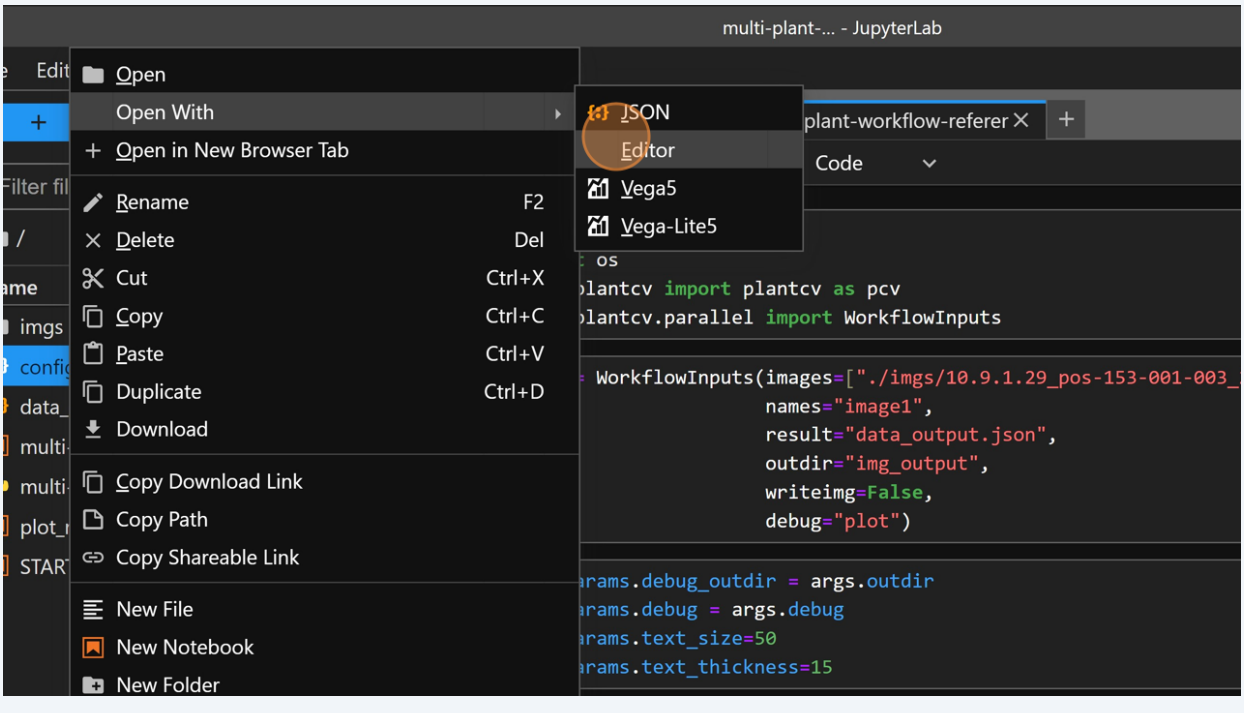
Since we have changed our working directory to our current folder, you should see **config.json** appear in the file explorer like shown below.



10 Since we have changed our working directory to our current folder, you should see **config.json** appear in the file explorer like shown below.

We will need to edit **config.json** so that we can configure the parallel analysis. Right-click **config.json** and hover over **Open With** and select **Editor** to make changes to the file.


File Edit View Run Kernel Tabs Settings Help
Launcher
multi-plant-workflow-referer
Code
Filter files by name
/
Name
Last Modified
imgs 7 minutes ago
config.json 10 seconds ago
data_outpu... 3 minutes ago
multi-plant... 3 minutes ago
multi-plant... 2 minutes ago
plot_results... 7 hours ago
START-HER... 15 minutes ago
[1]: %matplotlib widget
import os
from plantcv import plantcv as pcv
from plantcv.parallel import WorkflowInputs
[2]: args = WorkflowInputs(images=["./imgs/10.9.1.29_pos-153-
names="image1",
result="data_output.json",
outdir="img_output",
writeimg=False,
debug="plot")
[3]: pcv.params.debug_outdir = args.outdir
pcv.params.debug = args.debug
pcv.params.text_size=50
pcv.params.text_thickness=15


multi-plant-... - JupyterLab
Open
Open With
Open in New Browser Tab
Rename F2
Delete Del
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Duplicate Ctrl+D
Download
Copy Download Link
Copy Path
Copy Shareable Link
New File
New Notebook
New Folder
JSON Editor
Vega5
Vega-Lite5
plant-workflow-referer
Code
os
plantcv import plantcv as pcv
plantcv.parallel import WorkflowInputs
WorkflowInputs(images=["./imgs/10.9.1.29_pos-153-001-003_
names="image1",
result="data_output.json",
outdir="img_output",
writeimg=False,
debug="plot")
rams.debug_outdir = args.outdir
rams.debug = args.debug
rams.text_size=50
rams.text_thickness=15

11 In Editor, make the following changes to **config.json**:

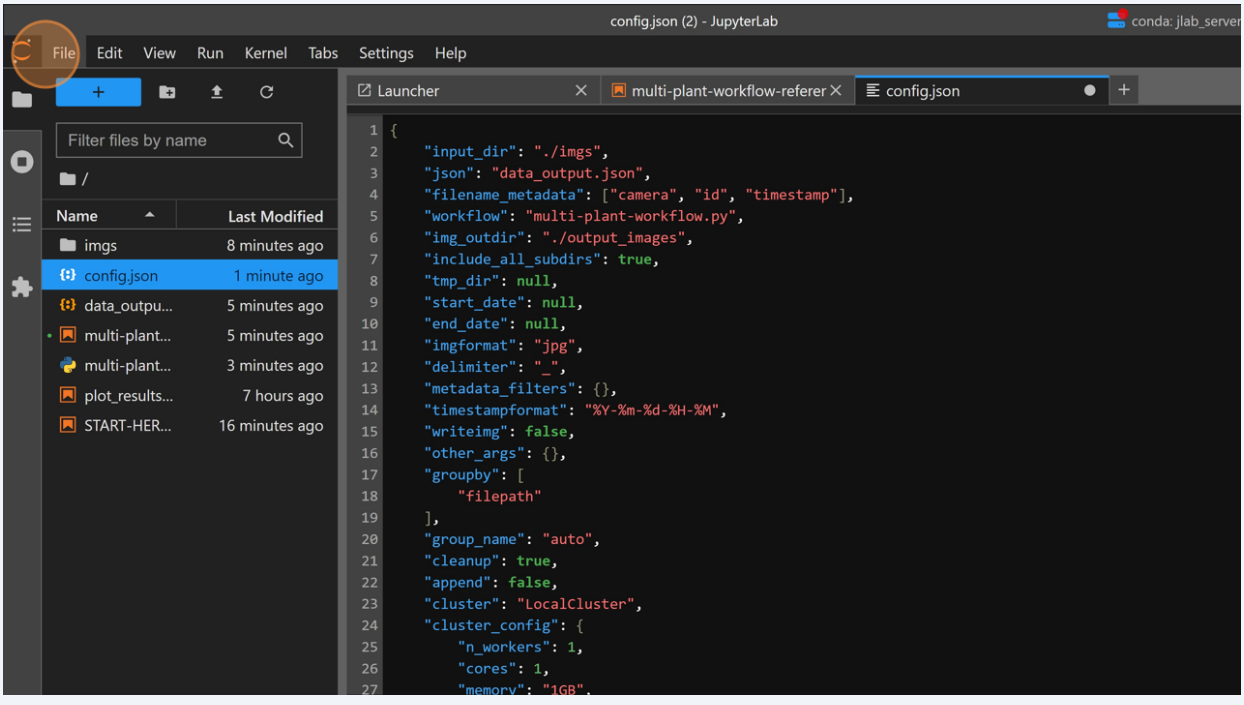
- "input_dir": "./imgs" *[Put file path/name of input directory for images you want analyzed]*
- "json": "data_output.json" *[Put the path/name of the data output file (located in the args container under results within your workflow)]*
- "filename_metadata": ["camera", "id", "timestamp"] *[list of metadata terms to collect. Supported metadata terms include: camera, imgtype, zoom, exposure, gain, frame, lifter, timestamp, id, plantbarcode, treatment, cartag, measurementlabel, and other]*
- "workflow": "multi-plant-workflow.py" *[path/name of user-defined (your) PlantCV workflow Python script]*
- "img_outdir": "./output_images" *[path/name of output directory where measured images will be stored. Default is "./output_images"]*
- "imgformat": "jpg" *[image file format/extension. Default is "png"]*
- "timestampformat": "%Y-%m-%d-%H-%M" *[date format as observed in your naming scheme. For explanation what each of the symbols mean, visit:* **https://docs.python.org/3.7/library/datetime.html#strftime-and-strptime-behavior** *]*
- "group_name": "auto" *[set this to auto]*
- "append": false *[(bool, default = True): if* **True** *will append results to an existing json file. If* **False***, will delete previous results stored in the specified JSON file.]*

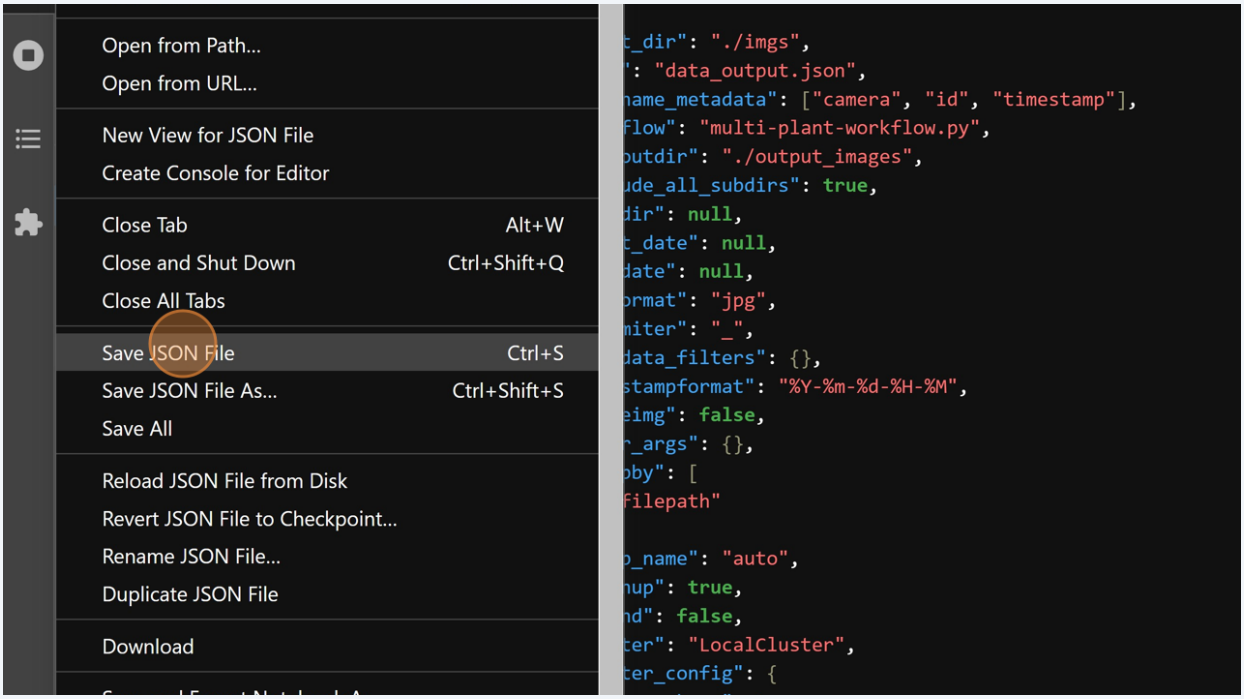
For more information on WorkflowConfig attributes, click here.

Launcher | multi-plant-workflow-referer | config.json

```
{
    "input_dir": "./imgs",
    "json": "data_output.json",
    "filename_metadata": ["camera", "id", "timestamp"],
    "workflow": "multi-plant-workflow.py",
    "img_outdir": "./output_images",
    "include_all_subdirs": true,
    "tmp_dir": null,
    "start_date": null,
    "end_date": null,
    "imgformat": "jpg",
    "delimiter": "_",
    "metadata_filters": {},
    "timestampformat": "%Y-%m-%d-%H-%M",
    "writeimg": false,
    "other_args": {},
    "groupby": [
        "filepath"
    ],
    "group_name": "auto",
    "cleanup": true,
    "append": false,
    "cluster": "LocalCluster",
    "cluster_config": {
        "n_workers": 1,
        "cores": 1,
```

12 Save the changes you have made to the **config.json**





13 Now that we have made the necessary changes to our parallel configuration file, it is time for us to run our workflow. To execute your parallel analysis, return to your terminal and type **plantcv-run-workflow --config config.json** into the prompt.

```
(plantcv) C:\Users\pbhatt\Documents\Workshop\Multi-Plant-Parallelization-Tutorial>plantcv-run-workflow --config config.json
```

If you successfully set up your **config.json** then you should see a ## of files found and a progress bar on your screen with how long it will take to analyze your dataset. You will also see that your job list will include X workflows.

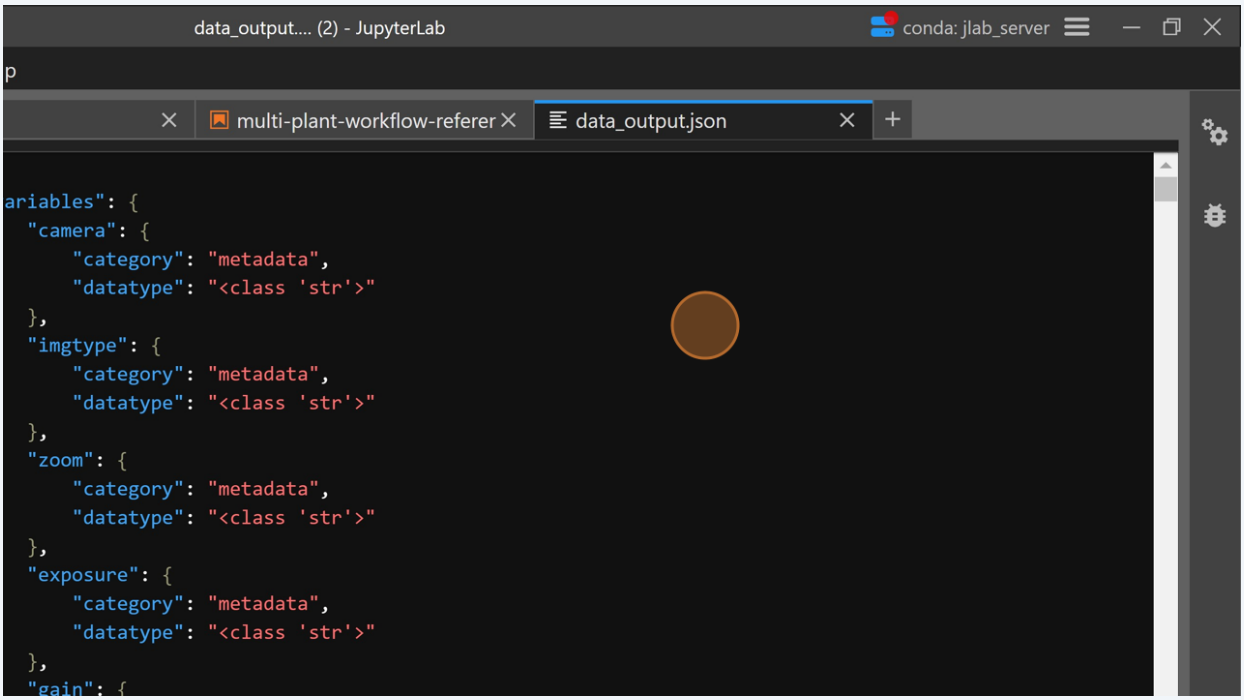
If you did not set up your **config.json** then you will receive error messages that detail where PlantCV is having issues finding an image directory, your workflow, incorrect date formats, etc.

```
(plantcv) C:\Users\pbhatt\Documents\Workshop\Multi-Plant-Parallelization-Tutorial>plantcv-run-workflow --config config.json
Starting run 2024-05-07_05-22-07

Reading image metadata...
Reading image metadata took 0.003996610641479492 seconds.
Building job list...
Task list includes 14 workflows
Building job list took 0.11158013343811035 seconds.
Processing images...
[                                        ] | 0% Completed | 19.2sWarning: Shrinking radius to make ROIs fit in the image
[#####################                   ] | 57% Completed |  1min 30.3sWarning: Shrinking radius to make ROIs fit in the image
Processing images took 154.34734082221985 seconds.Completed |  2min 29.4s←[2K
Processing results...
Processing results took 0.3885014057159424 seconds.

(plantcv) C:\Users\pbhatt\Documents\Workshop\Multi-Plant-Parallelization-Tutorial>
```

14 Go back to your file explorer and open **data_output.json** in the Editor to see if the file was changed and contains data. If the image you produced looks similar to the one below, then your workflow was successful.



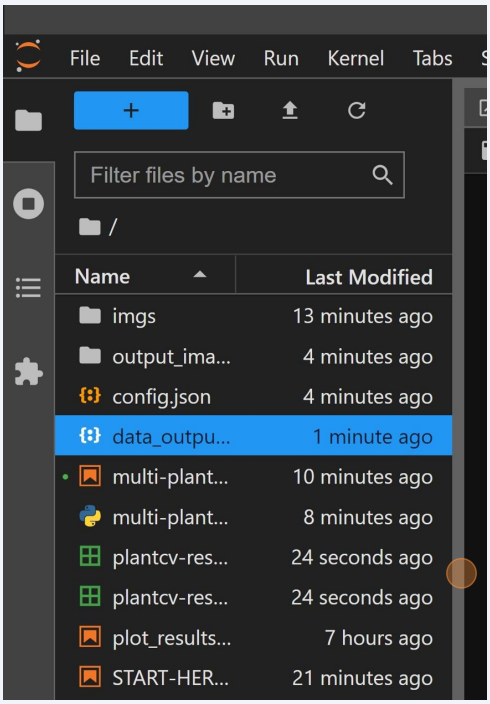
15 Once your analysis has been completed, you will notice a new output JSON file has appeared in your working directory. We will need to convert this JSON file to CSV in order to investigate our results further. To facilitate this conversation, submit the following line in the terminal: **plantcv-utils json2csv -j data_output.json -c plantcv-results**

```
(plantcv) C:\Users\pbhatt\Documents\Workshop\Multi-Plant-Parallelization-Tutorial>plantcv-utils json2csv -j data_output.json -c plantcv-results
```

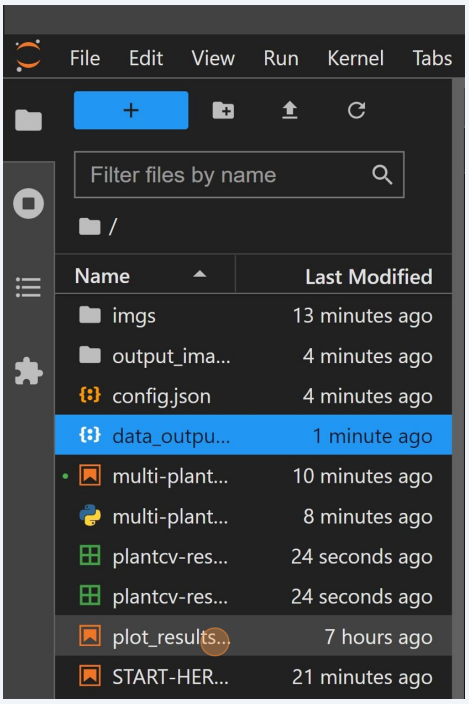
16 You should see two CSV files appear in your directory:

- **plantcv-results-single-value-traits.csv**
- **plantcv-results-multi-value-traits.csv**

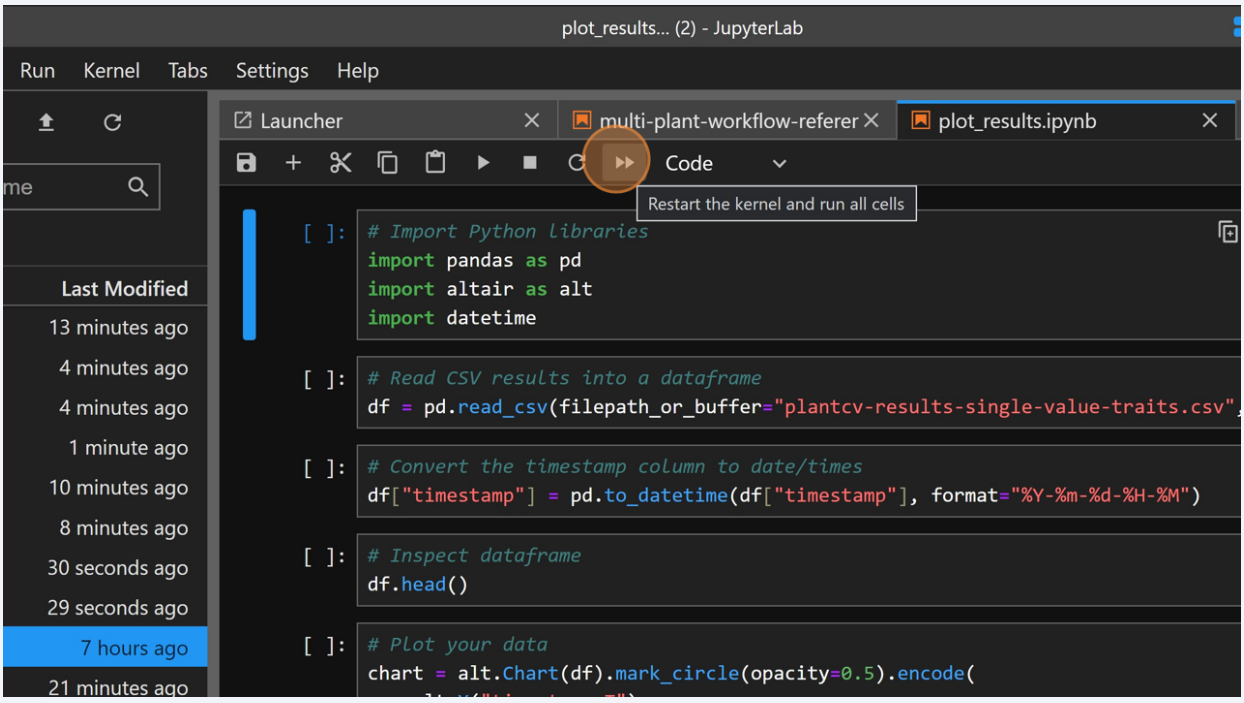
The **single-value-traits.csv** file will be in wide format, with a column per trait, whereas the **multi-value-traits.csv** file will be in long format, with one row per value/label. The hierarchical organization of these files enable more efficient data processing downstream.

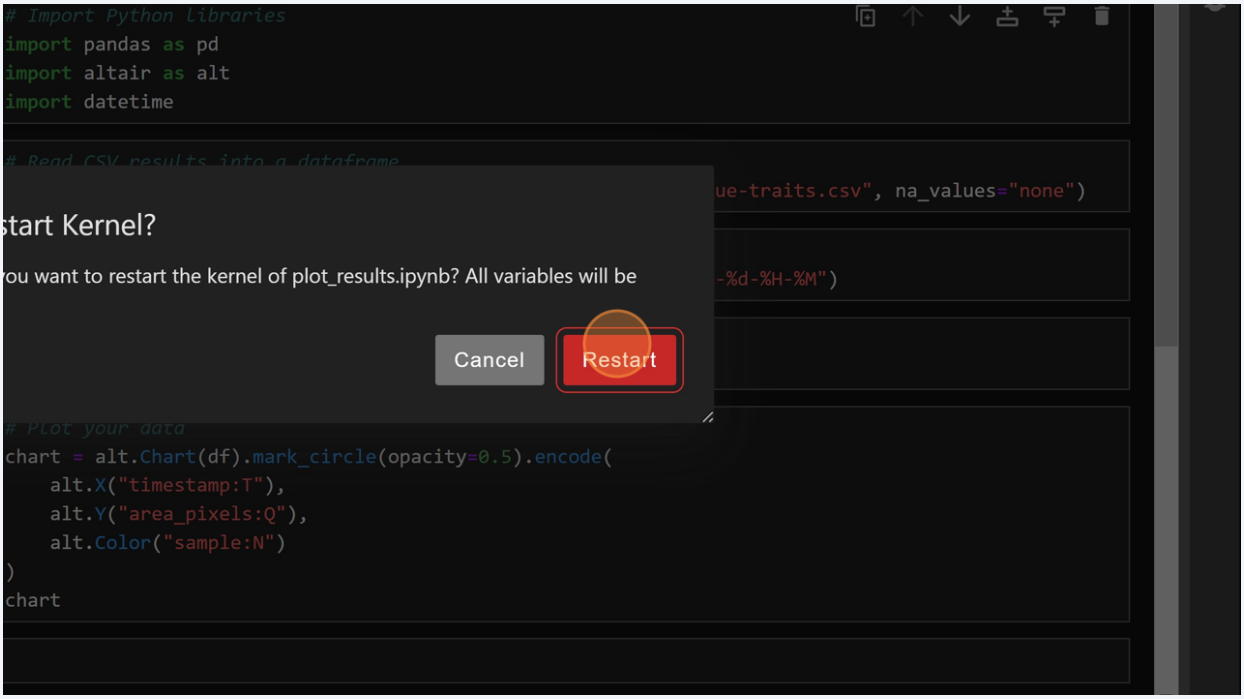


17 Next, we will inspect the data we just extracted. Double-click on **plot_results.ipynb** to open a Jupyter notebook that will allow us to visualize our results.



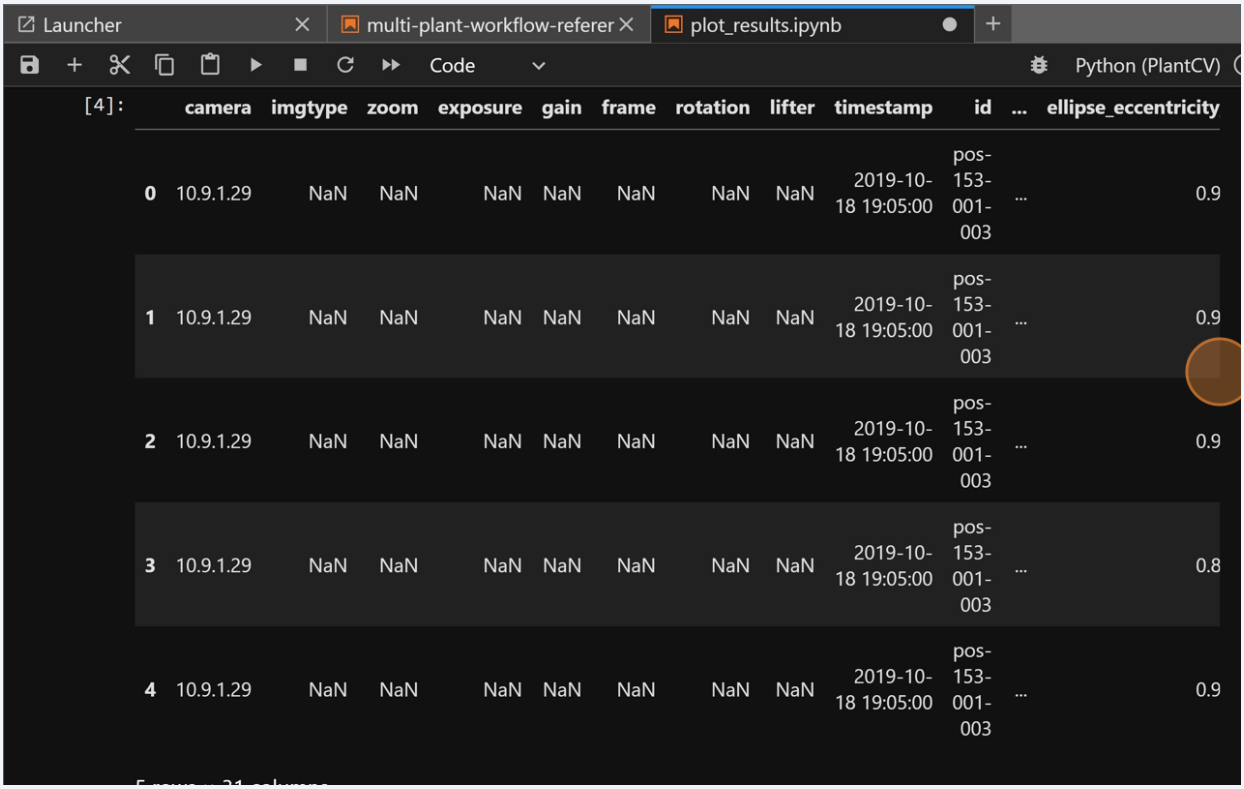
18 Click on the button highlighted below to restart the kernel and run all of the cells for the notebook. You will be prompted to restart the kernel, click **Restart**.



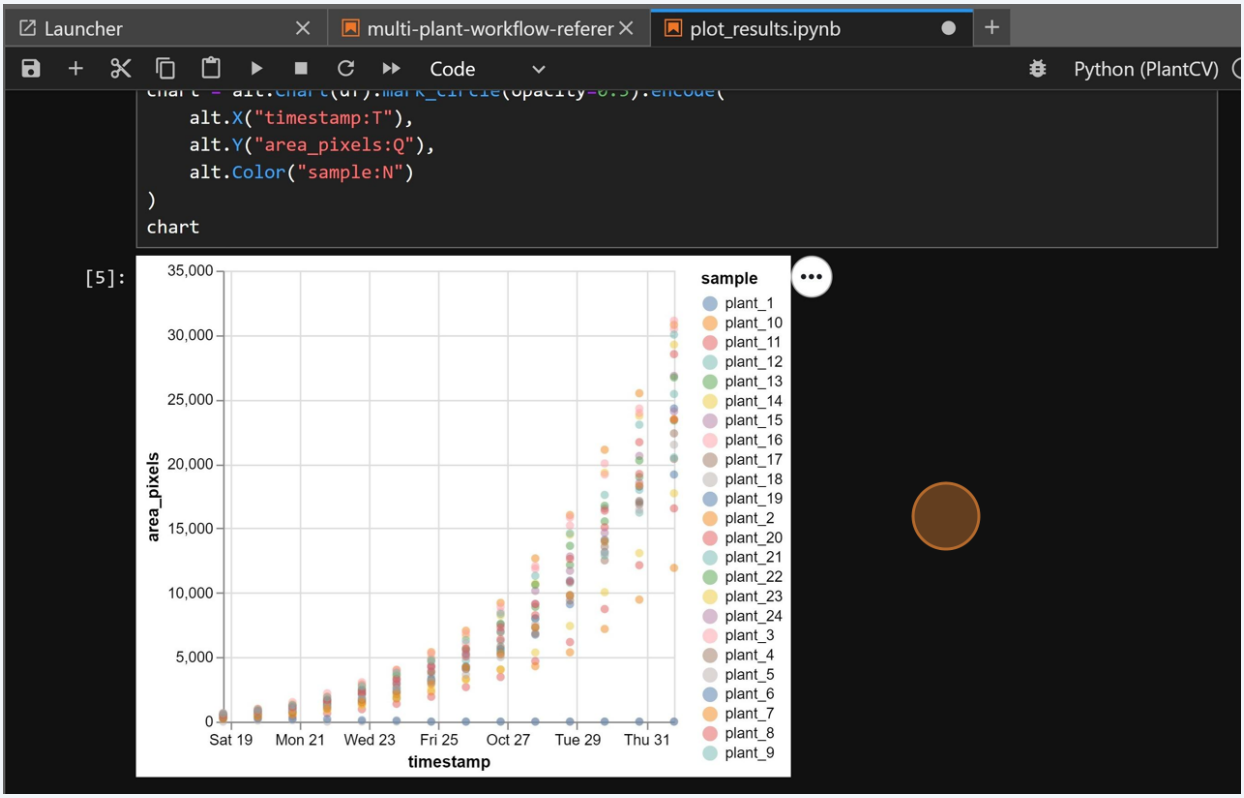


19 Investigate the tables yielded in the output.

What does the data tell you about this dataset?



	camera	imgtype	zoom	exposure	gain	frame	rotation	lifter	timestamp	id	...	ellipse_eccentricity
0	10.9.1.29	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2019-10-18 19:05:00	pos-153-001-003	...	0.9
1	10.9.1.29	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2019-10-18 19:05:00	pos-153-001-003	...	0.9
2	10.9.1.29	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2019-10-18 19:05:00	pos-153-001-003	...	0.9
3	10.9.1.29	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2019-10-18 19:05:00	pos-153-001-003	...	0.8
4	10.9.1.29	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2019-10-18 19:05:00	pos-153-001-003	...	0.9


Launcher
multi-plant-workflow-referer
plot_results.ipynb
Code
Python (PlantCV)
alt.X("timestamp:T"),
alt.Y("area_pixels:Q"),
alt.Color("sample:N")
)
chart
[5]:
35,000
30,000
25,000
20,000
15,000
10,000
5,000
0
area_pixels
Sat 19
Mon 21
Wed 23
Fri 25
Oct 27
Tue 29
Thu 31
timestamp
sample
plant_1
plant_10
plant_11
plant_12
plant_13
plant_14
plant_15
plant_16
plant_17
plant_18
plant_19
plant_2
plant_20
plant_21
plant_22
plant_23
plant_24
plant_3
plant_4
plant_5
plant_6
plant_7
plant_8
plant_9