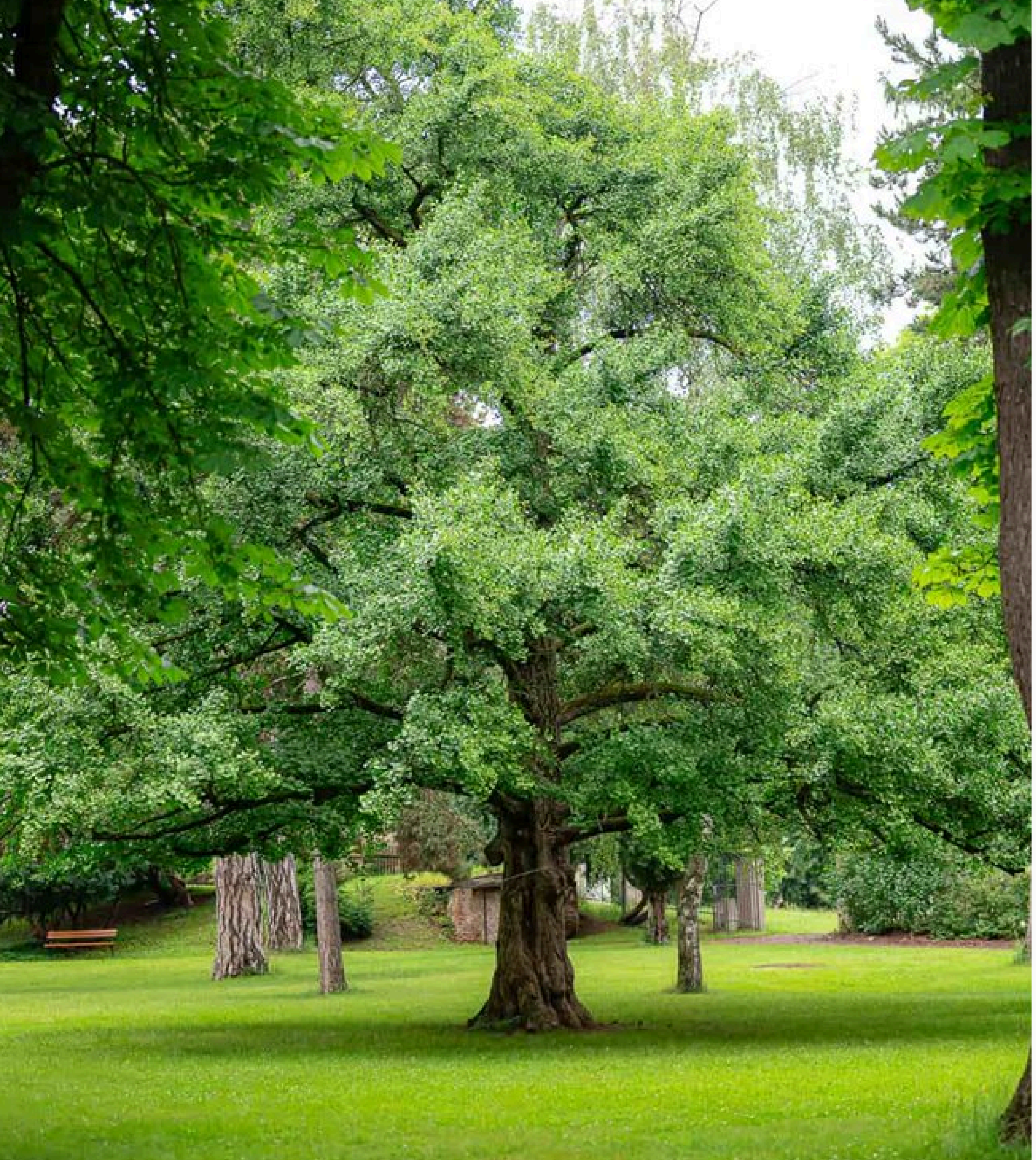
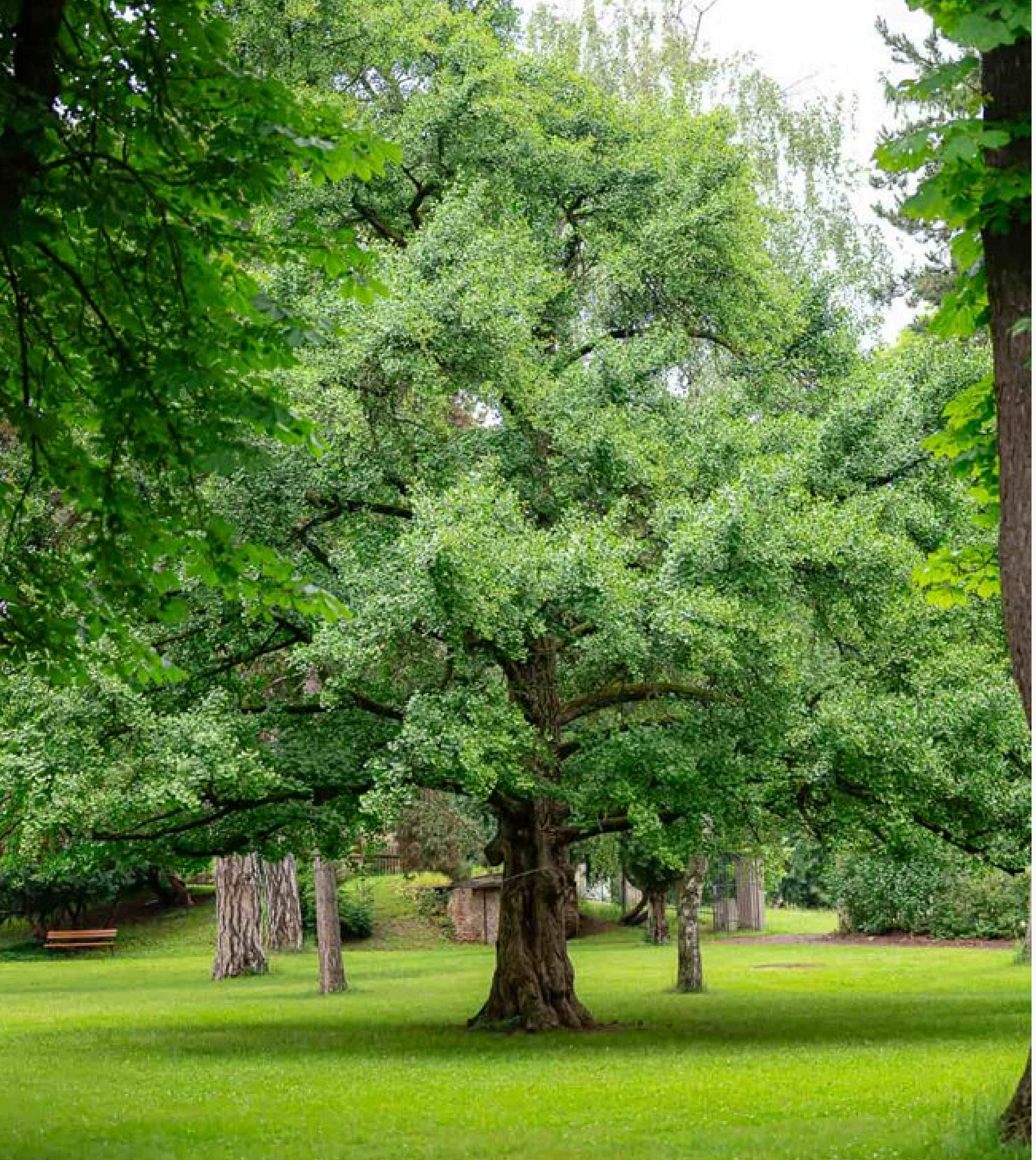


When acoustic tomography meets resistograph

Robert Mařík & Valentino Cristini

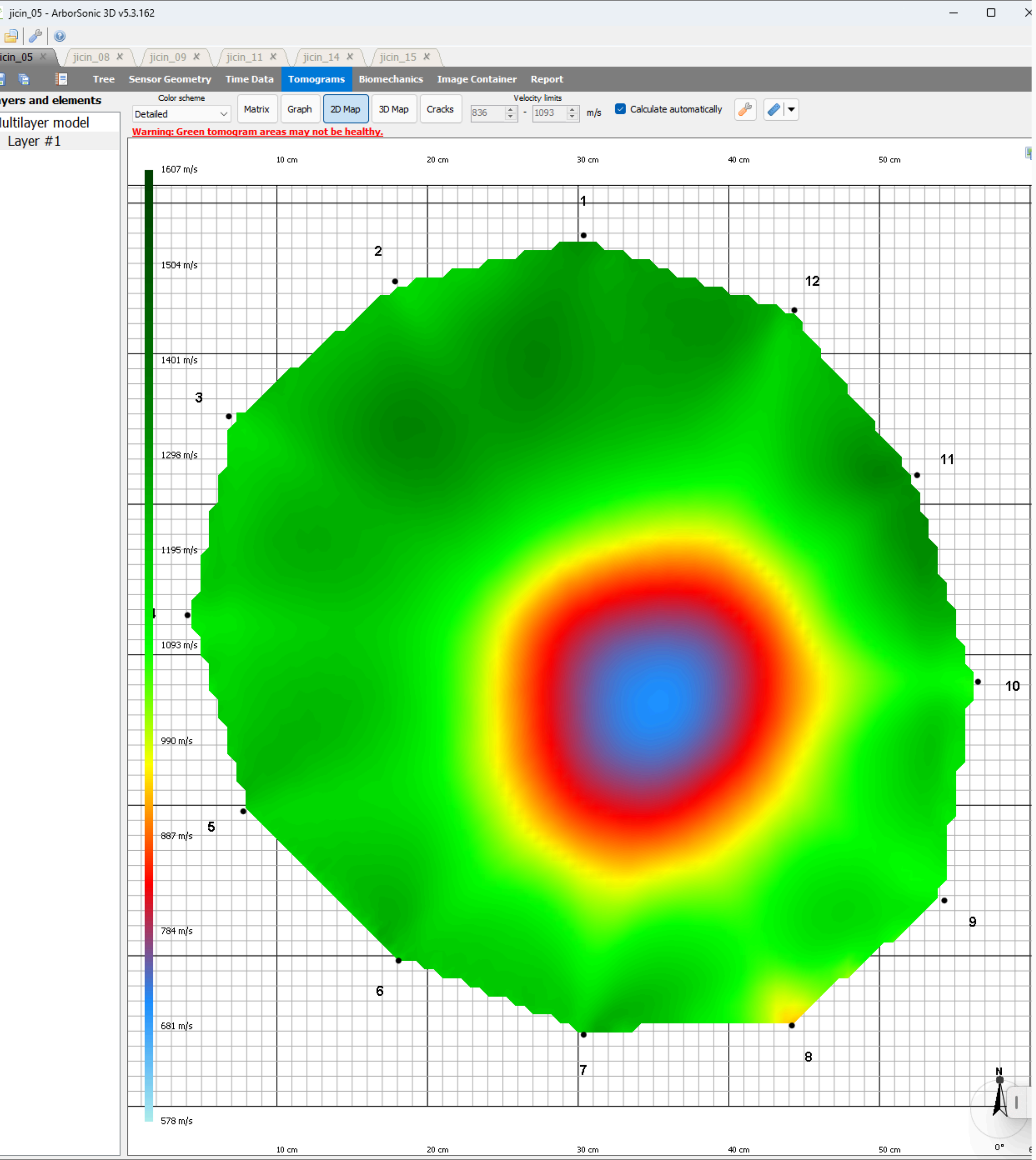
Mendel University in Brno

MENDELU
Faculty of Forestry
and Wood
Technology



Content of the talk

- Tomograph and resistograph: strengths and limitations
- Combined approach: a Python library for simultaneous interpretation
- Vibe coding (ChatGPT)
- Code containerization (Docker)



Acoustic tomograph

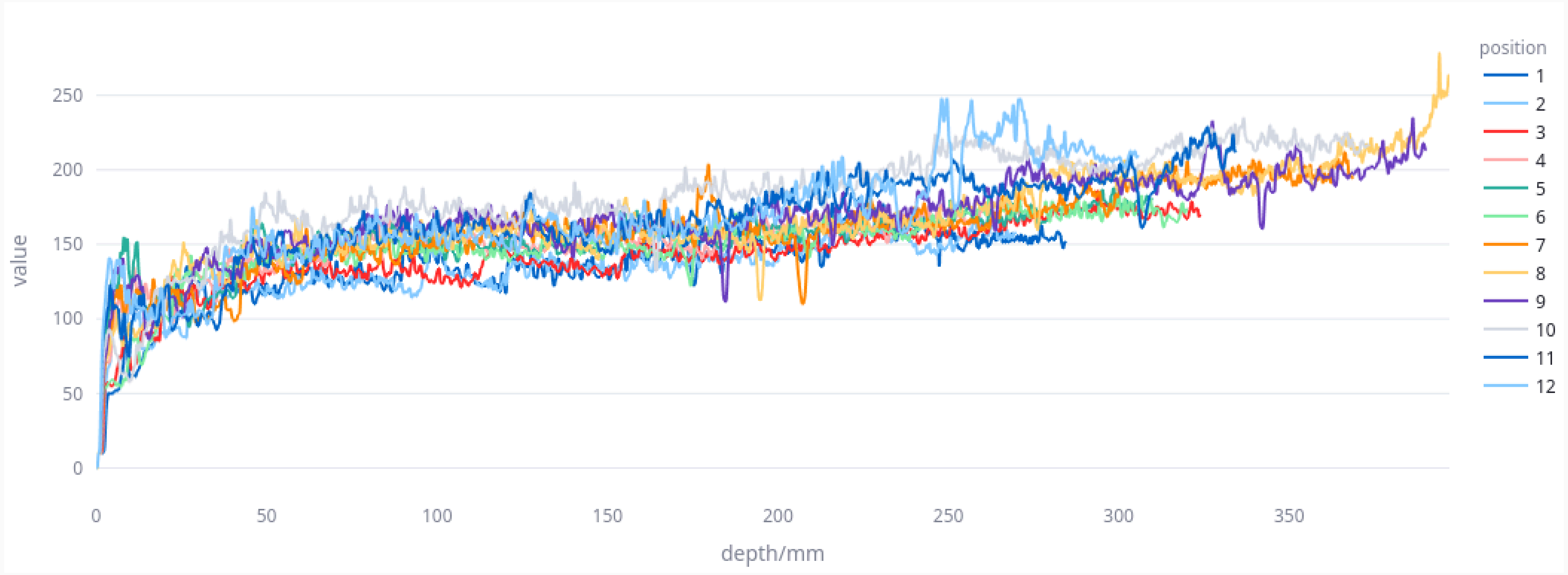
- Tool for fast stem inspection
- Provides **global information** across the whole cross section
- Green part - high sound speed value - sound wood
- Shows size and shape of internal defects

- Limited by long wavelength and small number of rays for reconstruction
- Cracks are reported as cavities

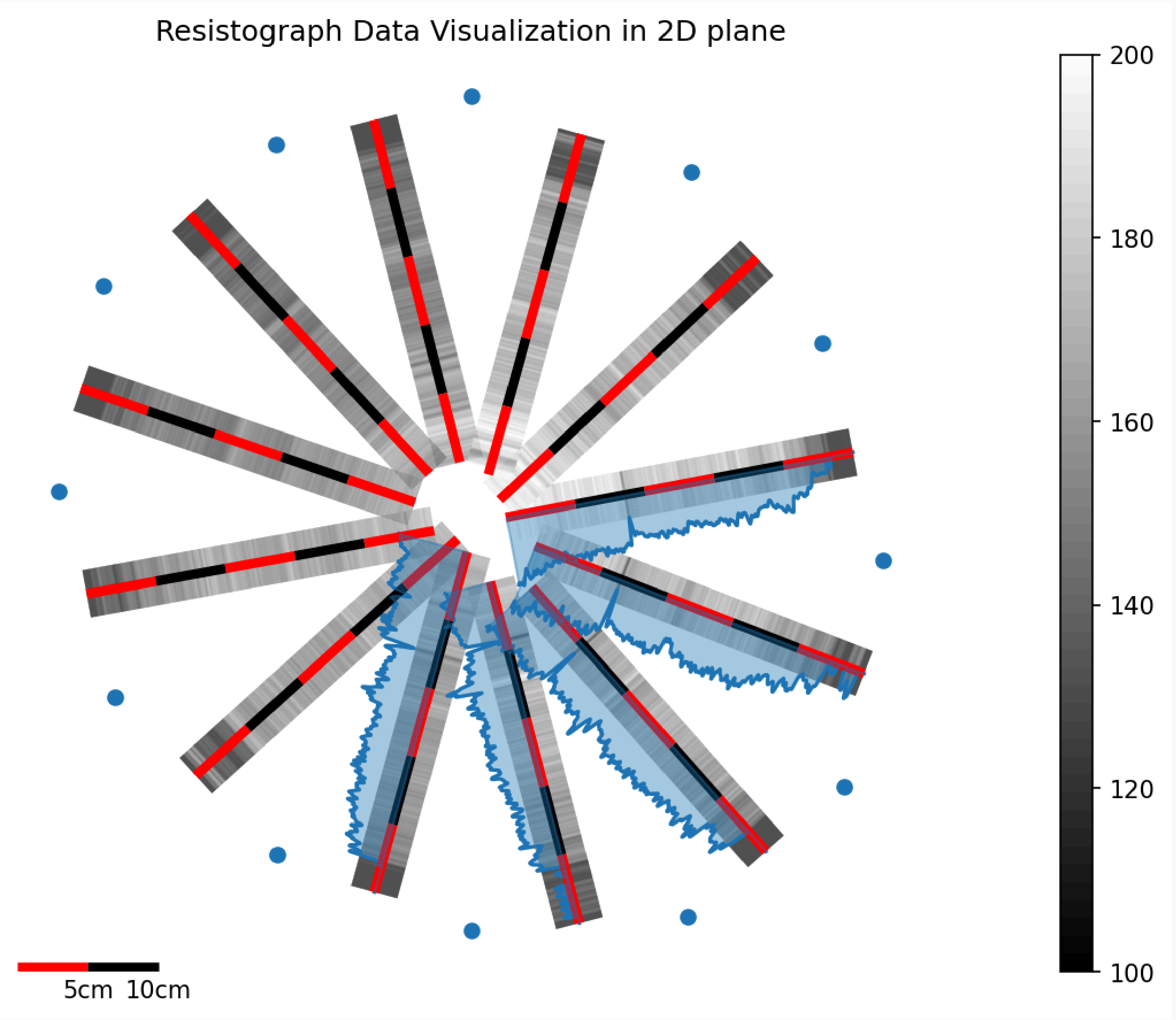
MENDELU
Faculty of Forestry
and Wood
Technology

Resistograph

- Measures the power required for microdrilling at a fixed speed
- Provides **local** mechanical properties of the material
- Note short(!) valleys: the central cavity hypotheses is not accepted
- Projection of the data to the stem cross section would reveal details of the defect



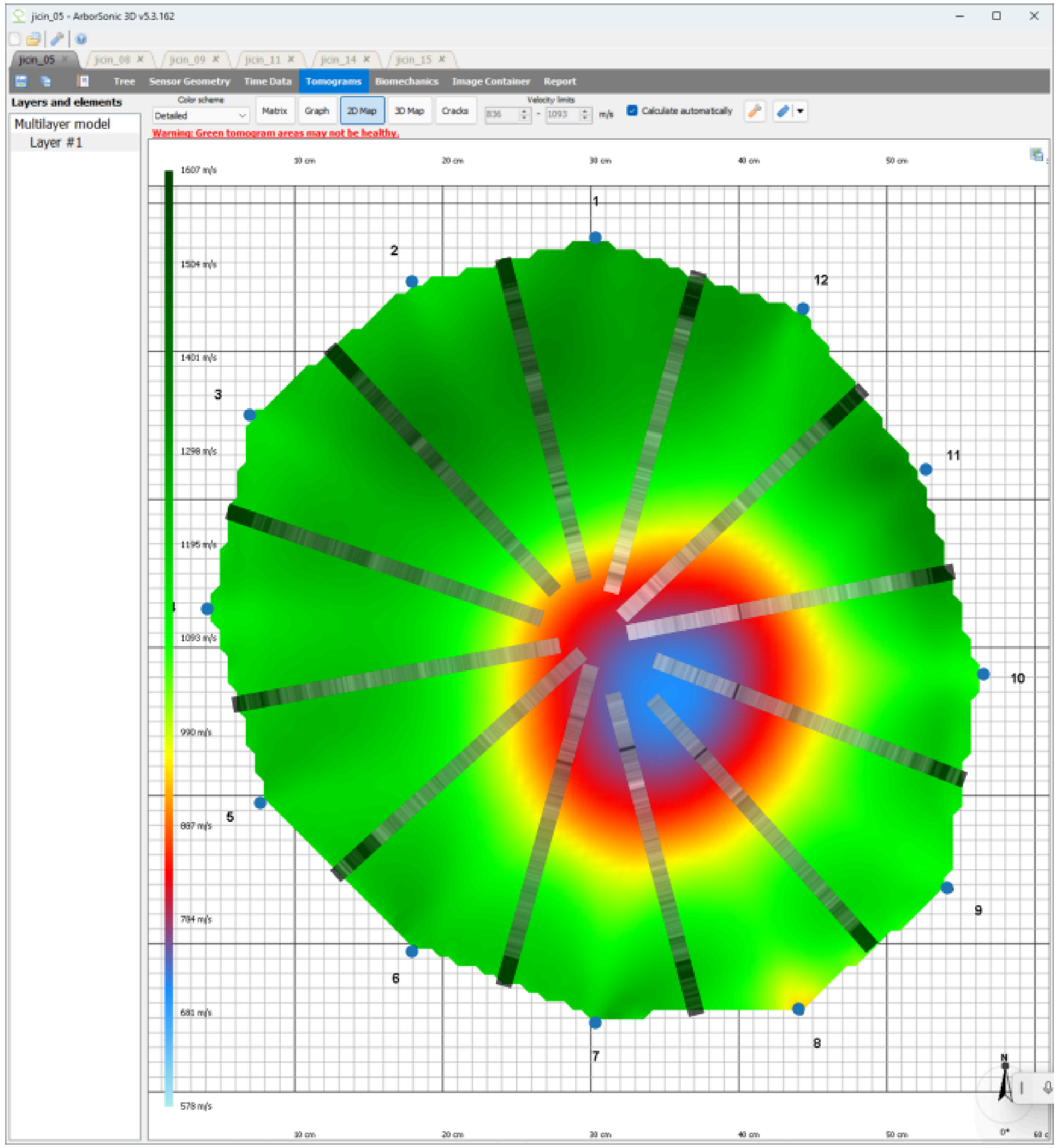
MENDELU
Faculty of Forestry and Wood Technology



Data in cross section geometry

- Transform resistograph data into 2D cross-section geometry
- Two types of visualization
  - curves
  - color scale
- Written as Python library
- Published on GitHub

MENDELU
Faculty of Forestry and Wood Technology



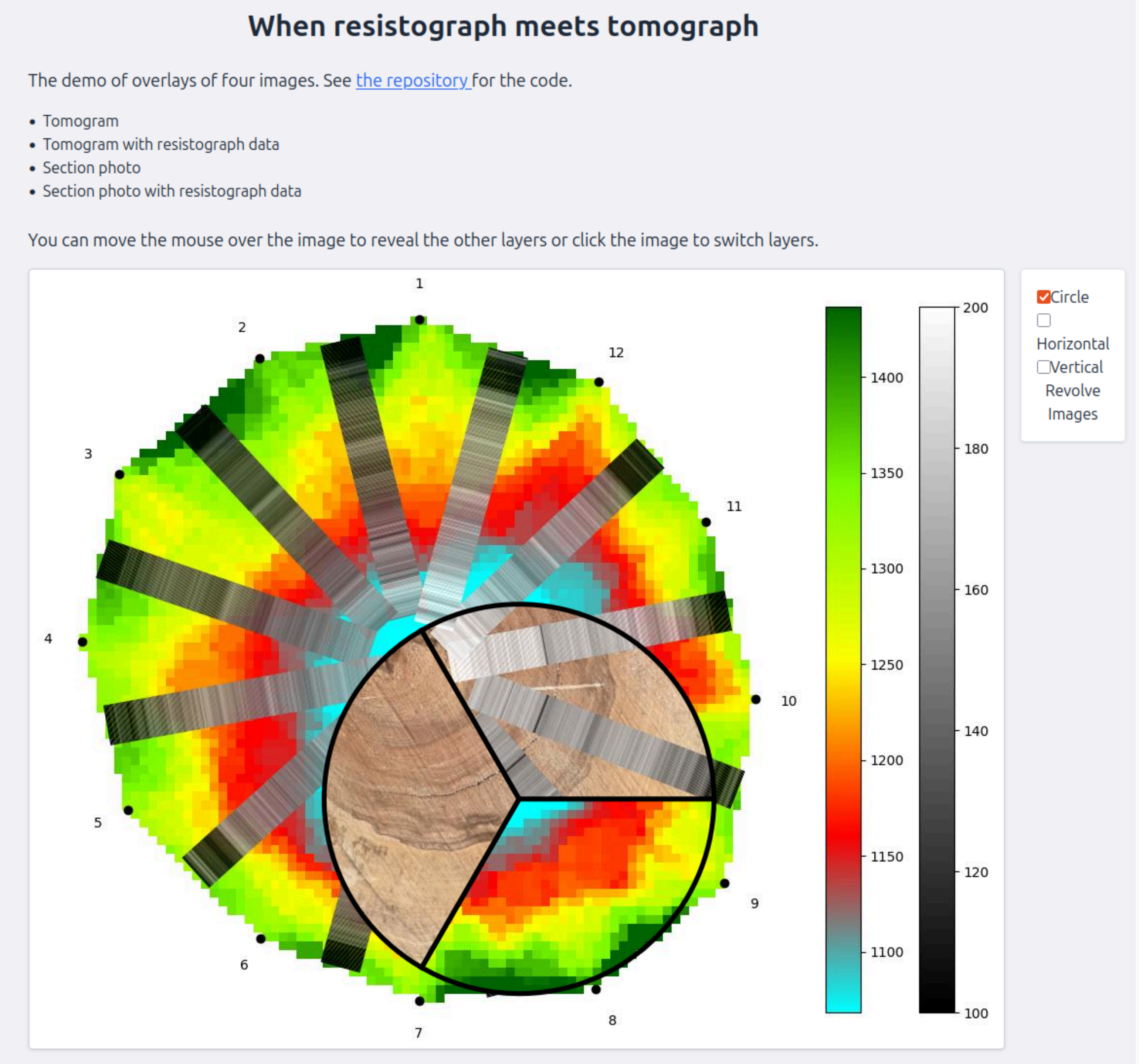
Color scheme: Detailed
Velocity limits: 836 - 1093 m/s
☑ Calculate automatically

Warning: Green tomogram areas may not be healthy.

Merge data

- Resistograph data in a tomogram
- Increases the accuracy of data interpretation
- The dark strips allow to localize the defect
- In our case the hypotheses have been confirmed by detailed inspection after felling the tree

MENDELU
Faculty of Forestry and Wood Technology



When resistograph meets tomograph

The demo of overlays of four images. See the repository for the code.

- Tomogram
- Tomogram with resistograph data
- Section photo
- Section photo with resistograph data

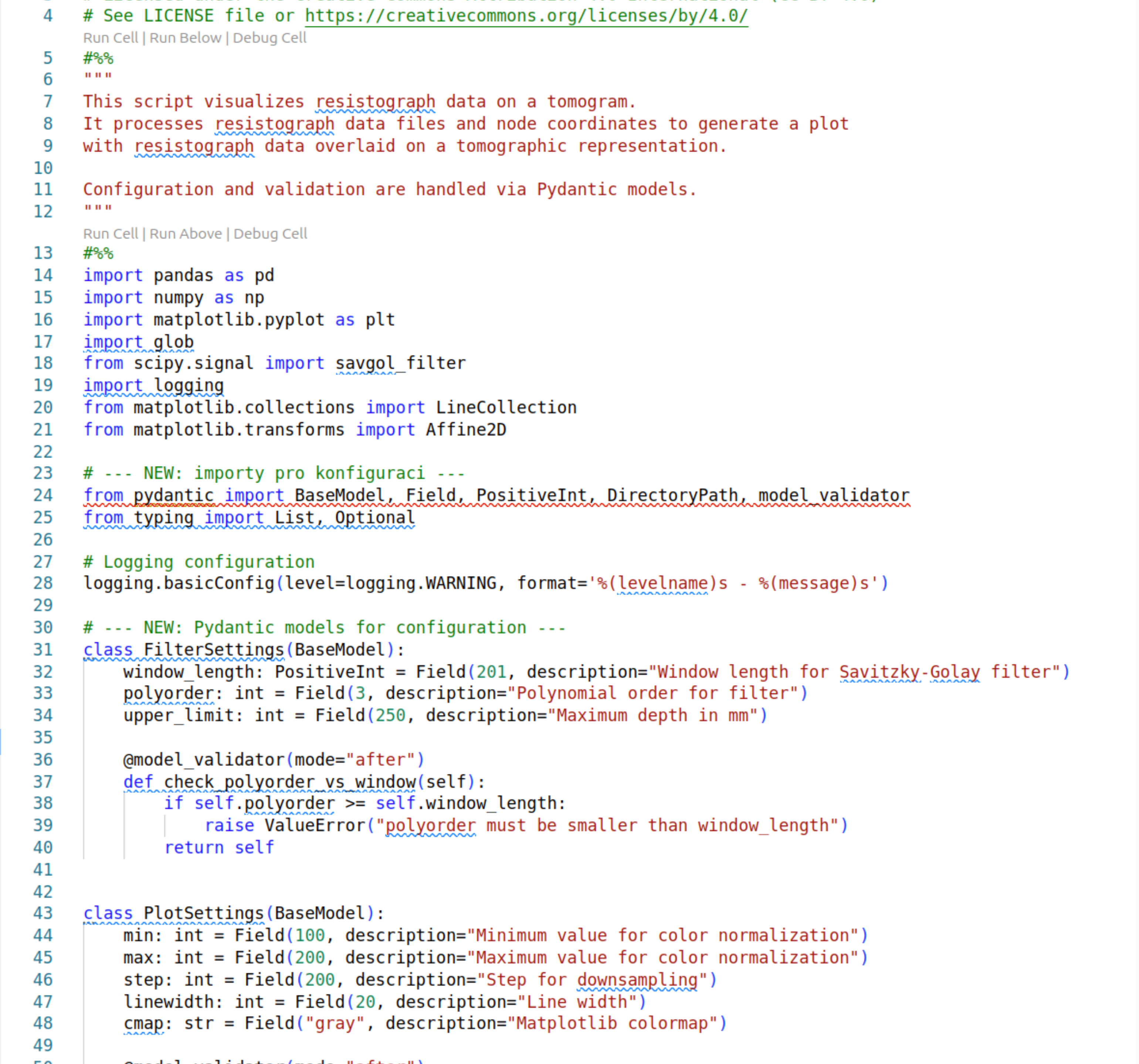
You can move the mouse over the image to reveal the other layers or click the image to switch layers.

Python library

Advantages

- Python is widely used in scientific data processing
- Easy automation, scaling, modification, sharing, and reuse
- Simple integration with other tools

MENDELU
Faculty of Forestry and Wood Technology


Run Cell | Run Below | Debug Cell
Run Cell | Run Above | Debug Cell

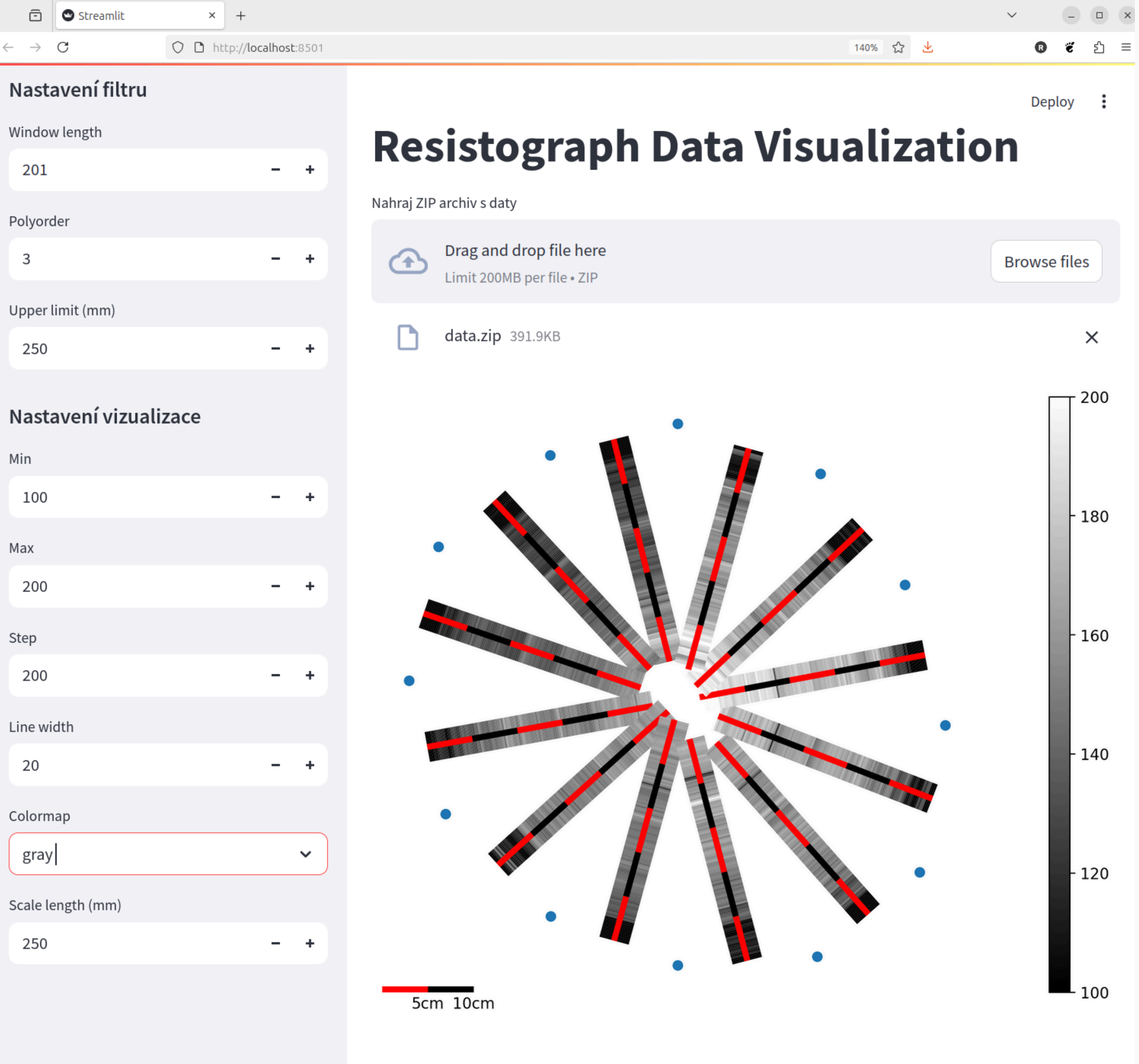
```python
# See LICENSE file or https://creativecommons.org/licenses/by/4.0/
#%%
"""
This script visualizes resistograph data on a tomogram.
It processes resistograph data files and node coordinates to generate a plot
with resistograph data overlaid on a tomographic representation.

Configuration and validation are handled via Pydantic models.
"""
#%%
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import glob
from scipy.signal import savgol_filter
import logging
from matplotlib.collections import LineCollection
from matplotlib.transforms import Affine2D

# --- NEW: importy pro konfiguraci ---
from pydantic import BaseModel, Field, PositiveInt, DirectoryPath, model_validator
from typing import List, Optional

# Logging configuration
logging.basicConfig(level=logging.WARNING, format='%(levelname)s - %(message)s')

# --- NEW: Pydantic models for configuration ---
class FilterSettings(BaseModel):
    window_length: PositiveInt = Field(201, description="Window length for Savitzky-Golay filter")
    polyorder: int = Field(3, description="Polynomial order for filter")
    upper_limit: int = Field(250, description="Maximum depth in mm")

    @model_validator(mode="after")
    def check_polyorder_vs_window(self):
        if self.polyorder >= self.window_length:
            raise ValueError("polyorder must be smaller than window_length")
        return self


class PlotSettings(BaseModel):
    min: int = Field(100, description="Minimum value for color normalization")
    max: int = Field(200, description="Maximum value for color normalization")
    step: int = Field(200, description="Step for downsampling")
    linewidth: int = Field(20, description="Line width")
    cmap: str = Field("gray", description="Matplotlib colormap")
```

Python library

Limitations

- Needs installation of Python ecosystem
- Requires programming skills
- No graphical user interface

MENDELU
Faculty of Forestry and Wood Technology



Nastavení filtru

Window length 201

Polyorder 3

Upper limit (mm) 250

Nastavení vizualizace

Min 100

Max 200

Step 200

Line width 20

Colormap gray

Scale length (mm) 250

Resistograph Data Visualization

Nahraj ZIP archiv s daty

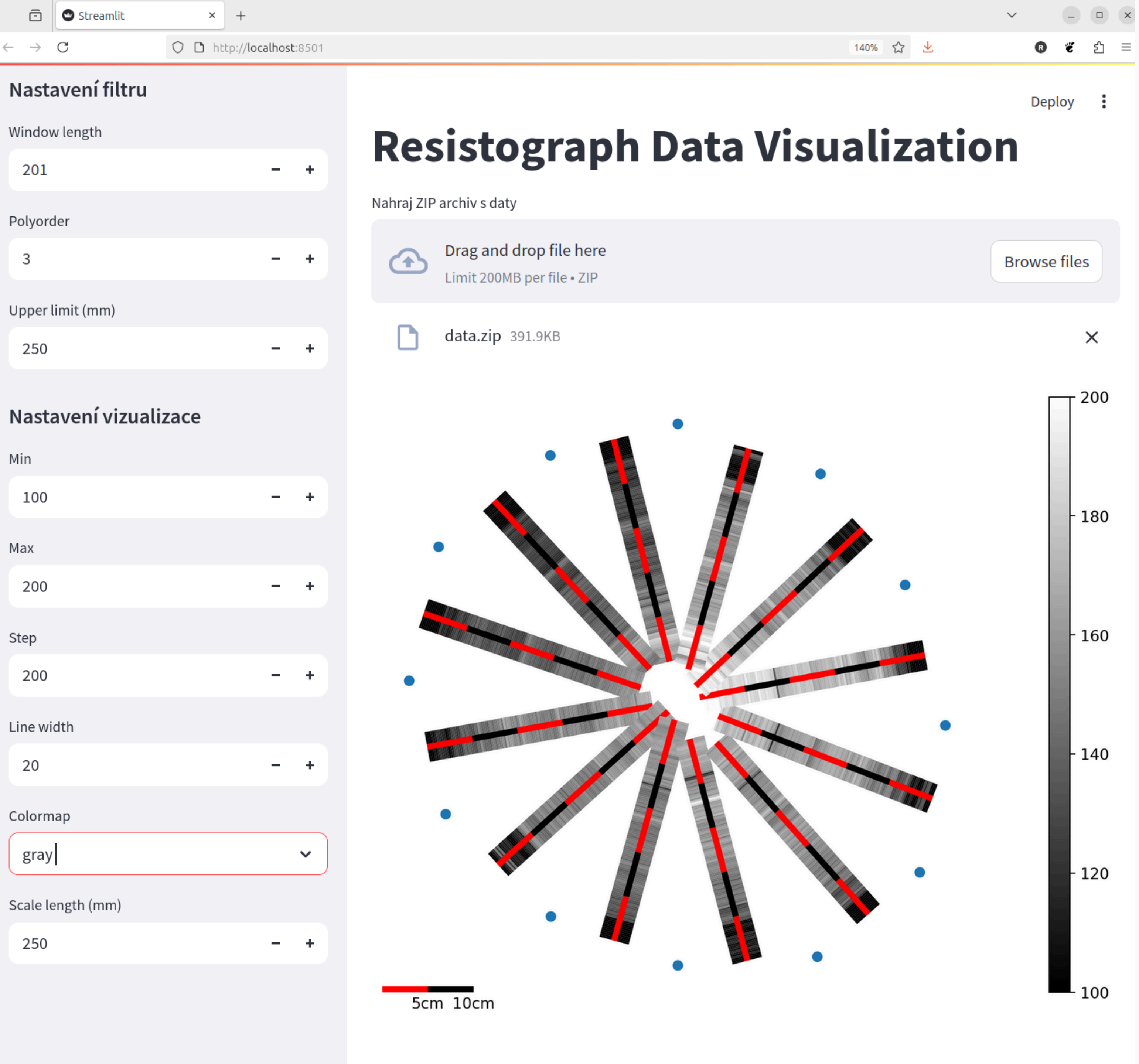
Drag and drop file here
Limit 200MB per file • ZIP

data.zip 391.9KB

Streamlit

- Library for building interactive web apps
- GUI in web browser
- Widely used in industry and academia
- Requires minimal code

MENDELU
Faculty of Forestry and Wood Technology



Nastavení filtru

Window length: 201

Polyorder: 3

Upper limit (mm): 250

Nastavení vizualizace

Min: 100

Max: 200

Step: 200

Line width: 20

Colormap: gray

Scale length (mm): 250

Resistograph Data Visualization

Nahraj ZIP archiv s daty

Drag and drop file here

Limit 200MB per file • ZIP

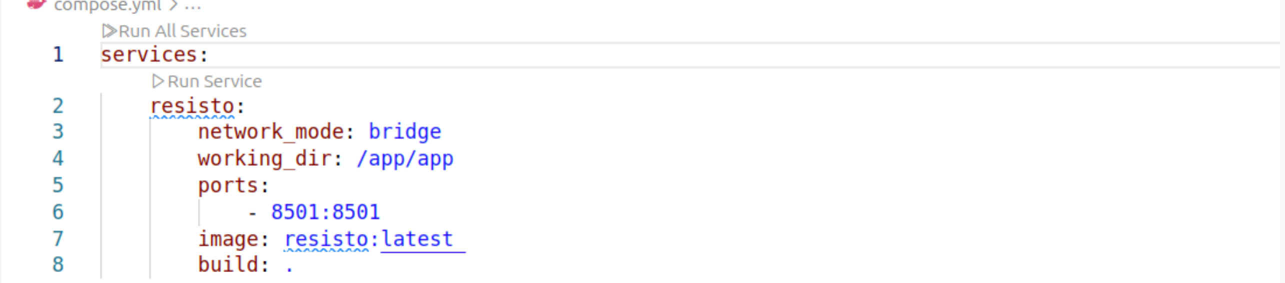
data.zip 391.9KB

Vibe coding

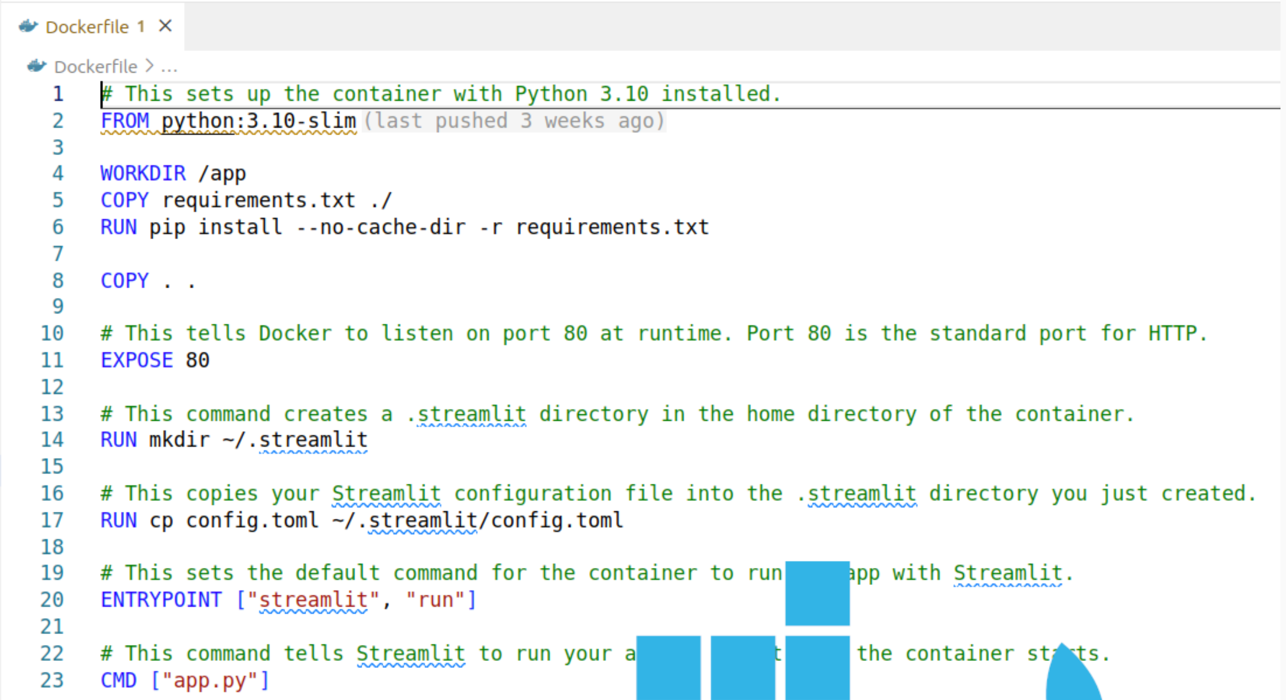
- Code written by AI (LLM)
- ChatGPT 5 in August 2025
- Web app produced in two prompts

```
I have the following library. Write
a streamlit program that allows you
to upload a zipped directory with data
and run commands corresponding to
the main function on it.
The output will be displayed.
```

```
OK. I want to be able to change
the preset options in the left panel.
```



```yaml
services:
    resisto:
        network_mode: bridge
        working_dir: /app/app
        ports:
            - 8501:8501
        image: resisto:latest
        build: .
```



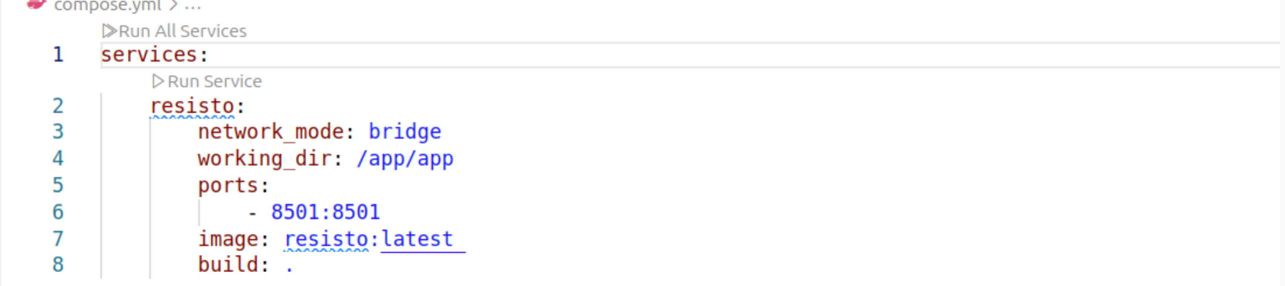
```dockerfile
# This sets up the container with Python 3.10 installed.
FROM python:3.10-slim (last pushed 3 weeks ago)

WORKDIR /app
COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

# This tells Docker to listen on port 80 at runtime. Port 80 is the standard port for HTTP.
EXPOSE 80

# This command creates a .streamlit directory in the home directory of the container.
RUN mkdir ~/.streamlit

# This copies your Streamlit configuration file into the .streamlit directory you just created.
RUN cp config.toml ~/.streamlit/config.toml

# This sets the default command for the container to run [illegible] app with Streamlit.
ENTRYPOINT ["streamlit", "run"]

# This command tells Streamlit to run your a[illegible]t[illegible] the container st[illegible]ts.
CMD ["app.py"]
```

Docker

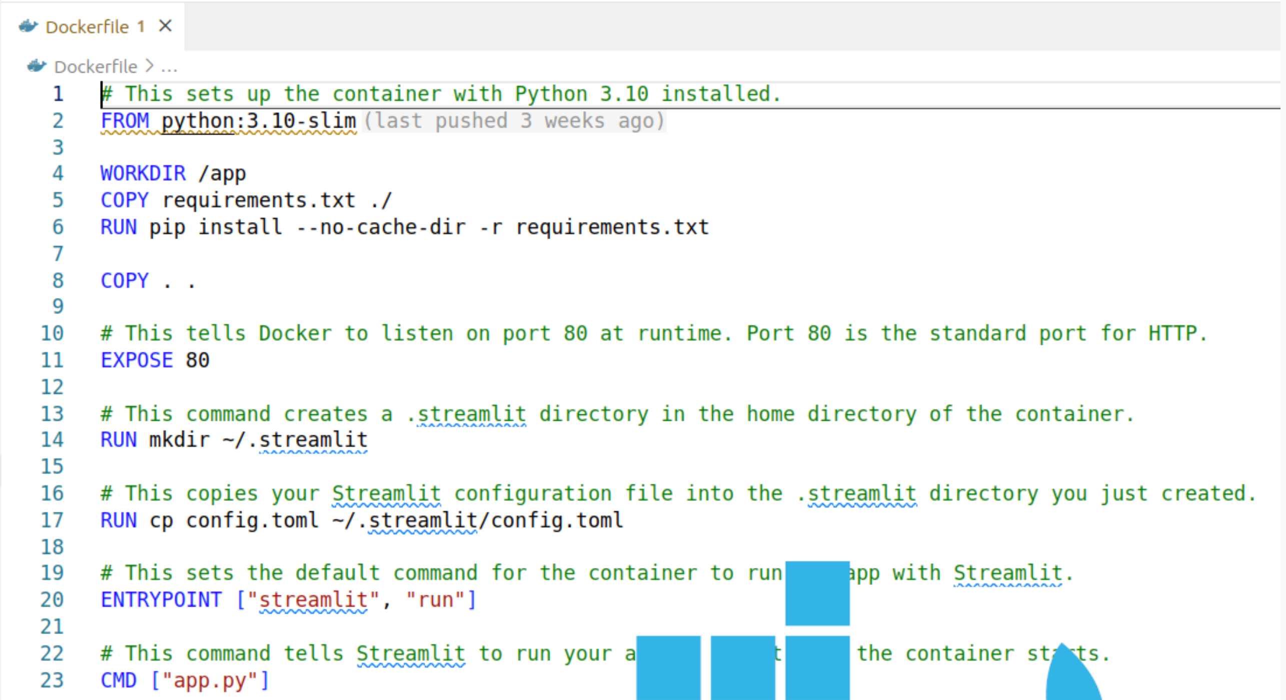
- Packages app and dependencies into a single container
- Ensures consistency across environments - ideal for **transparent and repeatable data processing**
- Widely used in industry and academia

- In some fields, the "compute capsule" is submitted together with the manuscript (Code Ocean).

MENDELU
Faculty of Forestry
and Wood
Technology



```yaml
services:
    resisto:
        network_mode: bridge
        working_dir: /app/app
        ports:
            - 8501:8501
        image: resisto:latest
        build: .
```

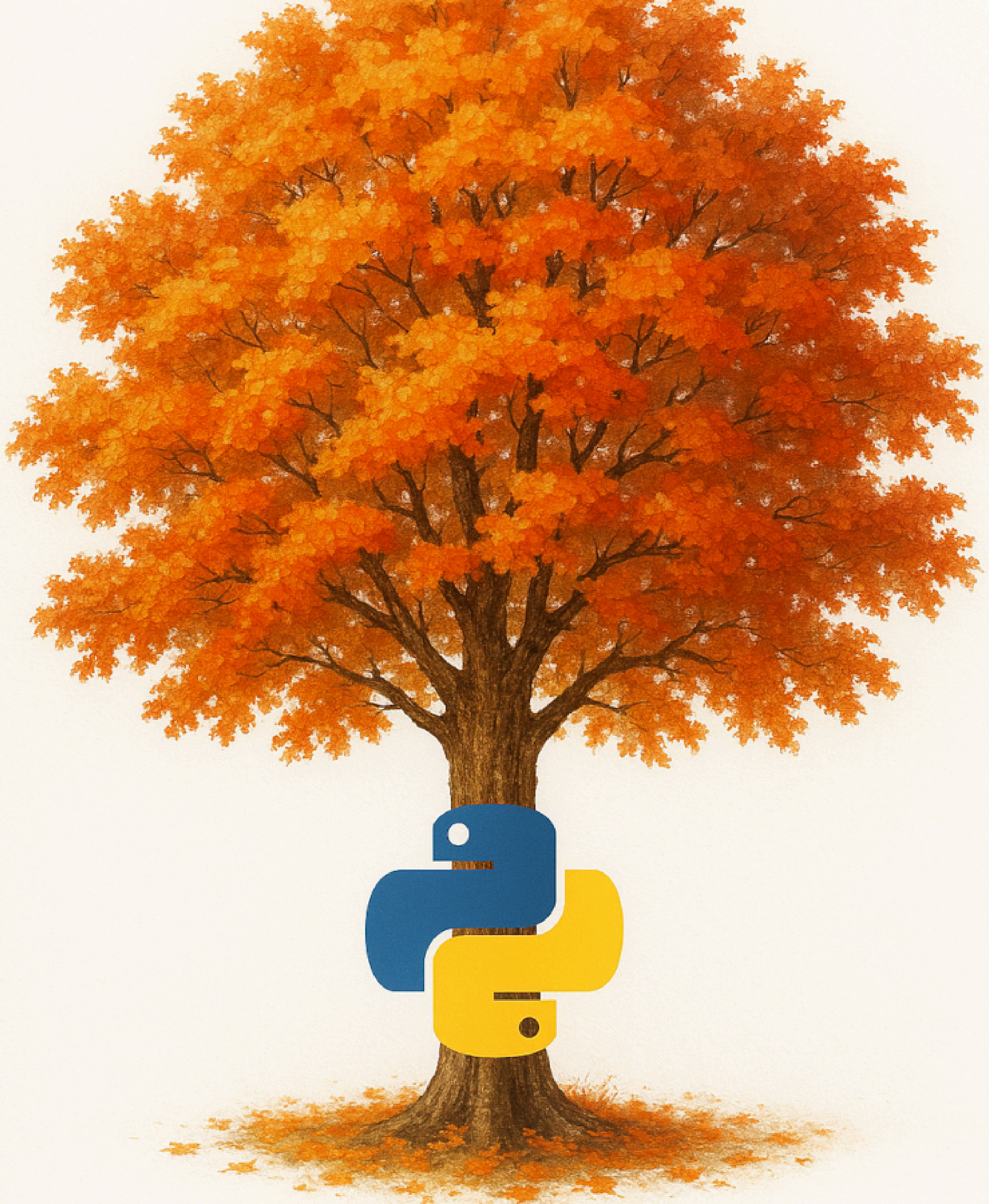


```dockerfile
# This sets up the container with Python 3.10 installed.
FROM python:3.10-slim (last pushed 3 weeks ago)

WORKDIR /app
COPY requirements.txt ./
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

# This tells Docker to listen on port 80 at runtime. Port 80 is the standard port for HTTP.
EXPOSE 80

# This command creates a .streamlit directory in the home directory of the container.
RUN mkdir ~/.streamlit

# This copies your Streamlit configuration file into the .streamlit directory you just created.
RUN cp config.toml ~/.streamlit/config.toml

# This sets the default command for the container to run [illegible]app with Streamlit.
ENTRYPOINT ["streamlit", "run"]

# This command tells Streamlit to run your a[illegible]t[illegible] the container st[illegible]ts.
CMD ["app.py"]
```

Run dockerized app

```
docker compose up
```

- First run takes minutes, later runs take ms
- No Python install required
- No dependency issues
- Works on Win / Mac / Linux



Summary

- Resistograph and tomograph are complementary tools for tree inspection

- A Python library was developed to enhance data interpretation
- GUI is possible with Streamlit
- Coding can be done with AI support
- Installation can be simplified and repeated with Docker

MENDELU
Faculty of Forestry and Wood Technology