Programming Project 03

Christine Robinson, Francèl Lamprecht, Regina Wehler

UNIVERSITÄT BONN

b-it Bonn-Aachen International Center for Information Technology

Plant Phenotyping

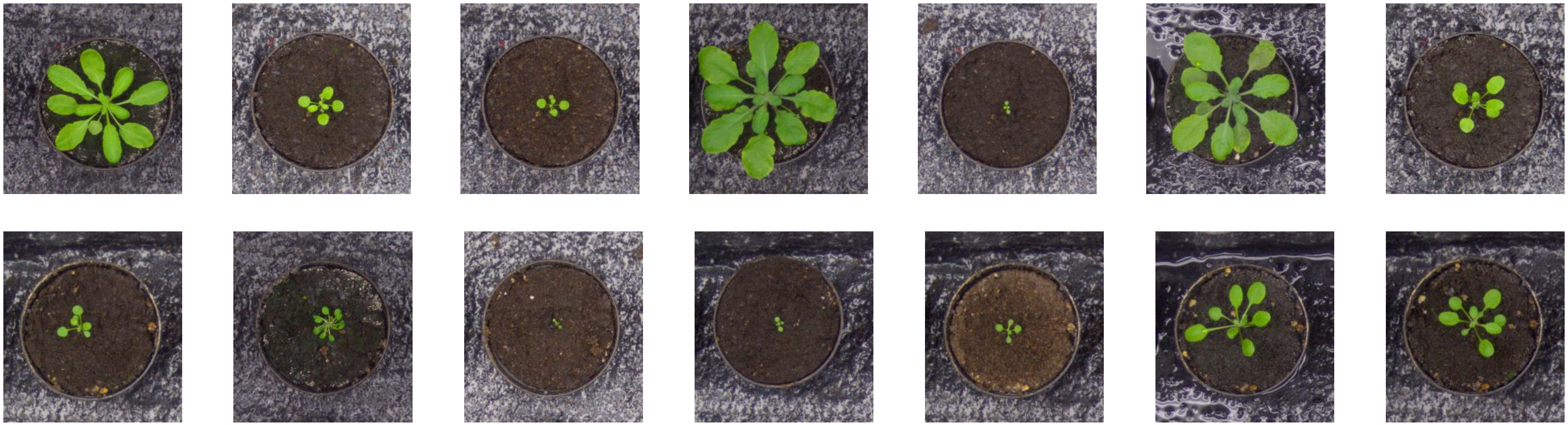
- Main bottleneck in plant research and breeding is the phenotyping capability
- Required: high-throughput phenotyping with non-invasive methods to screen root and shoot phenotypes
- Solution: robotic driven greenhouses
- Examples at the Forschungszentrum Jülich:

SCREEN House

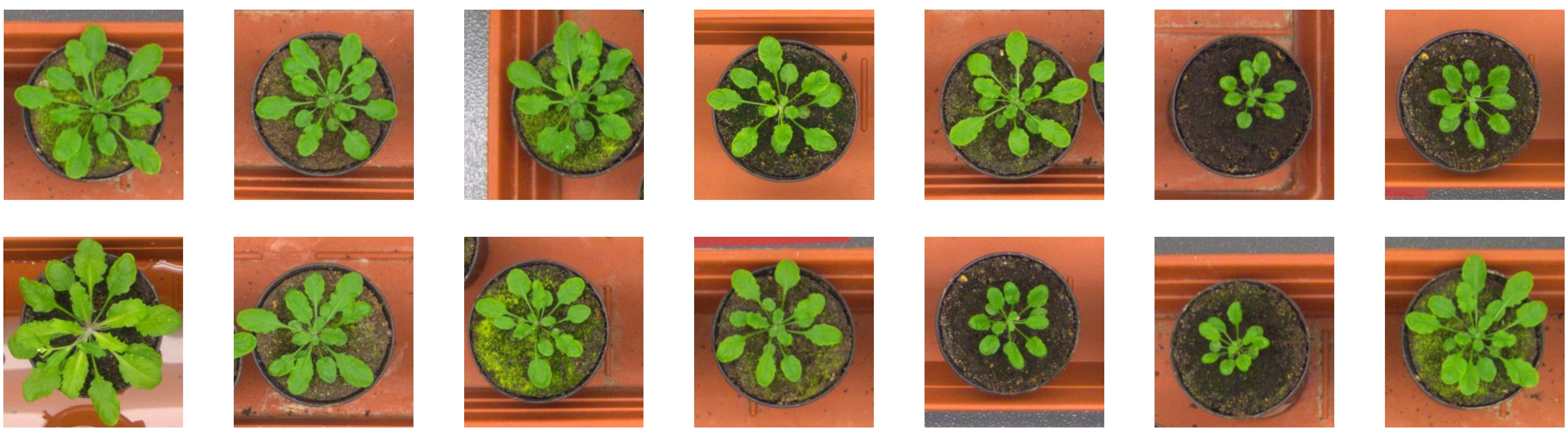
GROWSCREEN chamber

- Both greenhouses are equipped with robots that transport plants to imaging stations

Computer Science Part



➔ Automated analysis of automatically acquired plant images

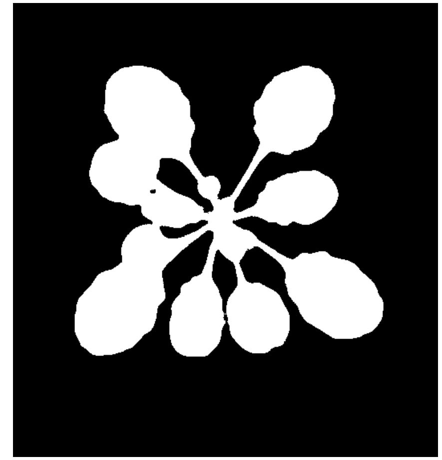


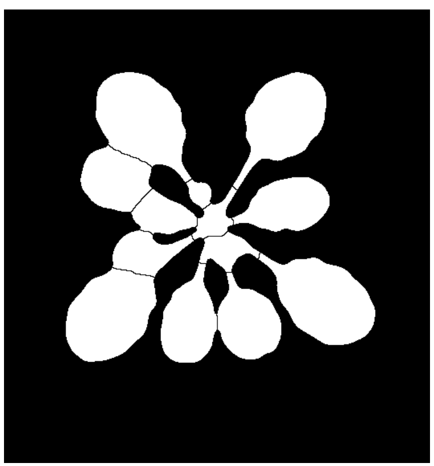
Goal of this Project

- Develop a simple approach for plant analysis from top view photos of seedlings
- Two main parts:
  - Image Analysis with Fiji in Java
  - Explorative Data Analysis with Python in Jupyter Notebooks
- Data Sets: training sets from the Leaf Segmentation and Counting Challenges of the International Plant Phenotyping Network which include wild type and mutant plants

Image Analysis Pipeline


1.) RGB input image


2.) Classification


3.) Leaf Identification

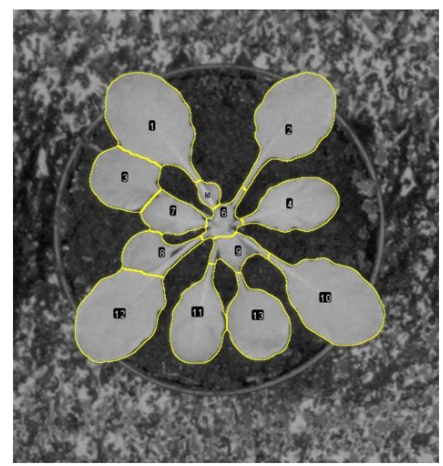
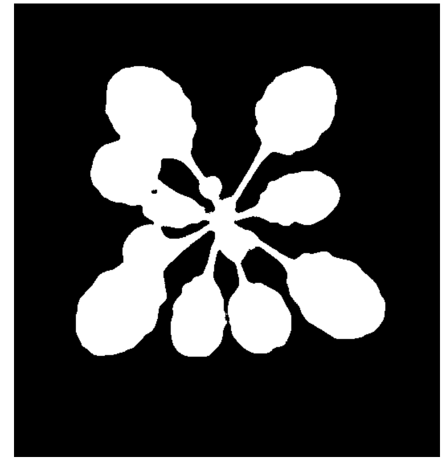

4.) Leaf Analysis

Image Analysis Pipeline

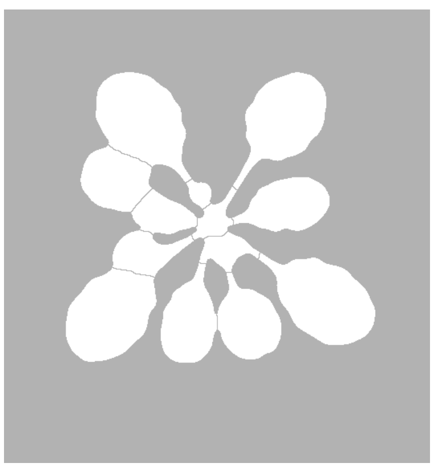
1.) RGB input image

Trainable Weka

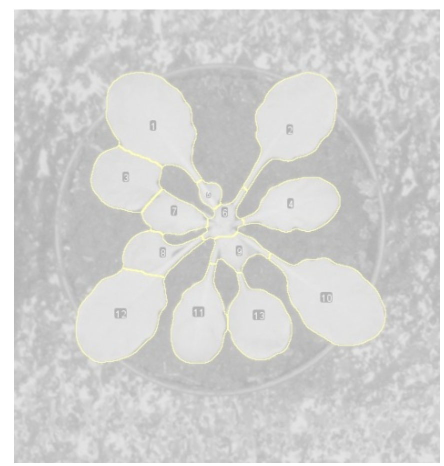
Segmentation



2.) Classification

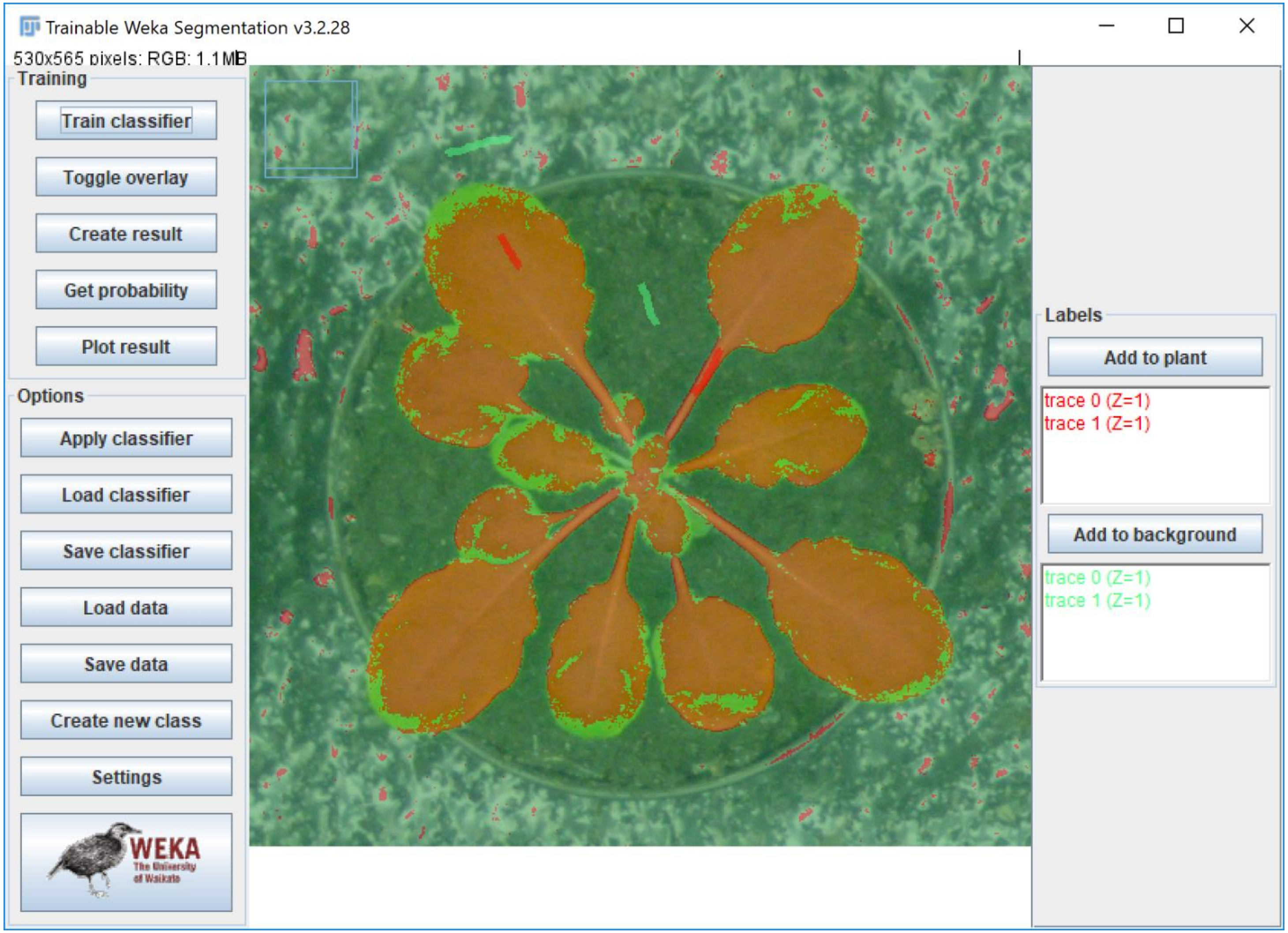


3.) Leaf Identification



4.) Leaf Analysis

Trainable Weka Segmentation



- Training a FastRandomForest classifier in the Weka GUI
- Using 6 diverse photos

Trainable Weka Segmentation Code

- Loads a trained classifier from a .model file
- Applies the classifier to a list of RGB images
- Returns classified image list

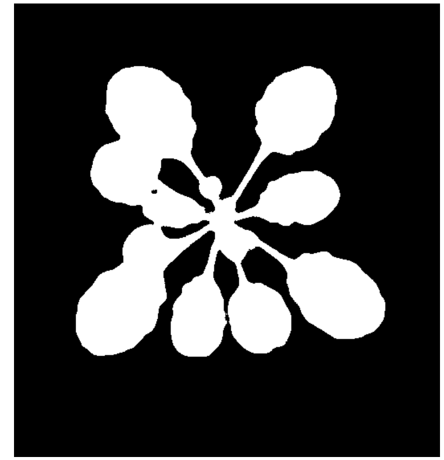
```java
public List<ImagePlus> applyClassifier(List<ImagePlus> imageList, String
path) {
    WekaSegmentation seg = new WekaSegmentation();
    boolean loaded = seg.loadClassifier(path);
    if (loaded) {
        System.out.println("***** Classifier loaded *****");
    } else if (!loaded) {
        System.out.println("***** Can't load classifier *****");
    }
    return applyClassifier(imageList, seg);
}
```

Trainable Weka Segmentation Code

```
public List<ImagePlus> applyClassifier(List<ImagePlus> imageList, WekaSegmentation
seg) {
    System.out.println("***** Apply classifier to folder *****");
    List<ImagePlus> resultList = new ArrayList<>();

    // iterate over imageList
    for (ImagePlus img : imageList) {
        // apply classifier and get results (0 indicates number of threads is auto-
detected)
        ImagePlus result = seg.applyClassifier(img, 0, false);
        result.setLut(Utils.getGoldenAngleLUT());

        // convert from red/green to grayscale
        ImageConverter imageConverter = new ImageConverter(result);
        imageConverter.convertToGray8();
        result.updateImage();

        // get B&W image
        IJ.run(result, "Convert to Mask", "Black Background");
        Prefs.blackBackground = true;
        IJ.run(result, "Make Binary", "white");

        result.setTitle(img.getShortTitle() + "_class");
        resultList.add(result);
    }
    return resultList;
}
```

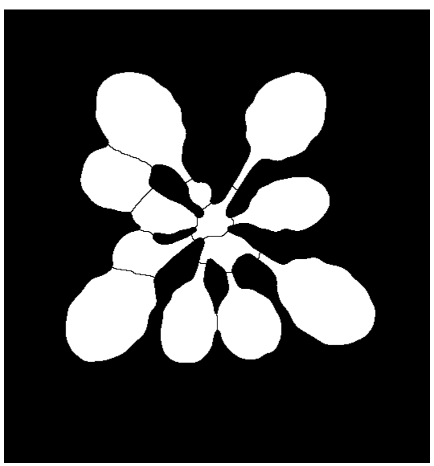
Image Analysis Pipeline

1.) RGB input image

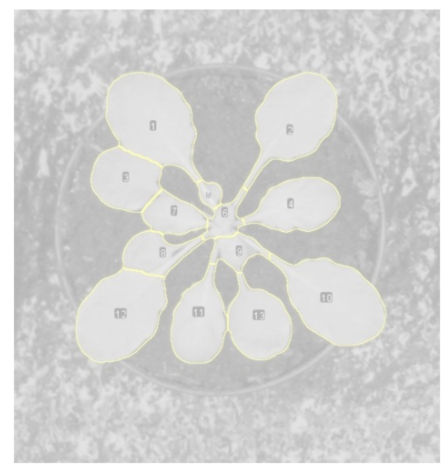


2.) Classification

Single Leaf Segmentation

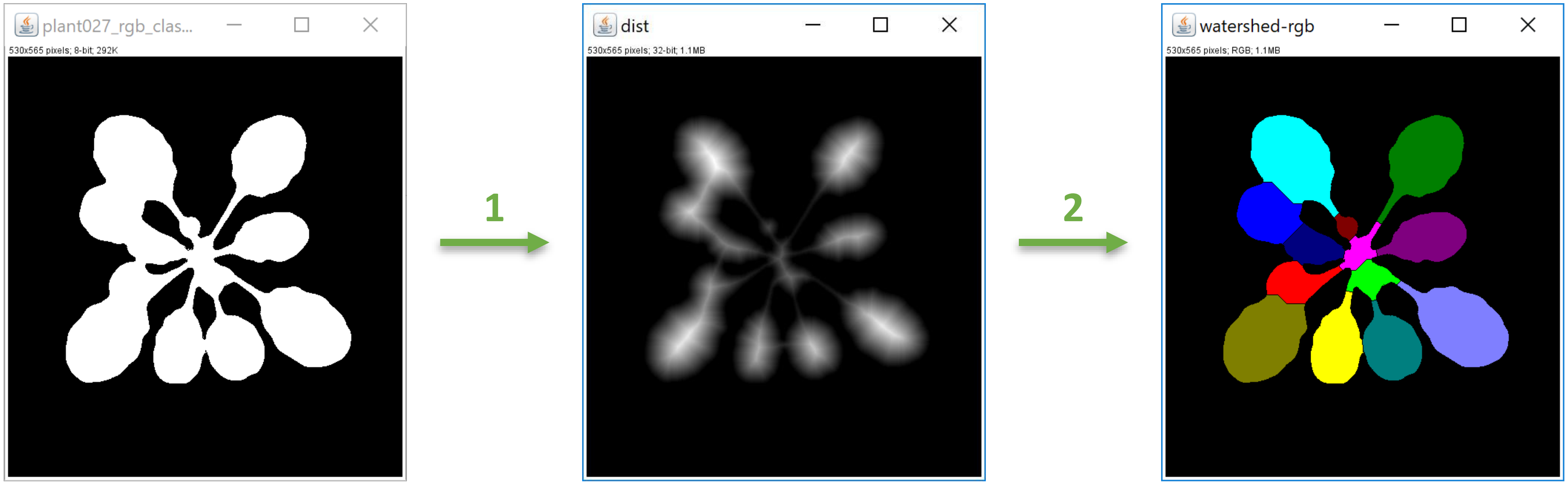


3.) Leaf Identification



4.) Leaf Analysis

Watershed Segmentation



- input: binary class image
- step 1: compute distance map, i.e. compute for each white pixel the chamfer distance to the nearest background (black) pixel
- step 2: draw a line (= separate two objects) at the „watershed“
- output: binary image with added watershed lines

Watershed Segmentation Code

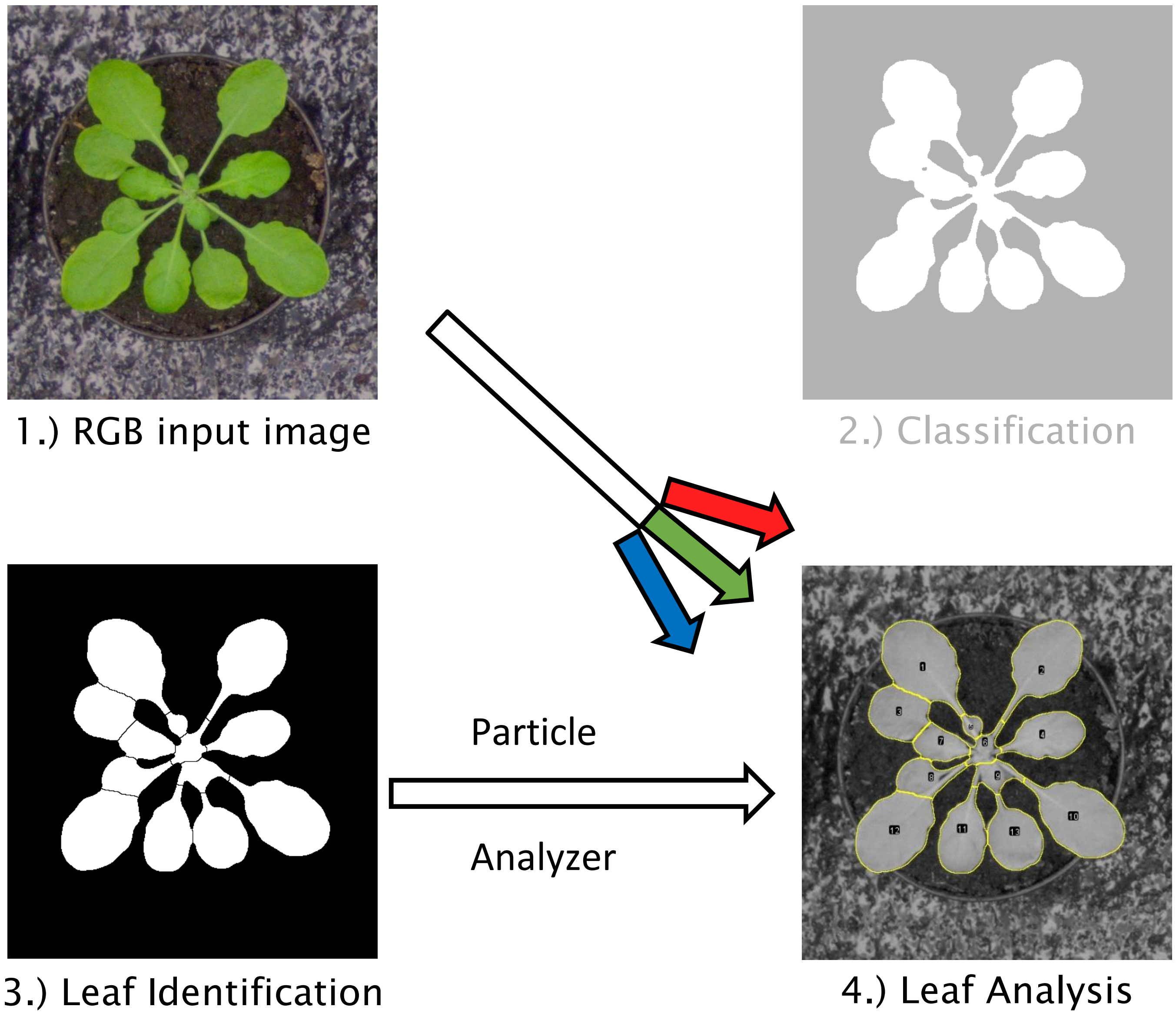
- Removal of outliers with a selective median filter to background and plant

```
public ImagePlus removeOutliers(ImagePlus input) {
    IJ.run(input, "Remove Outliers...", "radius=6 threshold=50 which=Dark");
    IJ.run(input, "Remove Outliers...", "radius=6 threshold=50 which=Bright");
    return input;
}
```

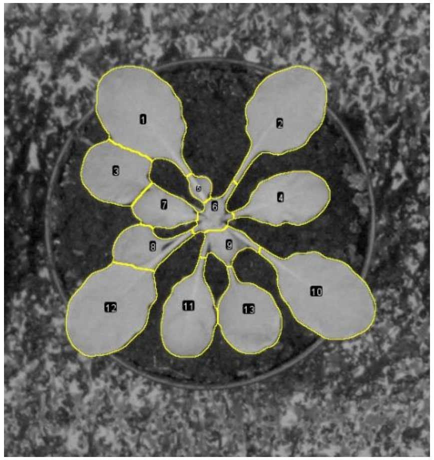
- Apply the watershed algorithm to a binary input image and returns a binary output image with watershed lines:

```
private ImagePlus findObjects(ImagePlus inputPlus) {
    ImagePlus resPlus = inputPlus.duplicate();
    IJ.run(resPlus, "Watershed", "only");
    String shortTitle = resPlus.getShortTitle();
    String DUPremoved = strip(shortTitle, "DUP_");
    resPlus.setTitle(DUPremoved + "_watershed");
    return resPlus;
}
```

Image Analysis Pipeline



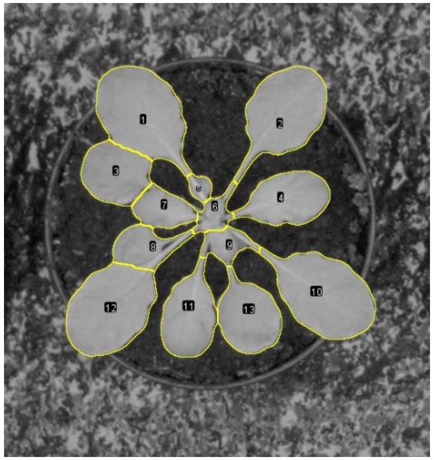
Particle Analyzer



- using particle analyzer (Fiji) to measure properties of single objects (leaves)
- do splitted red, green and blue channels for brightness calculation

Particle Analyzer	Information (in CSV output)
Area	Size
Centroid	pos_x and pos_y
Shape descriptors	roundness
Integrated density	brightness_sum
Mean gray value	brightness_average
Bounding rectangle	width_to_height_ratio
Feret‘s diameter	feret_min, feret_max, feret_ratio

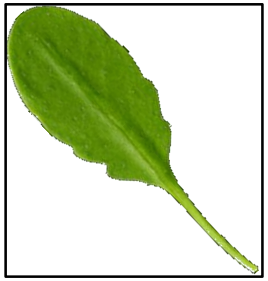
Particle Analyzer



- using particle analyzer (Fiji) to measure properties of single objects (leaves)
- do this for splitted red, green and blue channels

- Width-to-height ratio vs. feret ratio

height



width

Particle Analyzer Code

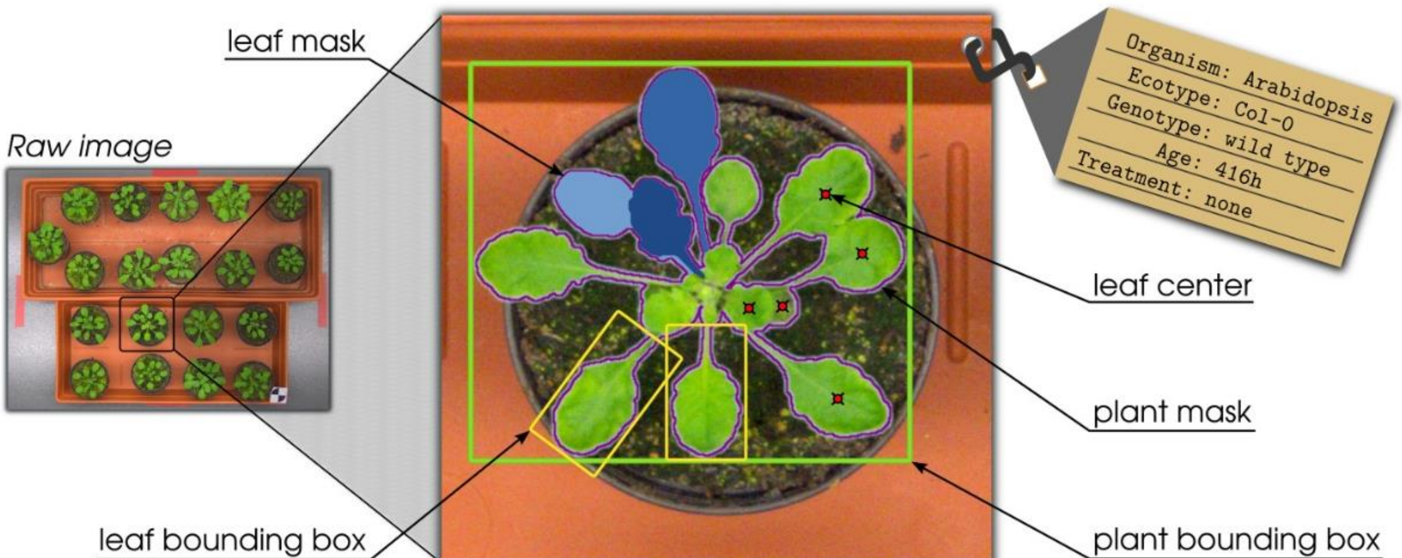
- Run the Fiji Particle Analyzer.
- Iterate over every leaf.
- Add measured data to a csv string.
- Export data to a csv file.

```
void analyzeImage(ImagePlus originalImage, ImagePlus watershedImage) {
    // Create a custom RoiManager to prevent it automatically opening a window
even in batch mode
    RoiManager rm = new RoiManager(true);
    ResultsTable resultsTable = new ResultsTable();
    ParticleAnalyzer.setResultsTable(resultsTable);
    ParticleAnalyzer.setRoiManager(rm);

    // Split the original image into RGB channels
    ImagePlus[] originalChannelImages = ChannelSplitter.split(originalImage);

    // Perform the analysis
    IJ.run("Set Measurements...", "area centroid bounding shape feret's display
redirect=None decimal=3");
    IJ.run(watershedImage, "Analyze Particles...", "size=70-Infinity exclude clear
add");
```

Live demonstration

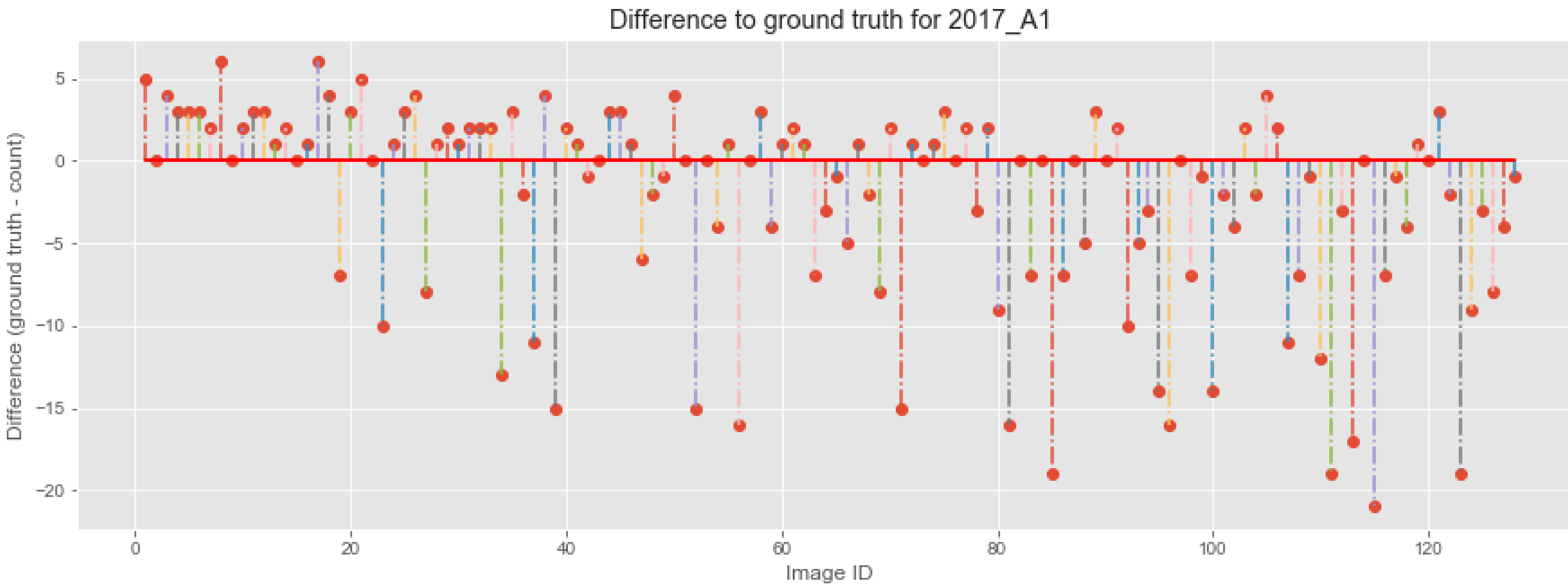
Explorative Data Analysis

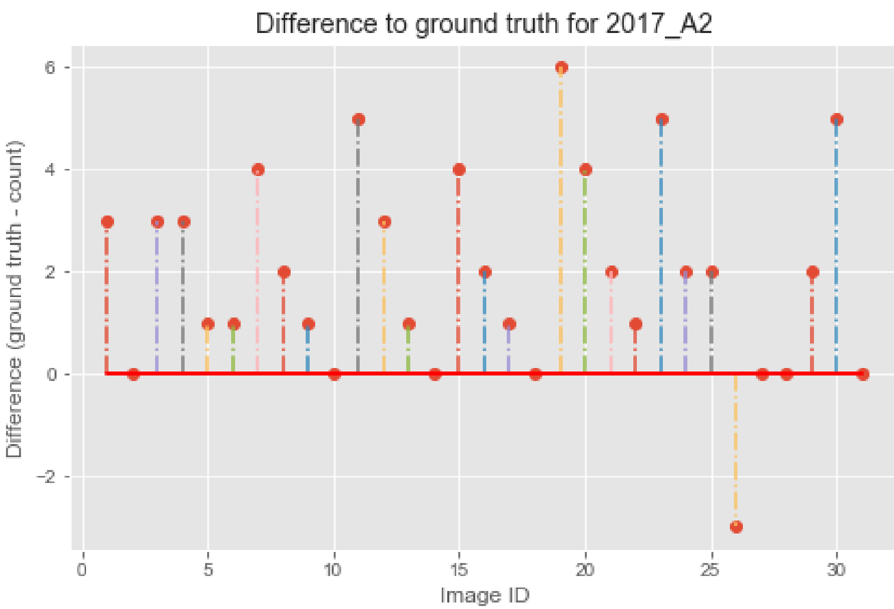
- Using pandas, seaborn, scipy, matplotlib in jupyter notebook



https://www.plant-phenotyping.org/datasets-home

Comparison of leaf counts to ground truth





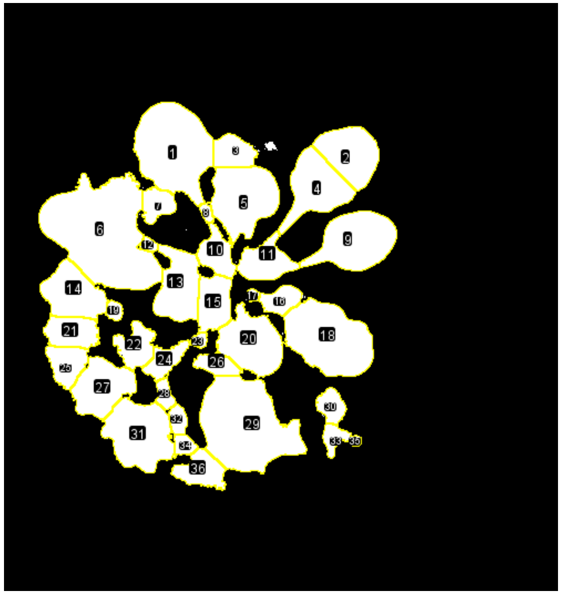
- Root mean squared error:
  - A1: 6.72
  - A2: 2.74

Bad performance of classifier with mossy plants

- high root mean squared error for A1 dataset
- trained classifier is not able to distinguish between moss and *Arabidopsis*

➔ high error rate in A1 dataset

Mossy input image



our classification

ground truth

Data Analysis – Overview of A2 Dataset

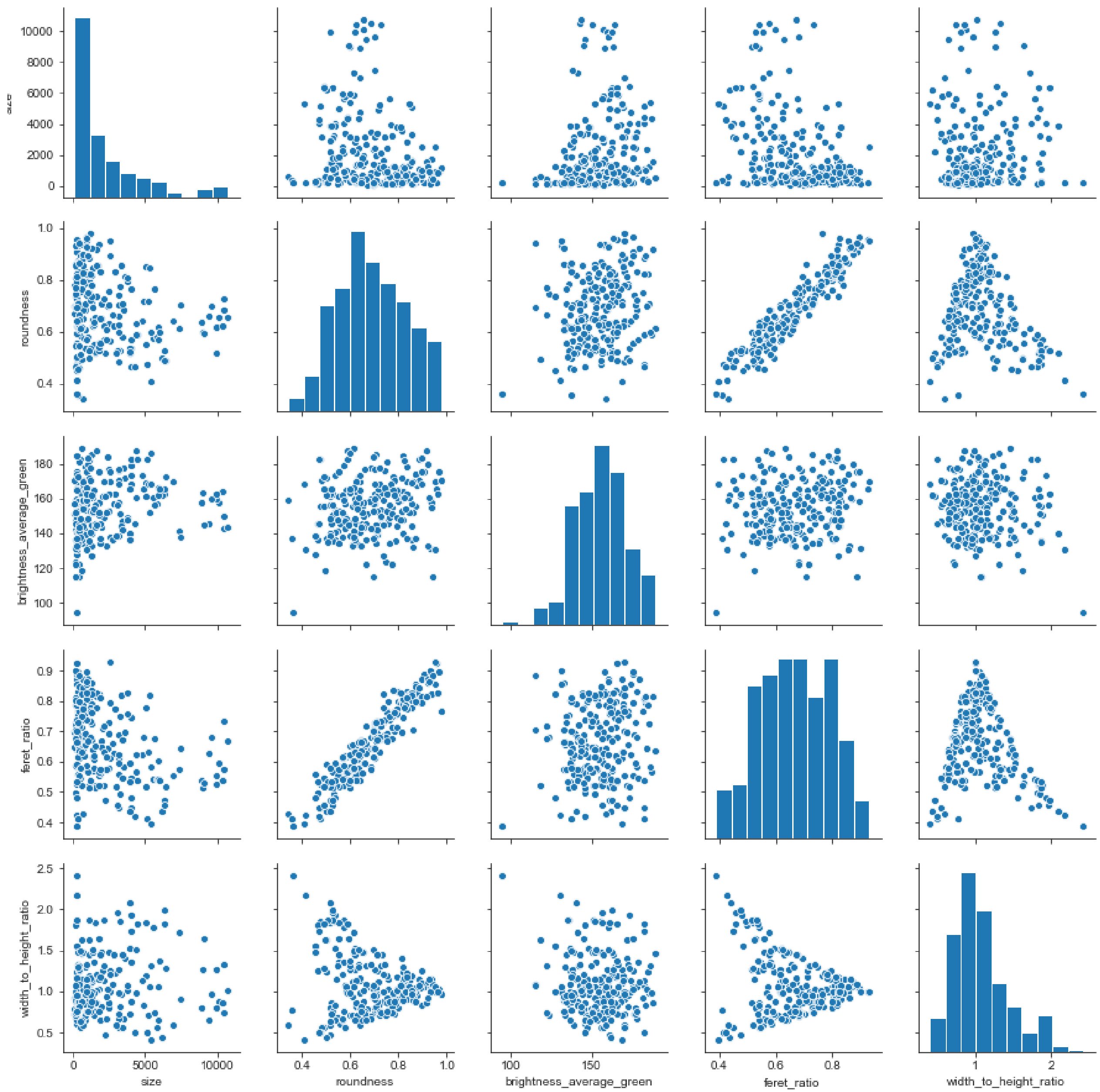
Size

Roundness

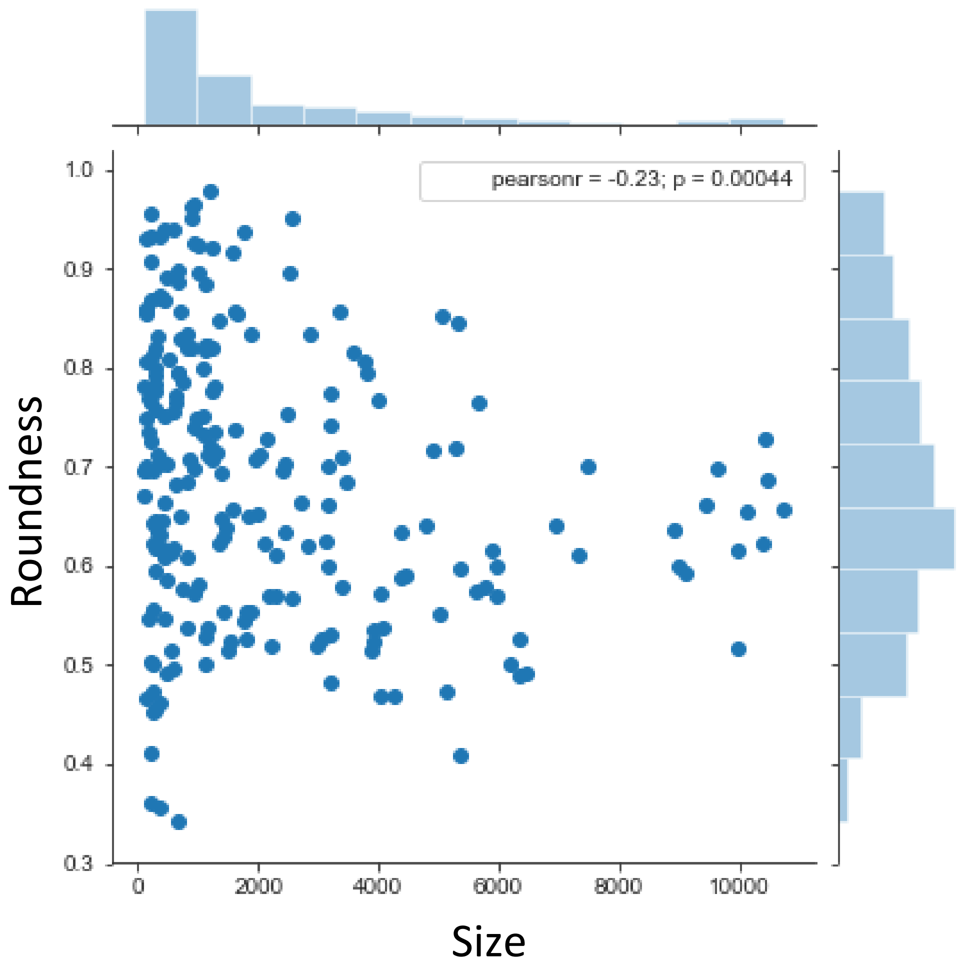
average brightness
of green

feret ratio

width to height ratio

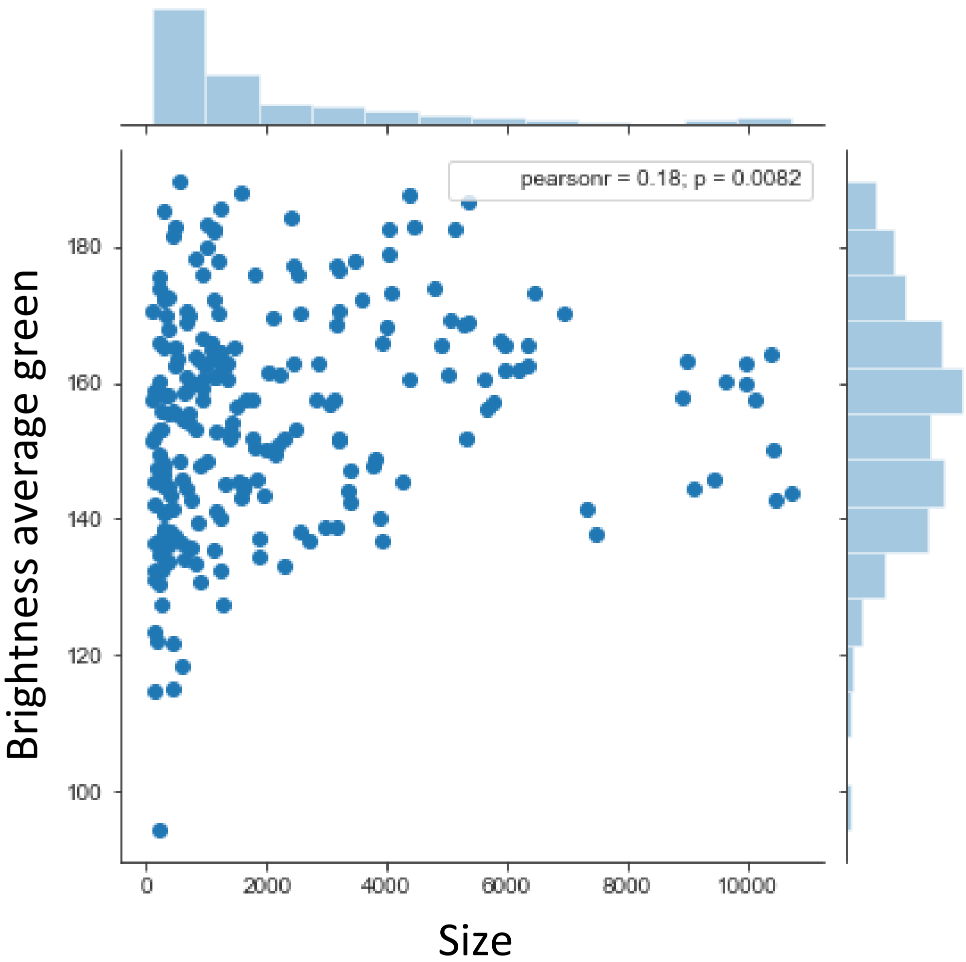


Does the size of a leaf relate to its roundness?



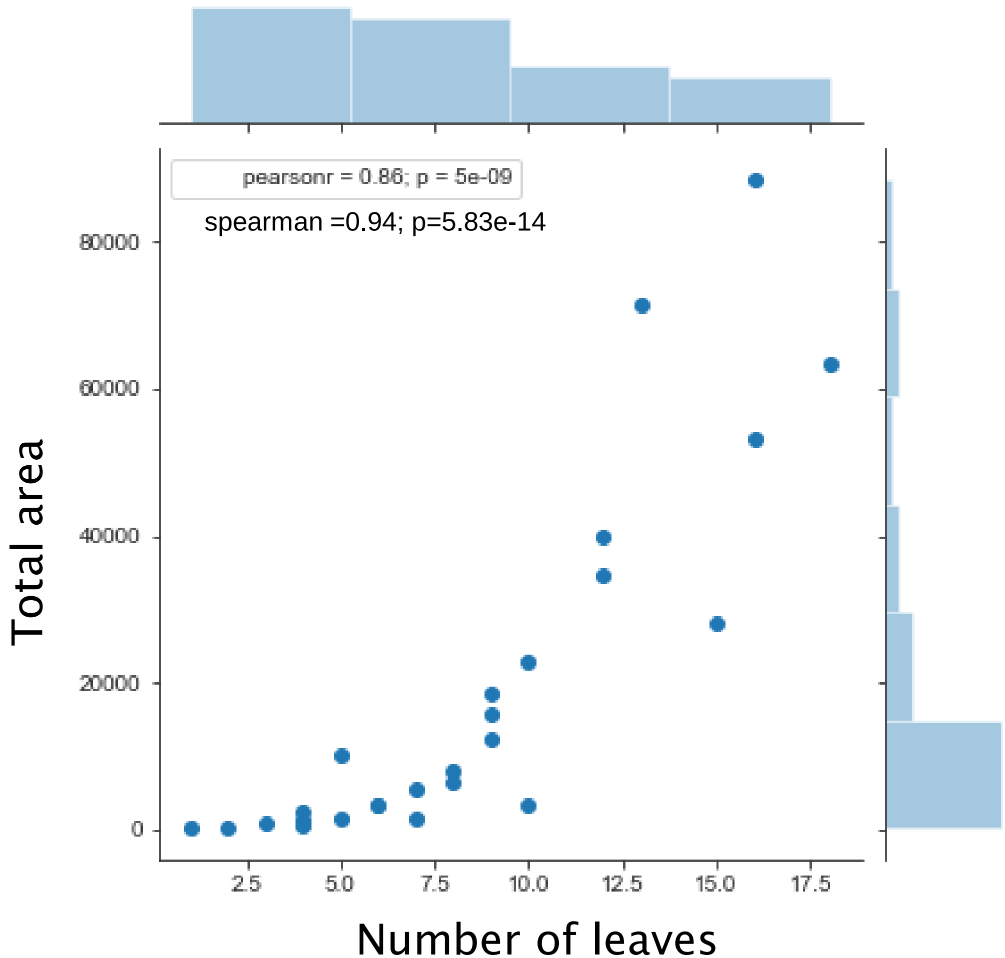
- No, size of a leaf and ist roundness are not correlated.

Does the size of a leaf relate to its color?



- No, size of a leaf is not correlated to its average green brightness.

Is the number of leaves correlated to total area?



- The number of leaves per plant is correlated to the total plant area.

Summary

- Our project's successes:
    - By using Trainable Weka Segmentation (with a FastRandomForest classifier) we were mostly able to separate a plant from its background.
    - We successfully managed to identify separate leaves from a plant, with the Watershed algorithm, in order to successfully analyze the individual leaves.
    - By applying various statistical functions, we were able to perform an in-depth data analysis on our results and display them in a meaningful way.

- What we can improve:
    - Improve the classifier to be able to identify moss as part of the background/noise and not as part of the image.

Thank you for your attention!

Questions?