Methods in Ecology and Evolution

pliman: an R package for plant image analysis

Journal:	*Methods in Ecology and Evolution*
Manuscript ID	MEE-21-06-465
Manuscript Type:	Application
Date Submitted by the Author:	11-Jun-2021
Complete List of Authors:	Olivoto, Tiago; Federal University of Santa Maria, Department of Agronomic and Environmental Science;
Keywords:	disease severity, count objects, image segmentation, leaf area, binary image
Abstract:	1. Quantitative measurements of leaf area, disease severity, number of disease lesions, number of grains, and object statistics such as grain length and width are vital to a growing range of researchers such as agronomists, breeders, phytopathologists, geneticists, ecologists, and biologists. Manual measurements are time-consuming and error-prone, so tools that automate these tasks and provide reliable results are needed. 2. Here I describe the pliman R package, a collection of functions designed (but not limited to) to plant image analysis. The package will help researchers to (a) manipulate, segment, and compute image indexes based on Red, Green, Blue, Red-Edge, and Near-Infrared bands; (b) measure plant leaf area; (c) measure disease severity; (d) count and extract statistics (area, perimeter, width, and length) of objects such as grains, pods, pollen; and (e) extract the RGB values for each object in an image. 3. In this paper, I present a summary of the functions implemented in the package guiding the user along a gentle learning curve with practical and reproducible examples. A validation study implemented to count objects in different scenarios (combination of seed sizes, background color, and image resolution) showed a high overall concordance (> 0.99), and that the reduction in accuracy is mainly related to the low contrast between objects and image background. 4. pliman offers a flexible, intuitive, and richly documented working environment with tools that will facilitate analyze plant images, being an interesting alternative to commercial and free point-and-click solutions.

`pliman`: an `R` package for plant image analysis

Tiago Olivoto[1,*]

[1] Department of Agronomic and Environmental Science, Federal University of Santa Maria, Frederico Westphalen, Rio Grande do Sul, Brazil

[*] Correspondence: Tiago Olivoto <tiagoolivoto@gmail.com>

Type: Application

Running headline: Plant image analysis

Abstract

1. Quantitative measurements of leaf area, disease severity, number of disease lesions, number of grains, and object statistics such as grain length and width are vital to a growing range of researchers such as agronomists, breeders, phytopathologists, geneticists, ecologists, and biologists. Manual measurements are time-consuming and error-prone, so tools that automate these tasks and provide reliable results are needed.
2. Here I describe the `pliman` R package, a collection of functions designed (but not limited to) to plant image analysis. The package will help researchers to (a) manipulate, segment, and compute image indexes based on Red, Green, Blue, Red-Edge, and Near-Infrared bands; (b) measure plant leaf area; (c) measure disease severity; (d) count and extract statistics (area, perimeter, width, and length) of objects such as grains, pods, pollen; and (e) extract the RGB values for each object in an image.
3. In this paper, I present a summary of the functions implemented in the package, guiding the user along a gentle learning curve with practical and reproducible examples. A validation study implemented to count objects in different scenarios (combination of seed sizes, background color, and image resolution) showed a high overall concordance (> 0.99), and that the reduction in accuracy is mainly related to the low contrast between objects and image background.
4. `pliman` offers a flexible, intuitive, and richly documented working environment with tools that will facilitate analyze plant images, being an interesting alternative to commercial and free point-and-click solutions.

Keywords: binary image, disease severity, count objects, image segmentation, leaf area.

1 Introduction

The term "phenotype" was characterized by Wilhelm Johannsen in 1911 as all "types" of organisms that can be distinguished by direct inspection or with finer methods of measurement or description (Johannsen, 1911). Of course, a wide range of professionals such as agronomists, geneticists, breeders, phytopathologists, ecologists, and biologists are interested in correctly and efficiently characterizing the "phenotype".

For breeders and agronomists, measuring leaf area is a very common task. The leaf is the most important photosynthesis organ of plants and is used as a key trait for computing indexes such as the Leaf Area Index (LAI), which quantifies the amount of leaf material in a canopy. LAI is being a key variable in a range of processes including gas exchange, water and nutrient cycling, and canopy health (Zheng & Moskal, 2009).

Phytopathologists are frequently interested in measuring plant disease severity, which is the proportion of the plant tissue exhibiting symptoms such as chlorosis and/or necrosis. Quantification of disease severity has been performed visually, with the adoption of appropriate diagrammatic scales to improve the accuracy of visual estimates (Franceschi, Alves, et al., 2020; Poletto et al., 2020). Visual estimation is time-consuming and error-prone. Therefore, correctly quantifying disease severity is essential for conducting epidemiological studies, predicting yield loss, and assessing control strategies. Image-based quantification of severity has advanced the capability of differentiating symptomatic from healthy tissue in digital images (Chiang, Bock, Lee, El Jarroudi, & Delfosse, 2016; Wang, Sun, & Wang, 2017). A review has shown that at least 20 software can be used to obtain actual severity measurements (Bock, Barbedo, Del Ponte, Bohnenkamp, & Mahlein, 2020). As far I know, there is no specialized R package or one with functions built particularly for plant disease severity measurement. The availability of free and open-source image analysis tools is of great importance, contributing to the advance of reproducible research.

Knowing the number of grains per plant (NGP), is also fundamental in many studies, allowing for computing grain density, e.g., thousand-grain weight (TGW). The NGP and TGW are frequently used in association studies such as path analysis (T. Olivoto et al., 2017; Del Conte, Souza Carneiro, De Resende, Da Silva, & Peternelli, 2020), thus, being used for indirect selection of high-yielding genotypes in plant breeding (Tiago Olivoto & Nardino, 2020). Morphological description of seeds such as width (W), length (L), area (A), perimeter (P), and aspect ratio (L/W ratio) are another set of traits used for germplasm characterization (Meira et al., 2017; Sari, Silverman, Reiland, & Wehner, 2021). Getting these measures manually, however, is time-consuming. Therefore, an automated and accurate image-based process would contribute significantly to speed up phenotyping in plant breeding studies.

My fundamental goal was to develop a user-friendly, R-based pipeline to accomplish the abovementioned tasks using images. The package, called `pliman`, short for **pl**ant **im**age **an**alysis, is built on the existing image analysis package `EBImage` (Pau, Fuchs, Sklyar, Boutros, & Huber, 2010), and is being distributed under the GNU General Public License 3 at https://github.com/TiagoOlivoto/pliman. In this application paper, I illustrate key features of the software providing example images and codes to ensure reproducibility.

2 The `pliman` package

The current stable version of the package (0.3.0) requires `R` 4.1.0 and can be installed from CRAN (https://CRAN.R-project.org/package=pliman) directly via the `R` console using `install.packages("pliman")`. The development version of the package is available on Github (https://github.com/TiagoOlivoto/pliman) and can be installed using `devtools::install_github("TiagoOlivoto/pliman")`.

Comprehensive details and examples of the functionality of `pliman` are available in

the online documentation (https://tiagoolivoto.github.io/pliman/). Indeed, I strongly encourage readers to refer to the vignettes as the primary source for information on `pliman`'s functionality since they are updated regularly with every package release. Table 1 presents the main functions available in `pliman` 0.3.0.

Table 1. Key features available in `pliman` version 0.3.0 for plant image analysis.

Function	Goal
Spatial transformations	
`image_autocrop()`	Crops the image to the area of objects
`image_dimension()`	Gives the dimension (width and height) of an image
`image_rotate()`	Rotates the image clockwise by the given angle
`image_horizontal(),image_vertical()`	Converts (if needed) an image to a horizontal/vertical image
`image_hreflect(),image_vreflect()`	Performs horizontal/vertical reflection of an image
`image_resize()`	Resize the image
Utilities for image resolution	
`dpi_to_cm(), cm_to_dpi()`	Conversions between dots per inch (dpi) and centimeters (cm)
`pixels_to_cm(), cm_to_pixels()`	Conversions between pixels and centimeters
Image segmentation	
`image_binary()`	Reduce a color or grayscale images to a binary image
`image_index()`	Builds image indexes using Red, Green, Blue, Red-Edge, and Near-Infrared bands
`image_palette()`	Creates image palettes by applying the k-means algorithm to the RGB values
`image_segment()`	Segment an image using RGB values
`prop_segmented()`	Iterative image segmentation with pixels proportion
Image analysis	
`count_lesions()`	Count the number of disease lesions in an image
`count_objects()`	Count the number of objects in an image
`get_measures(), plot_measures()`	Get object measures/plot measures in the image
`leaf_area()`	Computes the leaf area using leaf images
`symptomatic_area()`	Computes the percentage of symptomatic area

2.1 Spatial transformations

`pliman` extends functionalities from `EBImage` package (Pau, Fuchs, Sklyar, Boutros, & Huber, 2010) to perform the transformation of a single or list of images, possibly with parallel processing. Users import images with `image_import()` using either a character

vector of file names or URLs. Using the argument `img_pattern`, several images can be imported at once, and a list of images is created (See an example in section 3.1 of the supplementary material). Image operations such as spatial transformations and image segmentation (Table 1) can be applied to each image or list of images, that can further be exported with `image_export()`.

2.2 Image segmentation

Image segmentation is the process of partitioning a digital image into multiple segments, also known as image objects (Yanowitz & Bruckstein, 1989). In the context of plant image analysis, segmentation is used to simplify the representation of an image into something easier to analyze. For example, when using `count_objects()` to count crop grains (Figure 1a), first the grains need to be isolated (segmented) from the background. In `pliman`, Otsu's method of automatic threshold selection (Otsu, 1979) implemented in the `EBImage` package is used. First, `image_index()` is used to produce the grayscale image needed for automatic segmentation (Figure 1b) that after can be converted to a binary image with `image_binary()` function (Figure 1c) or to a segmented image with `image_segment()` function (Figure 1d).

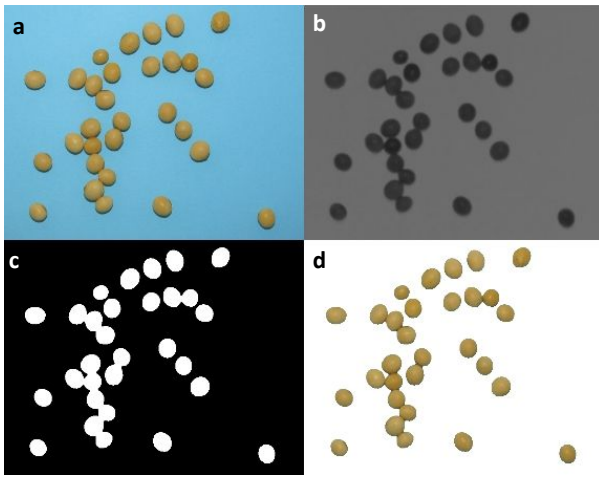

Figure 1. Image segmentation using `pliman`. The original image showing soybean grains (**a**) is first used to produce a grayscale image (**b**) with `image_index()`. The grayscale image is then used to produce a binary image (**c**) with `image_binary()`,

or a segmented image (**d**) with `image_segment()`.

2.3 Leaf area

The leaf area can be measured using leaf images in two ways. The first, using `leaf_area()`, uses a sample of leaves along with a template with a known area. Background, leaf, and template color palettes must be declared. The area correction (from pixels to cm^2) is performed internally since the template area is explicitly informed (See section 6.1 in the supplementary material for more details).

The second alternative is by using `count_objects()`. The advantage of using `count_objects()` is that the image binarization is performed with `image_binary()` using RGB-based indexes. So, sample palettes don't need to be informed. The disadvantage is that users will need to adjust the measures using the template area for each processed image. Figure 2 shows the steps for computing the leaf area of five tree leaves using `count_objects()` function (See the codes used in section 6.2 of the supplementary material).

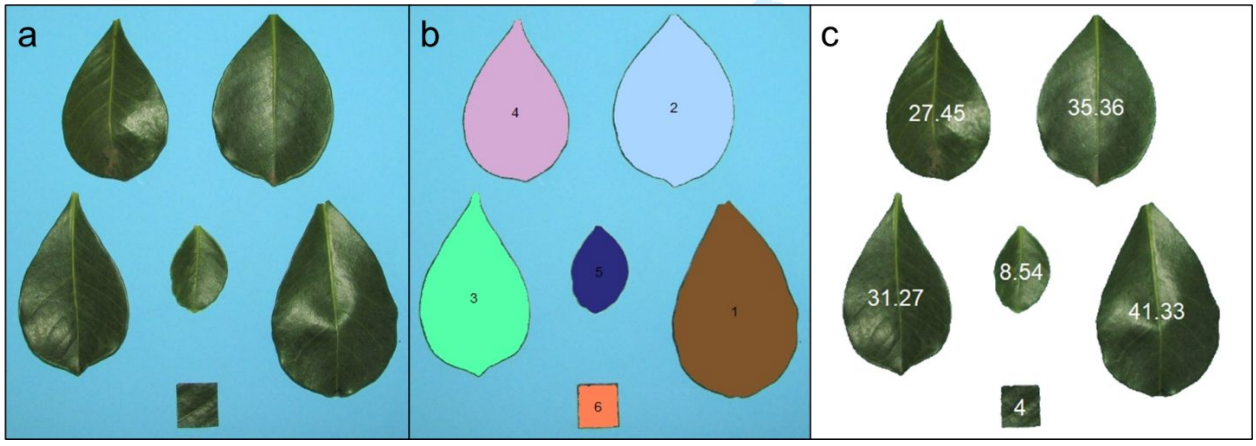

Figure 2. Steps used to compute the leaf area in `pliman`. A figure with the leaves for which the area needs to be computed along with a sample template of known area (4 cm^2) is required (**a**). The function `count_objects()` using the argument `marker = "text"` is used to compute the object features and plot the object id (**b**). After knowing the id of the sample template, the leaf area (originally given in pixels unit) is adjusted with `get_measures()` and the values (in this case in cm^2) plotted for each object (c)

using `plot_measures()`.

2.4 Disease severity

Disease severity is computed with `symptomatic_area()`. The function calculates the percentage of symptomatic leaf area in a sample or entire leaf based on provided color palette samples. A general linear model (binomial family) fitted to the RGB values is used to segment the lesions from the healthy leaf. If a background color palette is provided, the function takes care of the details to isolate it before computing the area of lesions. By using the `img_pattern` argument it is possible to process several images with common pattern names that are stored in the current working directory or the subdirectory informed in the `dir_original` argument.

Figure 3 shows an example of disease quantification. In this case, image segmentation is automated based on the color palettes provided. These reference palettes can be made by simply manually sampling small areas of representative images and producing a composite image. Of course, the results may vary significantly depending on how these areas are chosen and are subjective due to the researcher's experience. Thus, I strongly suggest that users perform a previous observation of the processed mask to create image palettes that are most representative of the respective class. At the end of the process, a data frame with the proportion of diseased and healthy tissues is returned. Optionally, users can save the processed image with `save_image = TRUE` argument. A high level of personalization is provided, allowing users to create masks, changing the color of the background, diseased, and healthy tissues (Figure 3).

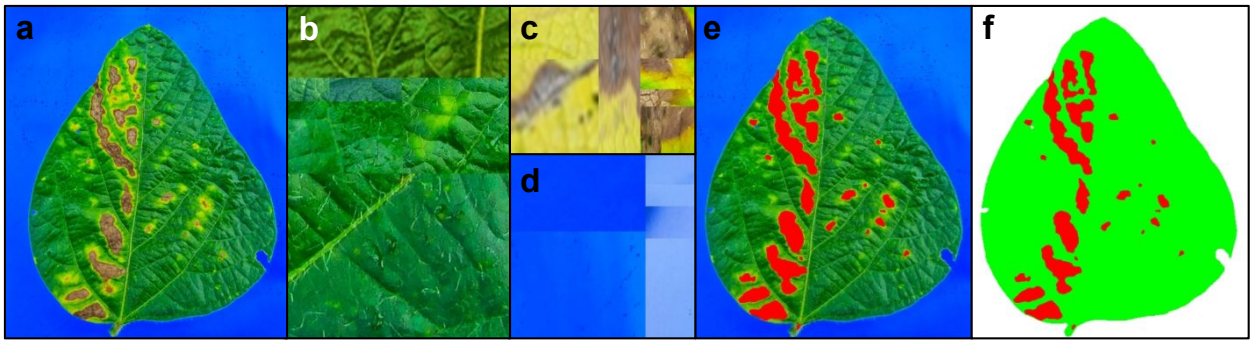

Figure 3. Disease severity quantification in `pliman`. A soybean leaflet in a blue background showing phytotoxicity symptoms is shown in **a**. The images in **b-d**, represent the color palettes for the healthy tissues, diseased tissues, and background, respectively. In **e**, the processed image with the symptomatic tissues highlighted in red color can be obtained with the argument `show_image = TRUE`. A mask (**f**) can also be obtained with the `show_original = FALSE` argument. The diseased area on this leaflet is 10.96%

A case study for determining the severity of soybean leaflets affected by soybean rust (*Phakopsora pachyrhizi*) was performed. Fifty soybean leaflets previously processed in QUANT software (Franceschi, Duarte, & Del Ponte, 2020) were used as actual/reference severity. The concordance between `pliman` measures and the actual/reference severity was measured by Lin's Concordance Correlation Coefficient (CCC) analysis (Lin, 1989). Using parallel processing, the 50 images were processed in 35 s. The CCC was 0.992 (95% CI [0.987-0.995]), suggesting an excellent agreement between pliman and QUANT measurements.

2.5 Count objects and compute features

In `pliman`, this is facilitated by the function `count_objects()`. This function uses the watershed segmentation (Meyer, 1994) routine from the `EBImage` package (Pau et al., 2010) to identify distinct objects even when they are touching one other (Figure 1a). After object segmentation, object features are computed and returned as a data frame.

Figure 4b shows the results of a validation study implementing the function

192 `count_objects()` to count seeds of different sizes and shapes (soybean, wheat, and
193 bean seeds). Thirteen different numbers of seeds (ranging from 12 to 100) were
194 manually counted twice and served as reference/actual values. Images were obtained
195 with a Digital Camera in an original resolution of 3264 x 2448 pixels (8 Megapixels) with
196 two background colors (blue and white). A resolution of 1632 x 1224 pixels (2
197 Megapixels) was created using the argument `resize = 50` to study the effect of the
198 image resolution on the accuracy. A total of 154 images were analyzed.

199 The overall CCC was 0.991 (95% CI [0.988-0.993]). Soybean seeds counted in
200 images with blue background presented a perfect concordance (CCC = 1),
201 independently of the image resolution. Underestimated values were observed when
202 counting bean seeds with a white background. This is justified because bean seeds have
203 pixels with a similar color to the background (See section 8.1 of supplementary
204 material).

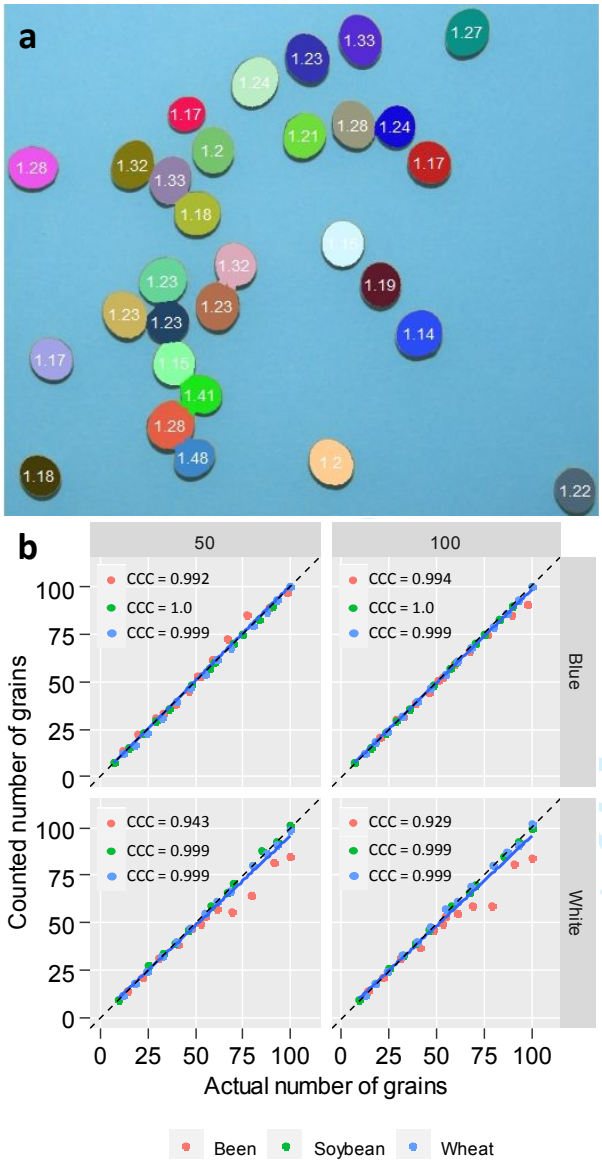

Figure 4. Counting soybean grains in `pliman`. (**a**) Soybean grains that were counted with `count_objects()`. The number in each grain represents the aspect ratio (maximum radius / minimum radius) and was plotted with `plot_measures()`. (**b**) Identity line with the concordance correlation coefficient (CCC) between the real and counted number of seeds `count_objects()`.

2.6 Batch processing

Usually, we don't have a single image to process but several. For example, in plant breeding, the number of grains per plant (e.g., wheat) is frequently used in the indirect selection of high-yielding plants. So, it is not a surprise if lots of images (say, 500 or

more) need to be processed. It would be very tedious and time-consuming to process the images manually, changing the image name in the function each time. In pliman, batch processing can be done when the user declares the argument `img_pattern`. This feature is available in the functions `count_objects()`, `leaf_area()`, `objects_rgb()`, and `symptomatic_area()`.

Image size and resolution can drastically impact the processing time. To speed up the process, I recommend two approaches: (i) reduce the image resolution using the argument `resize` available in the abovementioned functions; and (ii) use multiple sections by setting the argument `parallel = TRUE` in the same functions.

To assess how image size and parallel programming affect the computational efficiency and concordance of computed results, a benchmark study with 15 images was performed in an Intel(R) Core(TM) i7-9750H CPU @2.6GHz laptop with 16 GB RAM. Different scenarios were planned by the combination of sequential and parallel processing with different image resolutions, namely, original (100 = 769 x 802 pixels), and reduced (50 = 384 x 401 pixels). In sequential programming, the function is applied to each image sequentially using the `lapply()` approach. In parallel programming, the images are processed asynchronously (in parallel) in separate R sessions running in the background on the same machine using the R base `parallel` package. The number of sections is by default set up to 90% of available cores. This number can be controlled explicitly with the argument `workers`.

The reduction in time computation was more pronounced when reducing the image resolution to 50% of the original size (Table 2). The greater reduction (~77%) was observed for `count_objects()`, which taken 9.7s (~0.6 s image^{-1}) and 2.36s (~0.15 s image^{-1}) to count the number of objects in an image with original and reduced resolution, respectively. The parallel approach reduced the time processing for images with original resolution but increased the time processing of images with low resolution. This is explained since we need to export all objects (e.g., scope for variables,

libraries) to be accessible to all cores. So, it is expected that in such a case, parallel processing will provide a significant reduction in time when processing hundreds of images. For all the functions, the results observed in the two image resolutions were highly correlated (CCC > 0.999, P < 0.01).

Table 2. Benchmark results for multiple image processing. The Elapsed time (s) is shown for sequential (parallel = FALSE) and multi-section (parallel = TRUE) processing of 15 images with an original resolution (100), and a reduced resolution equivalent to 50% of the original resolution (50). The concordance correlation coefficient (CCC) shows the agreement between computed values of images with different resolutions.

Function	parallel = FALSE		CCC	parallel = TRUE		CCC
	100	50		100	50	
`count_objects()`	9.7	2.36	1.00	7.56	4.52	1.00
`leaf_area()`	17.17	8.54	0.9999	11.27	7.37	0.9999
`objects_rgb()`	13.58	4.04	0.9999	8.06	4.48	0.9999
`symptomatic_area()`	30.22	13.7	0.9999	17.87	10.11	0.9999

3 Concluding remarks and future improvements

The `R` package `pliman` was designed to make it easier to solve common problems faced by agronomists, breeders, biologists, and ecologists, such as counting objects, measuring leaf area, and disease severity using images. Therefore, `pliman` can become an interesting alternative to commercial software or other point-and-click open-source solutions such as imageJ. One important strength is the implementation of a batch processing approach, which can dramatically improve the speed of the assessments, especially when users run parallel processing. One limitation faced in plant disease severity measurements -which is no different from any other software based on color-threshold segmentation- is that users need a few preliminary runs to define the color palettes and check whether the segmentation is being performed correctly, based on visual judgment. In a near future, my next efforts will be focused on implementing alternative image segmentation methods such as adaptative thresholding, k-means clustering, and deep convolutional neural

267 network.

268

269 ## 4 Acknowledgment

270 I would like to thank Emerson M. Del Ponte and his collaborators for maintaining the Plant
271 Disease Severity Annotation Image Database, which allowed comparing the pliman measures
272 with previously processed images. Many thanks to Ângela Biondo and Ranye Zambam for
273 providing the image of the soybean tetrazolium test, which was used to show the feature of
274 computing the RGB for each image object. The author has no conflicts of interest to declare.

275

276 ## 5 Data accessibility

277 The source code used in this manuscript has been archived at
278 https://doi.org/10.5281/zenodo.4923021 as `pliman` version 0.3.0. A website with the
279 data, codes, and results used in this paper is available at
280 https://tiagoolivoto.github.io/paper_pliman/. If you want to keep in touch with the
281 latest `pliman`'s news, I invite you to download the development version from
282 GitHub (https://github.com/TiagoOlivoto/pliman). Package vignettes are also open-
283 source, accessible at https://tiagoolivoto.github.io/pliman/.

284

285 ## 6 References

286 Bock, C. H., Barbedo, J. G. A., Del Ponte, E. M., Bohnenkamp, D., & Mahlein, A.-K.
287 (2020). From visual estimates to fully automated sensor-based measurements of
288 plant disease severity: status and challenges for improving accuracy. *Phytopathology*
289 *Research*, *2*(1), 1–30. doi:10.1186/s42483-020-00049-8

290 Chiang, K. S., Bock, C. H., Lee, I. H., El Jarroudi, M., & Delfosse, P. (2016). Plant disease
291 severity assessment-how rater Bias, assessment method, and experimental design
292 affect hypothesis testing and resource use efficiency. *Phytopathology*, *106*(12), 1451–

1464. doi:10.1094/PHYTO-12-15-0315-R

Del Conte, M. V., Souza Carneiro, P. C., De Resende, M. D. V., Da Silva, F. L., & Peternelli, L. A. (2020). Overcoming collinearity in path analysis of soybean [Glycine max (L.) Merr.] grain oil content. *PLoS ONE*, *15*(5), e0233290. doi:10.1371/journal.pone.0233290

Franceschi, V. T., Alves, K. S., Mazaro, S. M., Godoy, C. V., Duarte, H. S. S., & Del Ponte, E. M. (2020). A new standard area diagram set for assessment of severity of soybean rust improves accuracy of estimates and optimizes resource use. *Plant Pathology*, *69*(3), 495–505. doi:10.1111/ppa.13148

Franceschi, V. T., Duarte, H. S. S., & Del Ponte, E. M. (2020). Soybean rust severity annotation. Retrieved from https://osf.io/4hbr6

Johannsen, W. (1911). The Genotype Conception of Heredity. *The American Naturalist*, *45*(531), 129–159. doi:10.1086/279202

Lin, L. I.-K. (1989). A Concordance Correlation Coefficient to Evaluate Reproducibility. *Biometrics*, *45*(1), 255. doi:10.2307/2532051

Meira, D., Meier, C., Olivoto, T., Nardino, M., Rigatti, A., Lunkes, A., … Souza, V. Q. (2017). Physiological traits and their relationships in black oat populations. *Genetics and Molecular Research*, *16*(4), gmr16039814. doi:10.4238/gmr16039814

Meyer, F. (1994). Topographic distance and watershed lines. *Signal Processing*, *38*(1), 113–125. doi:10.1016/0165-1684(94)90060-4

Olivoto, T., Souza, V. Q., Nardino, M., Carvalho, I. R., Ferrari, M., Pelegrin, A. J., … Schmidt, D. (2017). Multicollinearity in path analysis: a simple method to reduce its effects. *Agronomy Journal*, *109*(1), 131–142. doi:10.2134/agronj2016.04.0196

Olivoto, Tiago, & Nardino, M. (2020). MGIDI: toward an effective multivariate selection in biological experiments. *Bioinformatics*. doi:10.1093/bioinformatics/btaa981

Otsu, N. (1979). Threshold selection method from gray-level histograms. *IEEE Trans Syst Man Cybern*, *SMC-9*(1), 62–66. doi:10.1109/tsmc.1979.4310076

Pau, G., Fuchs, F., Sklyar, O., Boutros, M., & Huber, W. (2010). EBImage-an R package for image processing with applications to cellular phenotypes. *Bioinformatics*, *26*(7), 979–981. doi:10.1093/bioinformatics/btq046

Poletto, T., Muniz, M. F. B., Dal'Col Lucio, A., Fantinel, V. S., Heldwein, A. B., Reiniger, L. R. S., & Blume, E. (2020). Diagrammatic scale for quantifying severity of brown leaf spot on carya illinoinensis. *Anais Da Academia Brasileira de Ciencias*, *92*, 1–7. doi:10.1590/0001-3765202020180889

Sari, N., Silverman, E., Reiland, D., & Wehner, T. C. (2021). Seed characterization and relationships between seed and cotyledon properties in lagenaria spp. Accessions. *HortScience*, *56*(2), 185–192. doi:10.21273/HORTSCI15569-20

Wang, G., Sun, Y., & Wang, J. (2017). Automatic Image-Based Plant Disease Severity Estimation Using Deep Learning. *Computational Intelligence and Neuroscience*, *2017*. doi:10.1155/2017/2917536

Yanowitz, S. D., & Bruckstein, A. M. (1989). A new method for image segmentation. *Computer Vision, Graphics, & Image Processing*, *46*(1), 82–95. doi:10.1016/S0734-189X(89)80017-9

Zheng, G., & Moskal, L. M. (2009). Retrieving Leaf Area Index (LAI) Using Remote Sensing: Theories, Methods and Sensors. *Sensors*, *9*(4), 2719–2745. doi:10.3390/s90402719