DOWNLOAD PDF SCIPY V0.19 TUTORIAL

Chapter 1 : Getting Started â€" calendrierdelascience.com

TutorialÂ¶. Tutorials with worked examples and background information for most SciPy submodules.

Licence This is an open access article distributed under the terms of the Creative Commons Attribution License , which permits unrestricted use, distribution, reproduction and adaptation in any medium and for any purpose provided that it is properly attributed. Image analysis software for high-throughput plant phenotyping. Abstract Systems for collecting image data in conjunction with computer vision techniques are a powerful tool for increasing the temporal resolution at which plant phenotypes can be measured non-destructively. Computational tools that are flexible and extendable are needed to address the diversity of plant phenotyping problems. We previously described the Plant Computer Vision PlantCV software package, which is an image processing toolkit for plant phenotyping analysis. The goal of the PlantCV project is to develop a set of modular, reusable, and repurposable tools for plant image analysis that are open-source and community-developed. Here we present the details and rationale for major developments in the second major release of PlantCV. In addition to overall improvements in the organization of the PlantCV project, new functionality includes a set of new image processing and normalization tools, support for analyzing images that include multiple plants, leaf segmentation, landmark identification tools for morphometrics, and modules for machine learning. However, manual plant measurements are time-consuming and often require destruction of plant materials in the process, which prevents measurement of traits for a single plant through time. Targeted plant phenotypes can range from measurement of gene expression, to flowering time, to grain yield; therefore, the software and hardware tools used are often diverse. Here, we focus on the software tools required to nondestructively measure plant traits through images. This is a challenging area of research because the visual definition of phenotypes vary depending on the target species. For example, identification of petals can be used to measure flowering time, but petal color can vary by species. Therefore, software tools needed to process high-throughput image data need to be flexible and amenable to community input. The scale that might be considered high-throughput for root phenotyping might not be the same for shoot phenotyping, which can be technically easier to collect depending on the trait and species. Here we define high-throughput as thousands or hundreds of thousands of images per dataset. PlantCV is an open-source, open-development suite of analysis tools capable of analyzing high-throughput image-based phenotyping data Fahlgren et al. First, GitHub was used as a platform to organize the community by integrating version control, code distribution, documentation, issue tracking, and communication between users and contributors Perez-Riverol et al. Third, a focus on modular development fosters code reuse and makes it easier to integrate PlantCV with new or existing systems. Finally, the use of a permissive, open-source license MIT allows PlantCV to be used, reused, or repurposed with limited restrictions, for both academic and proprietary applications. The focus of the paper associated with the original release of PlantCV v1. Since the release of PlantCV v1. Here we document the structure of PlantCV v2 along with examples that demonstrate new functionality. The release for this paper is v2. Scripts, notebooks, SQL schema, and simple input data associated with the figures and results presented in this paper are available on GitHub at https: Project-specific GitHub repositories are kept separate from the PlantCV software repository because their purpose is to make project-specific analyses available for reproducibility, while the main PlantCV software repository contains general purpose image analysis modules, utilities, and documentation. Images of Arabidopsis thaliana were captured with a Raspberry Pi computer and camera in a Conviron growth chamber. Additional details about the imaging set-up are provided in a companion paper Tovar et al. Images of Setaria viridis A10 and Setaria italica B are from publicly available datasets that are available at http: Images of wheat Triticum aestivum L. Image analysis was done in PlantCV using Python v2. Statistical analysis and data visualization was done using R v3. Graphs were produced using Matplotlib v2. Results and Discussion The following are details on improvements to the structure, usability, and functionality of PlantCV since the v1. Further documentation for using PlantCV can be found at the project website http: A pipeline can be as long or as short as it needs to be, allowing for maximum flexibility for users using different imaging systems and analyzing features of seed, shoot, root, or other plant systems.

Suggestions on how to approach image analysis with PlantCV, in addition to specific tutorials, are available through online documentation http: Each function has a debugging option to allow users to view and evaluate the output of a single step and adjust parameters as necessary. A PlantCV pipeline is written by the user as a Python script. The parallelization script also functions to manage data by consolidating measurements and metadata into an SQLite database Fig. In terms of speed, the user is only limited by the complexity of the pipeline and the number of available processors. Diagram of the components of PlantCV. A PlantCV is an open-source, open-development suite of image analysis tools. B Overview of the structure of the SQLite database. Contributors to PlantCV submit bug reports, develop new functions and unit tests, or extend existing functionality or documentation. Core PlantCV developers do not filter additions of new functions in terms of perceived impact or number of users but do check that new functions follow the PlantCV contribution guide see the sections on contributing in the online documentation. Additions or revisions to the PlantCV code or documentation are submitted for review using pull requests via GitHub. The pull request mechanism is essential to protect against merge conflicts, which are sections of code that have been edited by multiple users in potentially incompatible ways. In PlantCV v2, several service integrations were added to automate common tasks during pull requests and updates to the code repository. A continuous integration framework using the Travis CI service https: Continuous integration provides a safeguard against code updates that break existing functionality by providing a report that shows which tests passed or failed for each build Wilson et al. The effectiveness of continuous integration depends on having thorough unit test coverage of the PlantCV code base. In addition to the code, the PlantCV documentation was enhanced to use a continuous documentation framework using the Read the Docs service https: The documentation was updated to cover all functions in the PlantCV library, tutorials on building pipelines and using specialized tools e. Improved usability PlantCV v1. In command-line mode, an entire pipeline script must be executed, even if only a single step is being evaluated. To improve the pipeline and function development process in PlantCV v2, the debugging system was updated to allow for seamless integration with the Juptyer Notebook system http: Jupyter compatibility allows users to immediately visualize output and to iteratively rerun single steps in a multi-step PlantCV pipeline, which makes parameters like thresholds or regions of interest much easier to adjust. Once a pipeline is developed in Jupyter, it can then be converted into a Python script that is compatible with PlantCV parallelization see online documentation for detailed instructions on conversion; http: Because ofthe web-based interface and useful export options, Jupyter notebooks are also a convenient method of sharing pipelines with collaborators, or in publications, and teaching others to use PlantCV. Several updates to PlantCV v2 addressed the need to increase the flexibility of PlantCV to analyze data from other plant phenotyping systems. New utilities were added to PlantCV v2 that allow data to be quickly and efficiently exported from the SQLite database into text files that are compatible with R R Core Team, for further statistical analysis and data visualization. Because standards for data collection and management for plant phenotyping data are still being developed Pauli et al. A common approach is to include metadata within image filenames, but because there is a lack of file naming standards, it can be difficult to robustly capture this data automatically. In PlantCV v2, a new metadata processing system was added to allow for flexibility in file naming both within and between experiments and systems. The PlantCV metadata processing system is part of the parallelization tool and works by using a user-provided template to process filenames. User-provided templates are built using a restricted vocabulary so that metadata can be collected in a standardized way. The vocabulary used can be easily updated to accommodate future community standards. Performance In PlantCV v1. For PlantCV v2, the parallelization framework was completely rewritten in Python using a multiprocessing framework, and the use of Matplotlib was updated to mitigate the issues and processor constraints in v1. The output of image files mainly used to assess image segmentation quality is now optional, which should generally increase computing performance. Furthermore, to decentralize the computational resources needed for parallel processing and prepare for future integration with high-throughput computing resources that use file-in-file-out operations, results from PlantCV pipeline scripts one per image are now written out to temporary files that are aggregated by the parallelization tool after all image processing is complete. New functionality PlantCV v2 has added new functions for image white balancing, auto-thresholding, size marker

normalization, multi-plant detection, combined image processing, watershed segmentation, landmarking, and a trainable naive Bayes classifier for image segmentation machine learning. The following are short descriptions and sample applications of new PlantCV functions. White balancing If images are captured in a greenhouse, growth chamber, or other situation where light intensity is variable, image segmentation based on global thresholding of image intensity values can become variable. To help mitigate image inconsistencies that might impair the ability to use a single global threshold and thus a single pipeline over a set of images, a white balance function was developed. If a white color standard is visible within the image, the user can specify a region of interest. If a specific area is not selected then the whole image is used. Each channel of the image is scaled relative to the reference maximum. Auto-thresholding functions An alternative approach to using a fixed, global threshold for image segmentation is to use an auto-thresholding technique that either automatically selects an optimal global threshold value or introduces a variable threshold for different regions in an image. Triangle, Otsu, mean, and Gaussian auto-thresholding functions were added to PlantCV to further improve object detection when image light sources are variable. The triangle threshold method uses the histogram of pixel intensities to differentiate the target object plant from background by generating a line from the peak pixel intensity Duarte,

DOWNLOAD PDF SCIPY V0.19 TUTORIAL

Chapter 2 : Advanced installation instructions â€” scikit-learn documentation

SciPy Tutorial Â¶ Introduction; Basic functions Compressed Sparse Graph Routines (calendrierdelascience.comh) Spatial data structures and algorithms Last updated on.

What are NumPy, SciPy, matplotlib, â€¦? NumPy also provides basic numerical routines, such as tools for finding eigenvectors. SciPy contains additional routines needed in scientific work: The matplotlib module produces high quality plots. With it you can turn your data or your models into figures for presentations or articles. No need to do the numerical work in one program, save the data, and plot it with another program. Using IPython makes interactive work easy. Data processing, exploration of numerical models, trying out operations on-the-fly allows to go quickly from an idea to a result. See the IPython site for many examples. There is a sizeable collection of both generic and application-specific numerical and scientific code, written using Python, NumPy and SciPy. See Topical Software for a partial list. As Python is a popular general-purpose programming language, it has many advanced modules for building for example interactive applications see e. Using SciPy with these is a quick way to build a fully-fledged scientific application. There is no single program that you can start and that gives an integrated user experience. Instead, there are several possible ways to work with Python. The most common is to use the advanced interactive Python shell IPython to enter commands and run scripts. Scripts can be written with any text editor, for instance Emacs , Vim or even Notepad. Some of the packages such as Python x,y mentioned in Installing packages also offer an integrated scientific development environment. Neither SciPy nor NumPy provide plotting functions. There are several plotting packages available for Python, the most commonly used one being matplotlib. One way of getting a handle on the scientific computation tools in Python is to take a look at the following online resources:

Chapter 3 : calendrierdelascience.com_test_split â€" scikit-learn documentation

SciPy TutorialÂ¶. Introduction; Basic functions; Special functions (calendrierdelascience.coml)Integration (calendrierdelascience.comate)Optimization (calendrierdelascience.comze)Interpolation (scipy.

Available functions include airy, elliptic, bessel, gamma, beta, hypergeometric, parabolic cylinder, mathieu, spheroidal wave, struve, and kelvin. There are also some low-level stats functions that are not intended for general use as an easier interface to these functions is provided by the stats module. Most of these functions can take array arguments and return array results following the same broadcasting rules as other math functions in Numerical Python. Many of these functions also accept complex numbers as input. Each function also has its own documentation accessible using help. You can write the function in either C, Fortran, or Python. Look in the source code of the library for examples of each of these kinds of functions. Here is an example of a circular drum head anchored at the edge: The following Cython code gives a simple example of how to use these functions: In the example the function csc. Note in particular that the function is overloaded to support real and complex arguments; the correct variant is selected at compile time. It might help to think of this as analogous to calling a ufunc with an output array: There are two potential advantages to using the Cython bindings: They avoid Python function overhead They do not require the Python Global Interpreter Lock GIL The following sections discuss how to use these advantages to potentially speed up your code, though of course one should always profile the code first to make sure putting in the extra effort will be worth it. Typically this approach works quite well, but sometimes it is more convenient to call a special function on scalar inputs inside a loop, for example when implementing your own ufunc. In this case the Python function overhead can become significant. Consider the following example: Obviously this example is contrived: The point is that if Python function overhead becomes significant in your code then the Cython bindings might be useful. For example, suppose that we wanted to compute the fundamental solution to the Helmholtz equation: The following example shows how we could compute this function in parallel: If the above Cython code is in a file test. Functions not in scipy. To prevent reinventing the wheel, this section provides implementations of several such functions which hopefully illustrate how to handle similar functions. In all examples NumPy is imported as np and special is imported as sc.

Chapter 4 : Special functions (calendrierdelascience.coml) â€" SciPy v Reference Guide

SciPy is a collection of mathematical algorithms and convenience functions built on the Numpy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data.

Chapter 5 : SciPy â€" SciPy v Reference Guide

You may have calendrierdelascience.com file that you want to read into Scipy. Or, you want to pass some variables from Scipy / Numpy into MATLAB. Or, you want to pass some variables from Scipy / Numpy into MATLAB. To save us using a MATLAB license, let's start in Octave.

Chapter 6 : pandas: powerful Python data analysis toolkit â€" pandas documentation

Special functions (calendrierdelascience.coml)Â¶The main feature of the calendrierdelascience.coml package is the definition of numerous special functions of mathematical physics. Available functions include airy, elliptic, bessel, gamma, beta, hypergeometric, parabolic cylinder, mathieu, spheroidal wave, struve, and kelvin.

Chapter 7 : SciPy Tutorial â€" SciPy v Reference Guide

DOWNLOAD PDF SCIPY V0.19 TUTORIAL

Using SciPy with these is a quick way to build a fully-fledged scientific application. How to work with SciPy Â¶ Python is a programming language, and there are several ways to approach it.

Chapter 8 : calendrierdelascience.comze â€" scikit-learn documentation

SciPy Release Notes. SciPy is the culmination of 7 months of hard work. It contains many new features, numerous bug-fixes, improved test coverage and.

Chapter 9 : Introduction â€" SciPy v Reference Guide

Related changesÂ¶. n_iter_ may vary from previous releases in calendrierdelascience.comicRegression with solver='lbfgs' and calendrierdelascience.com Scipy <= , the optimizer could perform more than the requested maximum number of iterations.