PyFigures: A Python-based tool for Rapid Creation of Publication-Quality Scientific Figures

Benoît Aigouy, Benjamin Prud'homme

Title: "PyFigures: A Python-based tool for Rapid Creation of Publication-Quality Scientific Figures"

Benoit Aigouy[1*] and Benjamin Prud'homme[1]

[1] Aix Marseille Univ, CNRS, IBDM, Marseille, France

Keywords : Figure, Scientific Visualization, Automated Layout, Python

Contact details :

Benoit Aigouy*
benoit.aigouy@univ-amu.fr

Author summary:

Scientists often spend hours arranging their data into complex figures for research papers. This process is especially challenging when dealing with microscope images, where maintaining the correct size and shape is crucial for proper interpretation.

We've developed PyFigures, a new tool that acts as a smart assistant for scientists creating figures for their research papers. It's a computer program that helps arrange images and graphs in a neat, organized way.

PyFigures is particularly useful for scientists working with microscope images. It ensures that these images keep their correct proportions and scale, essential for accurate comparisons. The program automatically labels different parts of a figure and can remember layouts, allowing scientists to easily create a consistent style across all their figures.

One of PyFigures' key features is its flexibility. Scientists can use it for simple tasks, like putting a few images side by side, or for more complex projects involving multiple images and graphs. It works with many types of image files and can save figures in various formats suitable for publications or presentations.

Built using Python, a popular programming language, PyFigures can tap into a vast array of existing scientific tools. This allows scientists to create custom graphs or perform complex data analyses within the program.

By automating many tedious aspects of figure creation, PyFigures aims to save scientists time and reduce errors, allowing them to focus more on their research and less on image formatting technicalities. It offers a user-friendly way to create professional-looking scientific figures for both seasoned researchers and students.

To learn more about PyFigures or to try it out, visit our GitHub page: https://github.com/baigouy/PyFigures

Summary

Modern scientific publications often require complex figures with multiple panels and graphs, demanding consistent formatting across diverse data types. Crucial to this formatting is the precise handling of scale, pixel size, and aspect ratio, especially for microscopy and imaging data where these factors are key to accurate representation and comparison of results. Hence, creating figures can be time-consuming and tedious. We present PyFigures, a Python-based software designed to streamline the process of creating scientific figures, particularly for microscopy and imaging data. PyFigures offers automated layout handling, flexible figure structuring, automated lettering, and template creation. It supports multi-dimensional images, various input formats, and multiple export options. The tool's integration with Python[1] and Matplotlib[2] allows for custom data visualization, while its text-based serialization ensures long-term compatibility. PyFigures aims to simplify daily figure creation tasks while providing the flexibility needed for publication-quality images.

Introduction

Visual representation of scientific data is crucial for effective communication of research findings. Well-designed figures can convey complex information and enhance the impact of scientific publications. However, creating high-quality figures that meet journal requirements is often time-consuming and technically challenging for researchers.

Existing tools for figure creation often lack the balance between automation and flexibility needed by researchers. Many resort to using general-purpose graphics software, which, while powerful, lack specialized features for scientific figure creation. A handfull of specialized scientific visualization tools exist, but are often too complex or lack adaptability for diverse research needs.

To address these challenges, we have developed PyFigures **(Figure 1)**, a Python-based toolkit designed to automate and simplify the process of creating scientific figures. PyFigures aims to strike a balance between ease of use for everyday tasks and the flexibility required for publication-quality figures. By leveraging the power of Python and integrating with popular scientific computing libraries, PyFigures provides a versatile platform for researchers across various scientific disciplines.

In this paper, we present the key features of PyFigures, including its automated layout handling, support for multi-dimensional scientific images, and integration with Matplotlib for custom data visualization. We also discuss the tool's technical implementation and compare it with existing solutions.

Software description

Technical Implementation and Dependencies

PyFigures is a Python tool that employs QtPy[3] to build a flexible graphical user interface, compatible with both PySide and PyQt frameworks[4], ensuring cross-platform functionality on Windows, MacOS, and Linux operating systems. Conveniently, these frameworks also offer tools for reading and writing vector graphics files (SVG, PDF).

For image handling, format support and export, PyFigures integrates several powerful libraries: Pillow[5] and SciPy[6] for general images, tifffile[7] for TIFF file support, czifile[8] for Zeiss microscopy files, read-lif[9] for Leica microscopy files, and Python-bioformats[10] for broad microscopy file format support. These libraries collectively enable PyFigures to open, convert, and export a wide array of scientific image formats to and from NumPy[11] arrays.

Figure Creation and Layout Management

PyFigures simplifies and enhances figure creation with its intuitive, streamlined, and distraction-free drag-and-drop interface, significantly improving both visibility and efficiency for users. The software enables users to easily add images to their workspace with a simple drag-and-drop action. The software's innovative layout system allows for the creation of complex, hierarchical structures. Objects can be combined into rows or columns by dragging them onto each other, forming the fundamental building blocks of scientific figures, while images can be easily removed from these structures by simply dragging them to an empty region of the workspace, providing a fluid and intuitive method for both assembling and disassembling figure components.

Recognizing that a simple row-column structure may be too limiting for many scientific figures, PyFigures allows for nested layouts. Rows and columns can contain multiple sub-rows and sub-columns, which can themselves be further subdivided (Figure 1A-D). This recursive structure enables the creation of intricate, multi-level layouts, accommodating even the most complex figure requirements.

For operations that extend beyond the scope of drag-and-drop functionality, such as channel splitting for multi-channel images, PyFigures offers context-sensitive right-click menus. These menus provide quick access to advanced features, ensuring that all tools are readily available when needed.

The synergy of intuitive drag-and-drop functionality, hierarchical layout capabilities, and context-sensitive menus makes figure construction in PyFigures not just efficient, but intuitive, significantly reducing the time and effort required to create complex scientific figures.

Layout Flexibility and Reusability

PyFigures offers layout flexibility and reusability, enhancing the efficiency of scientific figure creation. A key feature is its automatic layout adaptation; when a figure is resized, the software intelligently adjusts the layout to fit the new dimensions, maintaining the relative proportions and arrangements of all elements. This ensures that figures remain visually coherent regardless of size changes.

The tool's user-friendly interface extends to layout manipulation. Users can easily rearrange rows and objects within them using simple keyboard arrow commands, allowing for quick and intuitive adjustments to figure composition.

PyFigures includes an automated lettering system that assigns sequential labels (e.g., A, B, C) to panels within a figure. This feature saves time and ensures consistent labeling across complex multi-panel figures. The labeling system is flexible and can be customized to match specific journal requirements or personal preferences.

To further streamline the figure creation process, PyFigures allows users to convert existing figures into reusable templates. These templates preserve layout, styling, annotations, and other parameters, significantly reducing the time and effort required for repetitive tasks. This feature is particularly valuable for creating standardized figures across multiple datasets, such as in lab reports, grant applications, or journal submissions.

Image Annotation

PyFigures features a robust image annotation system to enhance scientific figure clarity. Double-clicking an image opens the annotation editor, offering tools to refine image presentation. Users can select specific channels, spatial, or temporal dimensions, and edit channel coloring using various look-up tables (LUTs, see also Figure 1E) to optimize visual contrast. The editor supports

maximum intensity projections for multi-dimensional data. Additionally, it facilitates the addition of essential scientific imagery elements such as scale bars, regions of interest (ROIs, see also Figure 1A), and cropping and rotation tools for optimal image composition. Furthermore, users can create insets to showcase magnified views of critical details. All these elements along with custom text labels can be strategically placed at various positions on the image to provide necessary context or explanations. Importantly, PyFigures maintains the layout of these annotative elements automatically, ensuring that the relative positioning and proportions are preserved even if the overall figure dimensions are altered. This automatic layout maintenance significantly reduces the time and effort required for figure revisions, allowing researchers to focus on content rather than formatting. The annotation editor's design balances user-friendliness with a comprehensive feature set, making PyFigures an effective tool for creating informative and visually appealing scientific images.

Save and Export

PyFigures employs text-based serialization for saving figures in a human-readable format that ensures long-term compatibility, allowing figures to remain accessible even as software evolves. The text-based nature of the serialization facilitates easy tracking of changes over time using standard version control systems, and its transparency allows users to inspect and manually edit the file if needed, offering an additional layer of control.

PyFigures offers diverse export options to meet various publication and presentation needs. Users can export high-resolution raster images in formats such as PNG and JPEG, which are ideal for web display and presentations or when file size is a concern. Additionally, the TIFF and vector graphics formats (PDF, SVG) are available for those who require high-resolution images with lossless compression, making it suitable for print and publication purposes. Users can also set the

desired dpi (dots per inch) when saving the image to ensure optimal print quality. These options ensure figures are suitable for both print and digital media, meeting the requirements of different journals and presentation formats.

To streamline collaboration and ensure long-term accessibility, PyFigures includes a project consolidation feature. This functionality collects all input files associated with a figure, including images and data files, and stores them in a single, organized folder. This approach simplifies the sharing of complete figure project with colleagues and ensures all necessary components are preserved for future reference or modifications.

Collectively, these save and export capabilities enhance the usability, shareability, and longevity of figures created with PyFigures, supporting efficient scientific communication and collaboration.

Color Accessibility Features

PyFigures incorporates comprehensive tools to ensure figures are accessible to individuals with color vision deficiencies. The software enables users to split multi-channel images into individual grayscale channels or merge them into color-blind friendly magenta-green pairs of channels. These features enable researchers to create figures that effectively convey information to all viewers, regardless of their color perception abilities.

Moreover, PyFigures offers a color-blind assistant, which can be accessed through the help menu. This tool provides real-time color information for any region under the user's cursor, displaying the grayscale version of each original color. This feature is especially useful for visually impaired users, as it enables them to design figures with a clear understanding of colors and how they will be perceived by others.

Scripting Capabilities and Advanced Data Visualization

PyFigures offers powerful scripting functionality, significantly expanding its capabilities beyond basic image handling. Users can execute custom Python scripts within the tool, enabling a wide range of advanced applications. This feature proves invaluable for loading non-natively supported image formats, applying complex image processing steps, or creating sophisticated data visualizations (Figure 1F).

A key strength of this scripting system is its ability to generate graphs and plots from various data sources, including CSV, TXT, Excel, or SQLite files. Users can adapt their existing Python scripts with minimal modifications to work within PyFigures.

The scripting feature supports multiple formats, including NumPy arrays, Matplotlib figures, and QSvgRenderer, offering flexibility in visualization types. This versatility allows for the creation of diverse visual elements such as genomic data plots, DNA vector diagrams, Matplotlib figures, and virtually any type of visual element or data representation that can be generated through Python scripting, seamlessly integrating them into the main figure.

PyFigures enhances transparency, reproducibility, and traceability by saving and storing custom scripts within the figure file itself. This approach ensures that all data processing and visualization steps are preserved alongside the final figure, allowing for easy verification and replication of results. The software video demonstrations provide practical guidance for utilizing this powerful scripting feature opening up virtually limitless possibilities for figure creation, enabling researchers to produce complex, data-rich visualizations tailored to their specific needs while maintaining a clear record of the processes involved.

Installation and Accessibility

PyFigures can be easily installed as a Python package from PyPI using pip. Detailed installation instructions, source code, documentation and videos are available on our GitHub repository https://github.com/baigouy/PyFigures. Our user-friendly guide provides step-by-step instructions for users with varying levels of technical expertise, making it easy for anyone to get started with PyFigures.

Comparison to Existing Tools

PyFigures offers distinct advantages over existing scientific figure creation tools. This comparison analyzes key aspects of various tools, highlighting PyFigures' unique position in the scientific figure creation landscape.

PyFigures stands out in image handling and layout by offering a flexible approach that accommodates images with varying aspect ratios without requiring cropping. This capability provides users with greater freedom in composing their figures. In contrast, tools like FigureJ[12] and QuickFigures[13] are optimized for working with pre-cropped images, particularly when arranging rows of images with different aspect ratios. Omero-figure[14] occupies a middle ground; while it doesn't mandate cropping, adjusting images with different aspect ratios within its interface is primarily achieved through cropping or manual positioning. This distinction underscores PyFigures' versatility in handling diverse image types and sizes without compromising the original image content.

PyFigures distinguishes itself through its robust Python integration, offering superior graphing capabilities and support for diverse plot types, including matplotlib graph, genomic and DNA plots. This integration provides a significant advantage over other tools in terms of graphing and data visualization, while ensuring extensibility and compatibility with a wide range of scientific computing tools. In contrast, other tools face various limitations. QuickFigures has constrained graphing capabilities due to the limitations of Java plotting libraries. ScientiFig[15] and EzFig[16]

rely on R integration for graphing, which can be challenging to maintain and often yields suboptimal results. FigureJ's graph integration is restricted to a script that forces graph sizes to fit the destination layout, limiting flexibility. Notably, Omero-figure lacks built-in functionality for easy graph addition, requiring users to prepare graphs externally before importing them into the figure. PyFigures' Python foundation not only allows for the creation of diverse plot types but also positions it as a powerful and future-proof solution for scientific figure creation.

PyFigures, ScientiFig, and EzFig share a common foundation in their approach to figure creation, utilizing similar underlying algorithms. However, PyFigures significantly expands upon this shared base by introducing nested image capability, a feature absent in its counterparts. This addition markedly enhances the complexity and sophistication of figures that can be produced. Moreover, while ScientiFig and EzFig imposed limitations on the number and variety of annotation elements, PyFigures removes these constraints.

In terms of export functionality, PyFigures offers raster, SVG, and PDF options, satisfying most scientific and publishing requirements, while QuickFigures supports additional formats like PowerPoint and Adobe Illustrator scripts, which may be advantageous in specific workflows. Although QuickFigures may have a slight speed advantage for certain tasks such as channel splitting, PyFigures' overall versatility for complex layouts gives it an edge. It's worth noting that while ScientiFig and EzFig offered journal guideline support, its practical utility was limited and subsequently omitted from PyFigures. In contrast to these desktop-based tools, Omero-figure stands apart with its server-based architecture. While this approach offers certain advantages, it may present accessibility challenges due to complex installation requirements and the need for specific infrastructure, potentially limiting its accessibility. Finally, a crucial differentiating factor among these tools is their approach to data persistence. PyFigures and Omero-figure utilize text-based serialization for saving figures, a method that ensures better long-term accessibility and cross-platform compatibility. This approach also facilitates easier debugging and manual editing if

necessary. In contrast, EzFig, ScientiFig, FigureJ and QuickFigures rely on binary serialization that can lead to compatibility issues across different versions of the software or different operating systems, and may pose challenges for long-term data preservation.

In conclusion, PyFigures emerges as a local, versatile, user-friendly tool that addresses many limitations of existing software in the scientific figure creation landscape. Its Python foundation not only enhances its capabilities but also positions it as a comprehensive and adaptable solution for the diverse needs of scientific figure creation.

Discussion

PyFigures addresses a critical need in the scientific community for a flexible, efficient, and feature-rich tool for figure creation. By automating many tedious aspects of figure preparation while maintaining the flexibility of general-purpose drawing software, PyFigures allows researchers to focus more on data analysis and interpretation rather than figure formatting. This shift in focus has the potential to accelerate scientific progress and improve the overall quality of visual communication in research publications.

The tool's seamless integration with the Python ecosystem is a particularly powerful feature, especially for researchers already using Python for data analysis. The ability to execute custom scripts within PyFigures enables the incorporation of sophisticated data processing and visualization techniques directly into the figure creation workflow. This integration not only streamlines the research process but also promotes reproducibility by maintaining a direct link between raw data and the final figure.

PyFigures' innovative approach to layout management, with its automatic adaptation to figure resizing and support for complex, nested structures, addresses limitations found in many existing tools. This flexibility allows researchers to create intricate, multi-panel figures that

effectively communicate complex scientific concepts and data relationships. The automated lettering system and reusable templates further enhance consistency and efficiency in figure production.

The tool's robust save and export functionalities, including text-based serialization and support for various output formats, ensure long-term accessibility and compatibility with different publication requirements. The project consolidation feature facilitates collaboration and long-term storage of figure projects, addressing important aspects of data management and sharing in scientific research.

Future development of PyFigures will focus on expanding its capabilities based on user feedback and evolving needs in scientific communication. Potential areas for enhancement include further integration with data analysis pipelines, support for interactive elements in digital publications, and expanded collaboration features for multi-author projects.

In conclusion, PyFigures represents a significant advancement in scientific figure creation tools, offering a powerful combination of automation, flexibility, and integration with the scientific Python ecosystem. By streamlining the figure creation process and providing features tailored to the needs of researchers, PyFigures has the potential to save time, improve consistency, and enhance the quality of scientific publications across various disciplines.

Acknowledgements

This work was supported by CNRS. BA designed the software and authored the manuscript with assistance from ChatGPT and ClaudeAI. Both BA and BP reviewed the manuscript.

References

1. The Python Tutorial. (n.d.). *Python documentation*. Retrieved July 16, 2024, from https://docs.python.org/3/tutorial/index.html
2. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55
3. spyder-ide/qtpy. (2024, July 15). Python, Spyder IDE. Retrieved from https://github.com/spyder-ide/qtpy
4. Qt for Python. (n.d.). Retrieved July 16, 2024, from https://doc.qt.io/qtforpython-6/
5. Pillow. (n.d.). *Pillow (PIL Fork)*. Retrieved July 16, 2024, from https://pillow.readthedocs.io/en/stable/index.html
6. Jones, E., Oliphant, T., Peterson, P., & others. (2001). SciPy: Open source scientific tools for Python. Retrieved from http://www.scipy.org/
7. Gohlke, C. (2024, July 14). cgohlke/tifffile. Python. Retrieved from https://github.com/cgohlke/tifffile
8. czifile: Read Carl Zeiss(r) Image (CZI) files. (n.d.). Python. Retrieved from https://www.lfd.uci.edu/~gohlke/
9. read-lif: A Python module for loading lif file as numpy array. (n.d.). Retrieved from https://github.com/yangyushi/read_lif
10. python-bioformats: Read and write life sciences file formats. (n.d.). Java, Python. Retrieved from http://github.com/CellProfiler/python-bioformats/
11. Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2
12. Mutterer, J., & Zinck, E. (2013). Quick-and-clean article figures with FigureJ. *Journal of Microscopy*, *252*(1), 89–91. https://doi.org/10.1111/jmi.12069
13. Mazo, G. (2021). QuickFigures: A toolkit and ImageJ PlugIn to quickly transform microscope images into scientific figures. *PLOS ONE*, *16*(11), e0240280. https://doi.org/10.1371/journal.pone.0240280
14. ome/omero-figure. (2024, July 11). JavaScript, Open Microscopy Environment. Retrieved from https://github.com/ome/omero-figure

15. Aigouy, B., & Mirouse, V. (2013). ScientiFig: a tool to build publication-ready scientific figures. *Nature Methods*, *10*(11), 1048–1048. https://doi.org/10.1038/nmeth.2692
16. baigouy. (2021, November 25). EZFig. Retrieved from https://github.com/baigouy/EZFig
17. The Cell Image Library. (n.d.). Retrieved July 16, 2024, from http://www.cellimagelibrary.org/home

Figures

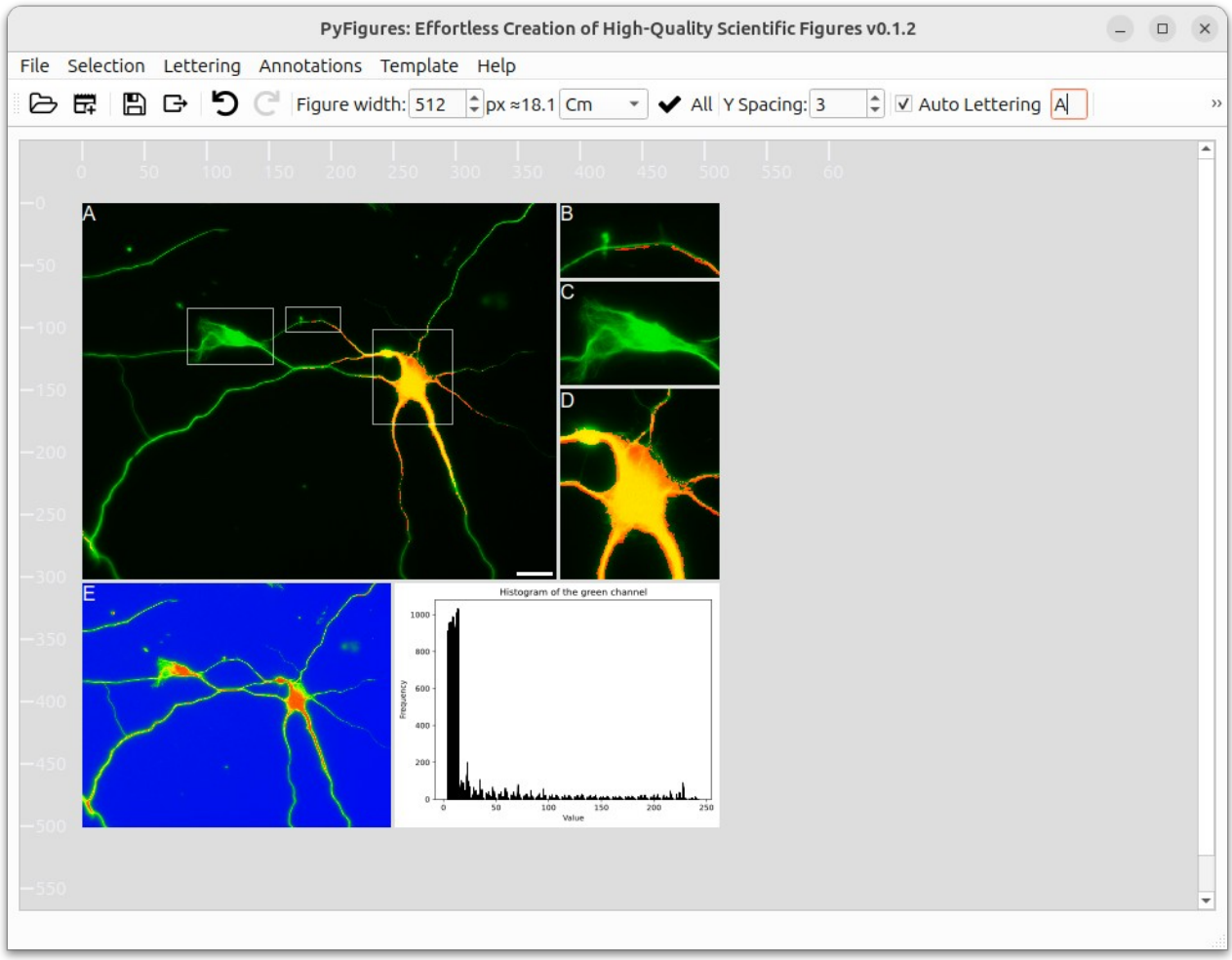

Figure 1: The PyFigures software interface

Screenshot demonstrating a figure created using PyFigures. a) Displays an original image from the CellImageLibrary[17]. Regions of Interest (ROIs) have been defined using the PyFigures interface. b-d) Show corresponding crops automatically generated from the ROIs defined in (a). e) Presents the green channel of image (a) with a rainbow lookup table applied by PyFigures, enhancing visual contrast. f) Depicts a histogram of the green channel intensity distribution, generated using PyFigures' built-in Python scripting capability. This figure showcases key features of PyFigures,

including ROI selection, image cropping, channel manipulation, color mapping, and data visualization through scripting.