Matplotlib save figure as pdf

(Mattplib) (is a powerful two-dimensional library for building the Python language. Matplotlib is able to create all sorts of graphs, plots, diagrams, histograms and more. In most cases, matplotlib simply displays a chart in viewport when the .show method is called, but we'll briefly wonder how to save the matplotlib creation for the actual file on the disk. Using matplotlib While the list of matplotlib features is almost limitless, we'll quickly go through how to use the library to create a base chart for your own testing purposes. Like all Python libraries, you need to start by installing matplotlib. We won't go through the installation process here, but there is a lot of information in the official documentation. Once installed, we import the matplotlib library. You probably also want to import a pyplot sub-ambilite, which is something you tend to use to create diagrams and plots when using matplotlib. In: Import matplotlib import matplotlib.pyplot as plt Now to create and display a simple chart, we first use the .plot () method and pass in a few arrays of numbers for our values. In this example, we'll find the number of books we've read in a matter of months. In 2: plt.plot (No 0, 1, 2, 3, 4), No0, 3, 5, 9, 11) We can also add a few axis tags: In 3: plt.xlabel ('Months') plt.ylabel ('Books Read') Finally, we can display the chart by calling .show (): The savefig method with a simple chart under our belts, now we can choose to display the chart in the file rather than displaying it (or both if necessary) using . In the .savefig ('books_read.png') method, the .savefig method requires that the file name be listed as the first argument. This file name can be a complete way and, as shown above, can also include a specific file extension if necessary. If there is no extension, savefig.format configuration is used instead. Additional savefig options In addition to the basic functionality of saving the chart in the file, .savefig () also has a number of useful additional arguments. dpi can be used to customize the file's numerical resolution. Transparent can be installed on True, which causes the background of the chart to be transparent. bbox_inches can be set to change the size of the wrapped window (white space) around the output image. In most cases, if there is no need to limit the box, the bbox_inches hard is ideal. If the bbox_inches is set to hard, then the pad_inches option determines the amount of padding around the image. There are a few additional options for specific cases, but overall this should get you started with easily generating file image output from your matplotlib In our series of tutorials on matplotlib, you learned how to create many different sites and how to customize their design. So far you've looked at the resulting result by phone .show But what if you need to share the story? For example, if you need to send it via email or use it in a presentation? In this tutorial you will learn how to export sites and save them in different file formats. Matplotlib Series Review Line Plot Before you can experiment with different export options, you need a plot to export ☺ For the rest of this tutorial, you can use a simple plot line because it is especially suitable to see the differences in quality: year 2014, 2015, 2016, 2017, 2018, 2019 tutorial_count 117, 111, 110, 67, 29 plt.plot (year, tutorial_count, color #6c3376, linewidth'3) plt.xlabel ('Year') plt.ylabel ('Number of futurestud.io Tutorials') Save as a PDF file If you want to export a graph with matplotlib, you will always call .savefig (way). matplotlib will find out the type of file based on the file path traveled. For example, if you want to keep the above plot in the PDF file: plt.savefig ('line_plot.pdf') this will keep the plot in line_plot.pdf. You can view the entire weekend files here. Save as an SVG file If you want to save the plot as an SVG file instead, you use the same method .savefig (way), but change the file ending at .svg: plt.savefig ('line_plot.svg') Both PDF and SVG are vector file formats and keep the plot in excellent quality. However, some programs don't easily support these modern formats (looking at you, PowerPoint) and require you to export sites as images. Fortunately, this is not a problem with matplotlib. Save as a PNG file you hopefully understood the pattern and can guess how to export the plot as a PNG file. Just go the way to .savefig () with png file ending: plt.savefig ('line_plot.png') The quality of all images based on file formats, such as PNG or JPG, will come with some loss of quality. You can increase (or reduce) the quality of the site by installing dpi. For example, if you want to create a better PNG export: plt.savefig ('line_plot_hq.png', dpi-300) If you look closely, it will make the plot much more enjoyable. However, keep in mind that it also tripled the file size! If you want to export sites to the image format, there is a trade-off between the quality and size of the file. Color Export Options in PNG has one advantage over JPG. PNG allows transparent colors. By default, matplotlib creates plots on a white background and exports them as such. But you can make the background transparent by passing the transparent savefig method (the) plt.savefig ('line_plot_hq_transparent.png', dpi'300, transparent)Truth) It can make the plots look much nicer on the non-white background. In general, the order of the passed parameters does not matter. You can install them as you want. Save JPG File Final Export Options you should know about is jpg files, which offers better compression and therefore smaller file sizes on some sites. plt.savefig ('line_plot.jpg', dpi-300) JPG JPG offers some additional export options that are only available for files .jpg. For example, quality (default 95), optimization (default: false) and progressive (default: false). plt.savefig ('line_plot.jpg', dpi-300, quality -80, optimize -Truth, progressive) With the options included above, the file size decreases by 50% while maintaining quality at a similar level. What export format to use? Exports as SVG or PDF vector files are generally preferable to PNG or JPG-based bit cards because they are richer formats, usually providing higher quality sites along with smaller file sizes. If you want to see the impact of different data formats, zoom in close to the storyline. With the PDF format and SVG, it will still be a smooth, straight line (left). However, with PNG or JPG, it will be clunky (right). In this tutorial you learned how to export files in different file formats and some of the options for plot quality control. All code and sample exports are available for easy copying and insertion and a closer look at the GitHub repository. Do you have questions or reviews where this series should go? Let us know on Twitter @futurestud_io or leave a comment below. Enjoy the conspiracy and make it rock! The mentioned resources are a GitHub repository with a code sample and a sample of output files matplotlib.pyplot.savefig (args, kwargs) Save the current figure. Signature call: savefig (fname, dpi'None, facecolor'w'w', edgecolor'w', orientation,'portrait,'portrait', papertype'None, format 'None, transparent'False, bbox_inches 'None, pad_inches'0.1, frameon'none, metadata-None) Output formats depend on the backend used. Options: fname : str or PathLike or file-like object path, or Python file-like object, or perhaps some backend-dependent objects such as matplotlib.backends.backend_pdf. PdfPages. If the format is not installed, the output format is withdrawn from the fname extension, if any, and from rcParamssavefig.format - png otherwise. If the format is set, it determines the output format. Therefore, if fname is not a trajectory or has no extension, be sure to specify the format to make sure that the correct backend is being used. Other options: dpi : 'figure' - resolution at points per inch. If not, the default is rcParamssavefig.dpi - figure. If 'digit', uses the value of the figure's dpi. quality : No 1 scalar 100 Image quality, on a scale from 1 (worst) to 95 (best). It only applies if the jpg or jpeg format is ignored differently. If not, rcParams savefig.jpeg_quality - 95 (95 by default) by default. Values above 95 should be avoided; 100 completely disables the JPEG quantitative evaluation stage. optimization : bool If True, points to that the JPEG coder has to make an extra pass over the image to select the optimal encoder settings. It only applies if the jpg or jpeg format is ignored differently. Is a false default. Progressive : bool bool However, it indicates that this image should be stored as a progressive JPEG file. It only applies if the jpg or jpeg format is ignored differently. Is a false default. facecolor : color specification or not, not necessarily the face of the color of the figure; if not, then by default rcParamssavefig.facecolor - white. edgecolor : color specification or no, not necessarily the edge of the shape; if not, then by default rcParamssavefig.edgecolor - white orientation : landscape, portrait Is currently supported only by postscript. Papertype: str One of 'letters', 'legitimate', 'executive', 'book', 'a0' via 'a10', 'b0' via 'b10'. Supported only for postscript output. format : str File format, for example, 'png', 'pdf', 'svg', ... Behavior, when not specified, is documented under fname. Transparent : bool If true, axis patches will all be transparent; The shape patch will also be transparent if the face is not indicated through the kwargs. This is useful, for example, to display a story on top of a colored background on a web page. The transparency of these patches will be restored to their original values after the release of this feature. bbox_inches: str or Bbox, extra Bbox in inches. Only the provided part of the figure is saved. If hard, try to figure out the tough bbox shapes. If not, use savefig.bbox pad_inches: scalar, extra padding around the figure when bbox_inches is tight. If not, use savefig.pad_inches bbox_extra_artists: Artist list, additional list of additional artists to be considered when calculating a tight bbox. metadata : dictation, additional pairs of keys/values to store image metadata. Supported keys and default depend on the image format and backend: 'png' with Agg backend: See the metadata of the print_png. 'pdf' with backend pdf: See the metadata of The PdfPages. 'eps' and 'ps' with PS backend: only 'Creator' is supported. pil_kwargs: dict, additional additional keyword arguments that are passed on to PIL. Image.save while retaining the figure. It only applies to formats that are stored with a cushion, i.e. JPEG, TIFF and (if the keyword is tuned to non-no value) PNG. Png. matplotlib save figure as png. matplotlib save figure as image. matplotlib save figure as jpg. matplotlib save figure as eps. matplotlib save figure as vector. matplotlib save figure as html. matplotlib save figure as bytes. matplotlib save figure as png example