Python for data science cheat sheet matplotlib pdf

Overtime you will be able to create plots like this easily. Here is a Zero to Hero cheat sheet to create a plot using matplotlib Pandas plotting library. It's not an inclusive cheat sheet but you have to feel confident enough with your beginner's plotting ability once you've passed it two or three times. Our first plot Let created the simplest plot. Then see what modifications we can make. The plot below takes only three inputs for the x-axis and the y-axis as any array and plots them. Pay attention to the code below each image determining how each plot is created. If you code together be sure to add plt.show() at the end of your code to see your plot. Go ahead and duplicate what you see below. The first bracket contains the x coordinates while the second bracket contains the y coordinates. Pay attention to the first code block we initialize our library as needed while our second code block creates our plot. Next we will give our plot titles as well as labels for the x-axis and y-axis. We can also resize our charts. See below the image for additional code that we will add to our existing code block or any variation to our code to change the visualization.plt.figure(figsize = (15 , 5) Previously using the plt.plot that we passed in two arguments. One array for x coordinates and another for y coordinates. Note that the default color and appearance for the point we plotted is the blue line. We can skip the third argument that allows us to change the color of our lines and/or turn our lines into points. 'g' to green 'r' to red. 'go' for green dots and 'ro' for red dots.plt.plot([12 , 20 , 35] , [12 , 20 , 24] , 'g') - - — — — — try both - — — — — — — — - -plt.plot([1 2 , 20 , 35] , [12 , 20 , 24] , 'r')plt.plot([12 , 20 , 35] , [12 , 20 , 24] , 'yo' - - - — — — — — — — ——————————— Both— — — — — -plt.plot([12 , 20 , 35] , [12 , 20 , 24] , 'go') It is possible to plan several data sets in the same plot by passing several arguments in the plot( ) methodx = np.arange(1 , 5) y = x**3plt.plot([12 , 20 , 35] , [12 , 20 , 24] , 'go' , y, x , 'r^')np.arange returns an evenly located value at a specified interval. For the np.arange plot we gave us four x values then set y equal to the cubed value of each x value. Now let's create multiple plots in one number. This can be done using the subplot() method. The subplot method takes three nrows(), ncols() and index arguments. This argument shows the number of rows of column numbers and the number of index subplots. Let's create an image with two subplots side by side. We want our figure to look like a picture below.plt.figure(figsize = (15 , 5))plt.subplot(1 , 2 , 1) - — — 1 , 2 , 1 indicates 1 1 2 column positions 1 plt.plot([1 , 2 , 3 , 4] , [1 , 4 , 9 , 16] ,'go')plt.title('1st Subplot') plt.subplot(1 , 2 , 2) — — — 1 , 2 , 1 indicates 1 row 2 positions column 1plt.plot(x , y , 'r^')plt.title('2nd Subplot')plt.suptitle('My sub-plot')If we prefer to have our plot in vertical rows all we have to do is change the plt.subplot argument () to (2 , 1 ,1) and (2, 1, 2). 2 rows, 1 column, position 1 or position 2.Making two subplots is pretty easy but when we try and make more subplots the above methods can be tedious. To overcome this we can use the method plt.subplots(). This method creates two objects (image and axis) and allows us to assign them arbitrary variable names for (fig & axe). We then assign variables to a subplot method that shows the number of rows, the number of columns, and the size of the image. (this points to how many plots we want to create).x = np.arange(1 , 5)y = x**3fig, axe = plt.subplot (nrows = 2 , ncols = 2 , figsize = (6 , 6) ax[0 , 1].plot([1, 2, 3, 4] , 4 , 9 , 16] ,'go')ax(1 , .set_title 0].plot(x , y , 'r^') ax(0 , 1] .set_title('Squqres') ax. I took this data science cheat sheet from DataCamp. This cheat sheet is especially useful for everyone who wants to learn data science using Python. On this page, I ordered a cheat sheet according to the steps to be taken to study data science using Python. Notebook Python Jupyter: Jupyter is very helpful. I highly recommend to use Jupyter notebook. Here jupyter notebook syntax cheat sheet. If you're unfamiliar with Jupyter Notebook, I have some old posts that might be useful to you: 1) setup anaconda 2) understand python libraries for data science. Basic Python data type: If you're unfamiliar with data types in Python. You must first understand what kind of data is in Python and how to use that type of data (for example, strings, int, float, bool, including how to manipulate lists, arrays, and dives). Here's a Basic Python cheat sheet for data science. Numpy Tutorial: Once you understand Python Basic, you can start learning Numpy, one of the most powerful libraries in Python to handle large arrays/matrices. I highly recommend to understand Numpy first before you go to Pandas/Scipy/or Scikit-learn. Here's numpy cheat sheet. Tutorial Pandas: Numpy is a very powerful library, but it is too difficult to manipulate our data using this library, especially if our data is in the form of tables. If you've ever used an Excel spreadsheet and found it very easy manipulate your data in excel tables. Therefore, that's why Python has Pandas. Pandas Library makes you feel very very to manipulate your data. Here are 1) Basic Pandas Python cheat sheets, and here 2) for more advanced data manipulation in Pandas (e.g., merging, joining, concat, merging, etc.). Importing data in Python: How to import data into your python environment using panda or numpy? Here the cheat sheet imports data in Python (for example, from excel, csv, postgre, etc.). SciPy / Linear Algebra tutorial: Sometimes we forget that Python has a very useful library like SciPy. It seems that if we want to learn data science, we can not escape from machine learning. In fact, we have to understand linear algebra to go there. SciPy is a linear algebra library in Python. If you want to learn deep learning for example (that is, image classification), you will deal with a large matrix of your images and you need to perform a lot of operations on your matrix. That's why we need SciPy. Here's a Cheat sheet of the SciPy library in Python. Machine Learning Library in Python: Have you heard of linear regression, logistic regression, bayesian classification, SVM, Decision Tree, and other machine learning algorithms? It's easy to call those functions into your Python environment. Just import the scikit-learn and everything is ready. Here is the Scikit-Learn Python library cheat sheet. Data Visualization/Plotting in Python: Data visualization is one of the most important parts of data analysis. Normally we would plot our results in the form of data visualizations to gain insights from our data sets. Python supports many data visualization libraries, the basis of which is Matplotlib, here matplotlib library cheat sheet. If you need a more beautiful or more sophisticated visualization, you can use 1) the Seaborn library (here seaborn library cheat sheet) or 2) the Bokeh library (here the Bokeh library cheat sheet). Advanced Python: There are three data science cheat sheets for python advance, 1) NLP using Python (here Python NLP cheat sheet) ; 2) Deep learning using Hard Python (Hard deep learning cheat sheet); 3) Data Science Python for Business (here's the Data Science for Business cheat sheet). Data Science CheatSheet Summary: Summary of many things including machine learning, data science, etc. Here. I hope this page can help everyone who wants to start learning data science using Python. Cheat sheet Coding is a lot about remembering: functions, arguments, technicians ... We all lose a lot of time remembering what we already know, and that is one of the reasons this website exists. This section provides a cheat sheet. These pictures have been done by a lot of different people that I try to admit as best I can. If you believe there is a reference please contact me. Thanks to all these people who made our lives easier! If you have a cheatsheet you want to share, don't hesitate! Color and shapes Palette diverging Categorical palette Sequential palette #121 Customize chart #131 Seaborn #196 Available Bookmark shapes Thanks to Datacamp for this cheat sheet. You may be interested in their Intro to Data Visualization which involves a lot of Seaborn. Seaborn Cheat Sheet by Datacamp Matplotlib Thanks to Datacamp for this cheat sheet. They also have a good online course in Matplotlib. Panda Thanks to Datacamp for this cheat sheet. Pandas Cheat sheet by Datacamp Photo by Giorgio Trovato on UnsplashMatplotlib is a plotting library for the Python programming language. Matplotib's most widely used module is Pyplot which provides an interface like Matlab but instead, it uses Python and it is open source. In this note, we will focus on the basic Matplotlib to help visualize our data. This is not a comprehensive list but contains a common type of data visualization format. Let's look forward to it! Structure of this record:Anatomical Matplotlib FigureStart with PyplotChart TypesAnatomy of a Figure (Image by Author)An image contains the entire window in which the plotting occurred. Axe: This is what we think of as a plot. Each Axis has a title, label x, and y-label.ote: We can have more than one Axis in the image that helps in building multiple plots. We can have more than one Axe in the picture which helps in building multiple plots. (Image by Author) Axes are number lines like objects and help generate graph borders. Each axis has an x-axis and an y-axis to plot. A bug is a marker that indicates a data point on an axis, that is, a value used to show a specific point on a coordinate axis. These values can be numbers or strings.

Every time we plot a graph, the axis adjusts and picks up the default bug. Matplotlib default ticks are generally sufficient in common situations but not at all optimal for each plot. The spine to the graph is the edge of the graph. It connects axis tick marks and records the limits of the data area. Starting with PyplotPyplot is a Matplotlib module that provides a simple function for adding plot elements such as lines, images, text, etc. to the current axis in the current image. At first, we set up the notebook to plan and import the packages we would use:Images and axes can be created as follows:Create images and axesWe usually use the name of the fig variable to refer to the image instance, and the axe to refer to the instance axis or axis group instance. After we create the axis, we can use the plt.plot function to plot some data. Let's start with a simple plot. Perhaps the simplest of all plots is the visualization of a single function y = 2x.y = 2xIf we want to create a single number with multiple lines, we simply call the plot function several GarisLine Color and StylesThe plt.plot() function takes additional arguments that can be used to determine line color and style. To customize, we can use the keyword color line style, which accepts string strings represents almost any color/style imaginable. Colors and styles can be determined in a variety of ways:Colors and StylesAxes LimitsMatplotlib does a decent job of selecting the default axis boundary for our plot, but sometimes it's nice to have smoother controls. The most basic way to adjust axis borders is to use the plt.xlim() and plt.ylim() methods: Set limitsLabel PlotsSebel The last part of this section, we will briefly look at plot labeling: titles, axis labels, and simple legends. Label plotTypes of PlotsThere are various plots that can be created using Matplotlib pythons. We'll cover the most used plots for data visualization in this section. We use Kaggle's World Happiness Report dataset. I cleaned up the data and merged all the files happiness_rank.csv the files. You can download and clean up the data or simply download the final result here. I recommend that you check my data cleanup code in Github.ScatterA scatter plot is a type of chart that is often used in statistics and data science. It consists of several data points plotted on two axes. Each variable depicted in the scatter plot will have several observations. This can be a very useful chart type whenever we want to see if there is a relationship between two data sets. When using scatter plotsWe use scatter plots to identify data relationships with each variable (i.e., correlations, or trend patterns.) It also helps in detecting outliers in the plot. In machine learning, sebakan plots are often used in regression, where x and y are continuous variables. They are also used in scatter grouping or outside detection. When to avoid scatter plotsScatter plots are not suitable if we are interested in observing time patterns. Scatter plots are used with data or numeric numbers. So, if we have categories like three divisions, five products, etc., the scatter plot won't reveal much. Python ImplementationScatter plot3D Scatterplot3D Scatterplot assists in visualizing three numeric variables in a three-dimensional plot.3D ScatterplotScatter plot with the most suitable linear regression lineWe will apply linear regression with Scikit-learn using LinearRegression. After creating a linear regression object, we can obtain the line that best corresponds to our data by calling the fit method. The scatter plot with the linear regression line of the BarsA diverging bar chart best suited is a bar chart that has marks for some dimension members pointing up or to the right, and signs for other dimension members pointing in the opposite direction (descending or left, respectively).ote:Signs that flow down or left do not always represent negative values. Divergent lines can represent zero, but also used to separate only marks for two-dimensional members. When to use diverging barsWe use diverging bars to see how items vary based on one and visualize the sequence and number of these variances. If our main goal is to compare the trends of each dimension member, a different bar chart is a good choice. When avoiding bar divergingSering To use bar chart diverging is not as easy as comparing values across member dimensions as is the case with grouped bar charts. Python ImplementationWe only use 2019 data as an example. Area Chart Ideas an area chart based on a line chart. Colored areas (areas) show us the development of each variable over time. There are three types of area charts: regular area charts, stacked area charts, and 100% stacked area charts. When to use area chartsWe use area charts to show how parts change completely over time. For example, the happiness score has six generating divisions; we would like to see the contribution of each of these divisions. In addition, if we are interested in the portion produced by each division and not as much as the total number of divisions ourselves, we can use a 100% stacked area chart. This will show the percentage contribution of each division over time. When avoiding area chartsAre not the best options if we want to compare different stock sizes with each other. If you want to show that one share overtook the other; or if the difference between our values is very small, consider a line chart instead. Implementation of The ChartStacked Area ChartRegular Area Chart100% Stacked Area Chart Bar Chart is one of the most frequently used chart types. As the name suggests, a bar chart consists of a series of bars describing the development of variables. There are four types of bar charts: regular bar chart, horizontal bar chart, group bar chart, and stacked bar chart. When to use bar charts Bar charts are great when we want to track the development of one or two variables over time. One chart axis shows the specific categories compared, and the other axis shows the values that are measured. When to avoid bar charts A simple bar chart does not match when we have a single period breakdown of variables. For example, if I wanted to describe a key line of business that contributed to the company's revenue, I wouldn't use a bar chart. Instead, I'll create a pie chart or one of its variations. Python ImplementationBarGroup chartsStacked bar chartsLollipop chartLollipop charts serve the same purpose as bar charts sorted in a visually pleasing way. We use lollipop charts to show the relationship between numeric variables and other numeric or categorical variables. When using a lollipop chart A lollipop chart is often claimed to be useful in comparison a normal bar chart, if we are dealing with a large number of values and when the values are all high, as in the range of 80-90% (from 100%). Then a large set of high columns can be visually aggressive. When to avoid lollipops Our data has unordered bars of very similar lengths – it is more difficult to compare the length of two very similar lollipops than the standard bar. Python ImplementationLollipop ChartHistogramA histogram is a vertical bar chart that describes the distribution of a data set. Histograms are used to display variable distributions while bar charts are used to compare variables. Histograms plan quantitative data with ranges of data grouped into bins or intervals while bar charts plot categorical data.ote: Bar graphs have spaces between columns, while histograms do not. Histograms are great when we want to show the distribution of data we work with. This allows us to group continuous data into bins and therefore, provide a useful representation of where observations are concentrated. Python Implementation Box Plot (mustache plot)A box plot or mustache plot is a way of summarizing a set of data measured at interval scales. This type of graph is used to indicate the form of distribution, its central value, and its variability. Box Plot (Image by Author)When to use box plotWe use box plot in explanation data analysis, showing whether the distribution is skewed and whether there is an unusual observation potential (outlier) in the data set. Box plots are also especially useful when a large number of observations are involved and when two or more data sets are compared. When avoiding box plotsBox plots do not show detailed distributions such as stem and leaf plots or histograms. Our Python ImplementationSuppose has a data set containing the number of Medium member articles read in the first six months of 2020. Box PlotPie ChartsPie charts are a classic way to show group composition. A pie chart is a circular graph divided into slices. The larger the slice is the larger part of the total amount it represents. However, it is generally not recommended to use at this time as the pie portion area can sometimes be misleading. So, when using pie charts, it is highly recommended to explicitly write down a percentage or number for each pie section. When to use pie chartsPie charts are best suited to describe the whole section. When to avoid pie chartsWe cannot use pie charts in situations when we want to show how one or more variables develop over time. Our Python ImplementationSuppose has a data set that contains information about Medium members. We'd like to see the percentage of articles read in the first six months of 2020. TreeMaps Chart A treemap chart is similar to a pie chart and works better without misleading each group's contributions. Map chart allows us to divide the total into hierarchies and then show the internal details of each of these hierarchies. When to use TreemapMebmap charts are often used for sales data, as they capture the relative size of data categories, allowing allow a high-level summary of similarities and anomalies in one category as well as among several categories. When to avoid Treemap chartsInsizing charts does not match when our data is not divided into categories and sub-categories. In addition, when we encode data with areas and intensity of color, our eyes are not good for detecting relatively small differences in any of these dimensions. If our data is such that our audience needs to make precise comparisons between categories, it's even more complicated when categories aren't aligned with common baselines. We must not make our audience do more work than is necessary to understand the charts! Python ImplementationFor example, we use treemap charts to present new users and article views this month. PlotTime-series Time Series graphs can be used to visualize trends in numerical counts or values over time. Because date and time information is continuous categorical data (expressed as a range of values), points are plotted along the x-axis and connected by continuous lines. The purpose of time series analysis is to find patterns in the data and use the data for prediction. Time series graphs can answer questions about our data, such as: How do trends change over time? Python ImplementationTime SeriesKodes in this record are available on Github.That's it! In this note, we learn how to build a data visualization plot using Matplotlib. We can now easily build plots to understand our data intuitively through visualization. Have a good study! Source: xkcdResources:My notes only cover everything I consider to be a basic need for Matplotlib. It takes months, sometimes years to master the skills, so don't stop learning! If you want to learn more about Matplotlib, start with the great link below. Customize ticksMatplotlib tutorialMatplotlib gallery User guideMatplotlib