Cheatography

Python matplotlib Cheat Sheet

by julenx via cheatography.com/168626/cs/35539/

Creación básica de matriz 2x2 de gráficos

fig, ax = plt.subplots(2, 2)	fig = plt.figure()	plt.figure() #Opcional

Creación básica de gráficos

ax[0,0].plot(y1)	fig.add_subplot(2, 2, 1)	plt.subplot(2, 2, 1)
	plt.plot(y1)	plt.plot(y1)
ax[0,1].hist(y2, bins = n)	**fig.add_subplot(2, 2, 2)**	**plt.subplot(2, 2, 2)**
	plt.hist(y2, bins = n)	**plt.hist(y2, bins = n)**

Import

```
import matplotlib.pyplot as plt
import matplotlib as mpl
```

Parámetros de plt.plot()

	fig, ax = plt.subplots()
	g = ax.plot(data)
linestyle = "-" / linestyle = "solid"	g[0].set_linestyle("-")
linestyle = "--" / linestyle = "dashed"	
linestyle = "-." / linestyle = "dashdot"	
linestyle = "." / linestyle = "dotted"	
linestyle = "None" / linestyle = ""	
linewidth = n	g[0].set_linewidth(n)
marker = "o"	g[0].set_marker("o")
markeredgewidth	ancho del borde del marcador
markeredgecolor	color del borde del marcador
markerfacecolor	color de fondo del marcador
markerfacecoloralt	color alternativo para el color de fondo del marcador
markersize = n	g[0].set_markersize(n)

Parámetros de plt.plot() (cont)

Parámetros de plt.plot() (cont)

drawstyle = "steps"	g[0].set_drawstyle("steps")
ax.plot(y, "-.y*", markersize = 15)	En lugar de especificar el color de la gráfica, el marcador a usar y el estilo de línea mediante los parámetros color, marker y linestyle que hemos visto, podemos añadir tras los parámetros x e y (o solo y) el parámetro [fmt] con el mismo objetivo. Este parámetro es una cadena de texto en la que podemos incluir (en cualquier orden) un carácter que indique el color de la línea, otro que indique el marcador a usar y uno o dos más para indicar el estilo de la línea. No es necesario añadir los tres datos.

color	Este parámetro controla el color de la gráfica y acepta gran cantidad de formatos distintos: Uno de los siguientes nombres: 'blue', 'orange', 'green', 'red', 'purple', 'brown', 'pink', 'gray', 'olive' o 'cyan' Uno de los siguientes caracteres repres-entando colores: 'b', 'g', 'r', 'c', 'm', 'y', 'k', 'w' (por ejemplo, "b" representa "blue", azul) Tuplas de tres o cuatro valores entre 0 y 1 representando colores en formato RGB o RGBA, por ejemplo: (0.1, 0.3, 0.2) Una cadena de texto representando un número hexade-cimal correspon-diente a un color en formato RGB o RGBA, por ejemplo #A055CC, o #99CC3377 Un color en formato X11/CSS4 Un nombre de color del xkcd color survey precedido por "xkcd:" (por ejemplo, "xkcd:burnt orange") Cualquier nombre de color web, por ejemplo "-Tomato" o "Cornfl-owerBlue"

Cheatography

Python matplotlib Cheat Sheet

by julenx via cheatography.com/168626/cs/35539/

Parámetros de plt.plot() (cont)

plt.plot(y1, "", y2)	Si se utiliza este parámetro [fmt] es posible mostrar dos o más gráficas en el mismo conjunto de ejes ejecutando la función plot una sola vez
label = "name"	g1[0].set_label("name")
plt.legend()	Activamos leyenda

Función legend

ax.plot(y1, label = "Ene")

ax.plot(y2, label = "Feb")	Creación básica de etiquetas
ax.legend()	Activar leyenda
ax.legend(["Enero", "Febrero"])	Cambiar nombres mostrados de etiquetas
ax.legend(["Enero", "Febrero"], loc = "lower right")	Mostrar leyenda en esquina inferior derecha. Funciona también con números. 0 es la posición más óptima y 10 es el centro.
loc = (0.2, 0.6))	tupla con la posición x e y de la leyenda con respecto al ancho y alto del área de la gráfica

Función legend (cont)

ncol = 2	permite especificar el número de columnas en las que se van a mostrar las etiquetas en la leyenda. En este caso, si tenemos dos etiquetas, se mostrará una al lado de la otra.
fontsize	
shadow	
facecolor	color de fondo
edgecolor	
title	
title_fontsize	

Curvas de nivel

```
X = np.linspace(-1.2, 1.2, 100)
Y = np.linspace(-1.2, 1.2, 100)
X, Y = np.meshgrid(X, Y)
Z = np.abs(0.4 2 - (0.6 - (X 2 + Y 2) 0.5) 2) 0.5
```

```
fig = plt.figure(figsize = (12, 6))
ax = fig.gca(projection='3d')
surface = ax.contour3D(X, Y, Z, cmap = "hot")
fig.colorbar(surface)
plt.show()
```

levels = 20

> escoger el número de niveles a mostrar

zdir = "x"

> modificar el eje de referencia

cmap = "hot"

> Mapa de color

Gráficos de dispersión

```
fig = plt.figure(figsize = (12, 6))
ax = fig.gca(projection='3d')
```

> Alternativa 1

```
ax = fig.add_subplot(projection = "3d")
```

> Alternativa 2

```
fig, ax = plt.subplots(subplot_kw = {"projection":"3d"})
```

> Alternativa 3

```
ax.scatter3D([0,1], [3,4], [0,0])
```

> Hacemos 2 puntos distintos

```
fig = plt.figure(figsize = (12, 6))
ax = fig.gca(projection='3d')
for n in range(3):
 x = np.random.normal(0, (n + 1) * 3, 100)
 y = np.random.normal(0, (n + 1) * 3, 100)
 z = np.random.normal(0, (n + 1) * 3, 100)
 scatter = ax.scatter3D(x, y, z, label = n)
plt.legend()
plt.show()
```

> Ejemplo con etiquetas

Histograma

```
h = ax.hist(y, bins=n, range=(start, finish))
```

> Histograma con n bins, y mostrando solo los valores entre start y finish

By **julenx**
cheatography.com/julenx/

Published 19th November, 2022.
Last updated 2nd December, 2022.

Cheatography

Python matplotlib Cheat Sheet

by julenx via cheatography.com/168626/cs/35539/

Histograma (cont)

v, l, g = h

Un histograma contiene tres elementos, el primero es un array con los valores de las barras/bins (altura)
el segundo es un array que contiene los límites de cada bin en el eje x y el tercero es una lista de objetos gráficos (rectángulos) que representan cada barra

h[2][5].set_facecolor("OrangeRed")

Referenciamos el objeto gráfico de la quinta barra y cambiamos esa barra a naranja

density = True

Mostrar en el eje y la densidad de probabilidad de 0 a 1. Otra opción es False y el valor por defecto es None. v saldría con los valores normalizados al usar True

cumulative = True

determina si el histograma es construido de forma que cada bin incluya sus valores y los anteriores

cumulative = -1

el histograma se construye al revés

orientation = "horizontal"

Mostrar el histograma en posición horizontal en vez de vertical

color

Mostrar valores justo encima de cada bin

```
fig, ax = plt.su bpl ots (fi -
gsize = (10, 8))
v, m, g = ax.hist(y, bins = 20)
for i, rect in enumer ate(g):
       posx = rect.g et_x()
       posy = rect.g et_ hei -
ght()
        ax.t ex t(posx + 0.03,
posy + 30, int(v[i]),
color= 'bl ack', fontsize = 12)
plt.show()
```

Funciones

ax[0,0].set_title("Title")	**plt.title("Title")**
ax[0].set_xticks(rang-e(0,n), minor = name_list)	**plt.xticks(rang-e(0,n), name_list)**

Basic functions

ax.plot(list)

Crear de forma explícita una figura y un conjunto de ejes

Personalización de figuras estilo OO

fig.set_size_inches(width, height)

figsize

fig.set_facecolor("color")

facecolor

fig.set_edgecolor("color)

edgecolor

fig = plt.figure(linewidth = 6)

linewidth (no tiene equivalente)

plt.title(""), plt.xlabel("") y plt.ylabel("")

alpha	transparencia del texto
backgroundcolor	
color	
fontfamily	
fontname	
fontsize	
fontstretch	ancho de la fuente
fontstyle	'normal', 'italic', 'oblique'

plt.title(""), plt.xlabel("") y plt.ylabel("") (cont)

fontvariant	'normal', 'small-caps'
fontweight	'ultralight', 'light', 'normal', 'regular', 'bold', etc.
horizonta-lalignment	'center', 'right', 'left'
verticala-lignment	'center', 'top', 'bottom', 'basel-ine', 'center_baseline'
linespacing	
rotation	ángulo de rotación del texto (en grados)
position	posición x e y del título. Toma (x,y)
x	posición x del texto
y	posición y del texto

Los parámetros position, x e y toman normalmente valores entre 0 y 1, correspondiendo el 0 al extremo izquierdo del espacio ocupado por el conjunto de ejes y el 1 al extremo derecho (o a los extremos inferior y superior, si nos referimos a la y). Pueden tomar también valores negativos o superiores a 1, pero en este caso el texto se mostrará fuera de los límites de los ejes.

Todas las variantes orientadas a objetos toman title.set_..., o ax.set_label("name", parameter)

Límites de los ejes

plt.xlim(start, finish)

ax.set_xlim(start, finish)

ax.get_xlim

> método para leer los límites del eje en ambos lados

nos interesa mostrar solo la parte de la gráfica situada entre los valores start y finish

MultipleLocator (OO)

from matplotlib.ticker import MultipleLocator

> Es otra forma de establecer las marcas principales y secundarias

ax.xaxis.set_major_locator(MultipleLocator(10))

ax.xaxis.set_minor_locator(MultipleLocator(1))

> queremos que una gráfica muestre en el eje x las marcas principales cada 10 puntos y las marcas secundarias cada punto. Las marcas principales tienen etiquetas y las secundarias no.

Grid

ax.grid()

> Muestra el grid

alpha

> grado de transparencia

color

> color de las líneas del grid

linestyle

> estilo de las líneas ('-', '--', '-.', ':', etc.)

linewidth

> ancho de las líneas

Grid (cont)

b

> booleano que indica si se muestran o no las líneas

which

> puede tomar los valores 'major', 'minor' o 'both', indicando si estamos configurando el grid correspondiente a las marcas principales, a las secundarias o a ambas (habiendo usado MultipleLocator antes)

axis

> puede tomar los valores 'both', 'x' o 'y', y que indica a qué eje vamos a aplicar la configuración

Gráficos estáticos vs. dinámicos

%matplotlib inline

> provoca que las imágenes se muestren estáticas dentro del código (comportamiento por defecto en Jupyter)

%matplotlib notebook

> provoca que las imágenes generadas sean interactivas (y se muestren también insertadas en el cuaderno jupyter)

Scatter plot

ax.scatter(x, y)

> Scatter plot

pd.unique(data.species)

colors = { "setosa": "Crimson", "versicolor": "RoyalBlue", "virginica": "DarkSeaGreen" }

species_color = data.species.map(colors)

Scatter plot (cont)

ax.scatter(data.sepal_length, data.sepal_width, color = list(species_color))

> Ejemplo de cómo crearíamos un scatter plot con diferentes colores dependiendo de una tercera variable.

mostrar tres etiquetas. s=dotsize

```
fig, ax = plt.subplots()
for species in set(data.species):
  ax.scatter(
      data.sepal_length[data.species ==
species],
      data.sepal_width[data.species ==
species],
       s = 30,
      c = colors[species],
      label = species
  )
plt.legend()
plt.show()
```

Gráficos en 3D

from mpl_toolkits.mplot3d import Axes3D

```
X = np.arange(-10, 10, 0.25)
Y = np.arange(-10, 10, 0.25)
X, Y = np.meshgrid(X, Y)
Z = np.sin(np.sqrt(X2 + Y2))
```

```
fig = plt.figure(figsize = (12, 6))
ax = fig.gca(projection='3d')
surface = ax.plot_surface(X, Y, Z, cmap = "-
coolwarm")
fig.colorbar(surface)
plt.show()
```

> Ejemplo

surface = ax.plot_wireframe(X, Y, Z, rcount = 25, ccount = 25)

> Wireframes, especificando el número de muestras de los datos a utilizar en cada dirección (opcional)

Python matplotlib Cheat Sheet

by julenx via cheatography.com/168626/cs/35539/

Gráficos de líneas con barras de error

fig, ax = plt.subplots()

ax.errorbar(x, y, yerr = 1)

plt.show()

ecolor

> define el color de las barras de error

elinewidth

> define el ancho de las barras de error

capsize

> define el ancho de los topes que limitan cada barra de error

marker = "o"

> Marcar cada punto

y_error = y * 0.25 + 0.5

> Ejemplo de error con porcentajes

Gráficos circulares

sns.set()

fig, ax = plt.subplots()

g = ax.pie(data)

labels = list

> Lista de etiquetas para cada valor

colors = list

> Lista de colores para cada valor

shadow = True

> Añadir sombreado

explode = (0.2, 0, 0, 0)

> Un valor para cada etiqueta. Hace que salgan separadas del resto.

labeldistance = 1.1

> Distancia del texto de las etiquetas respecto al gráfico

Gráficos circulares (cont)

autopct = '%.1f%%'

> Mostrar porcentajes encima de los slices del gráfico

pctdistance = 0.7

> Distancia de los porcentajes del centro

plt.show()

Personalización de figuras estilo MatLab

fig = plt.figure(figsize = [width, height])

> especificar el tamaño de la figura en pulgadas

fig = plt.figure(facecolor = "color")

> color del fondo de la figura

fig = plt.figure(edgecolor = "color", linewidth = n)

> damos color al borde de la figura (por defecto es de color blanco) y con el parámetro linewidth definimos el ancho de dicho borde (por defecto es 0)

plt.suptitle('Title', fontsize=n)

fig.suptitle('Title', fontsize=n)

> Título y tamaño de letra de la figura, no de las gráficas que vayan dentro.

Poner una gráfica encima de otra

fig, ax = plt.subplots(2, 2, sharex = True, sharey = True)

> todos los ejes de los diferentes conjuntos de ejes compartirán las mismas propiedades

fig = plt.figure()

ax1 = plt.axes()

ax1.plot(y1)

ax2 = plt.axes([0.0, 0.0, 0.5, 0.5])

Poner una gráfica encima de otra (cont)

ax2.plot(y2, color = "red")

plt.show()

> Si hacemos esto, la gráfica roja saldrá encima de la otra.

Marcas de ejes

plt.xticks(range(start, finish, intervals), xtick_labels)

ax.set_xticks(range(start, finish, intervals), xtick_labels)

> Si intervals no se indica, va de 1 en 1. xtick_labels (opcional) es una lista cuyos valores reemplazarán los números. si la lista de etiquetas es más larga que la lista de marcas, se ignoran las etiquetas extras. Y si es más corta, las marcas para las que no haya etiqueta se muestran sin ella.

minor=True

> Si usamos este parámetro, se añaden las nuevas marcas que indiquemos sobre las marcas que había antes

ax.set_xticklabels(xtick_labels)

> También podemos indicar únicamente las etiquetas con esta función

Relación de aspecto

ax.set_aspect("aspect")

> "aspect" puede ser "auto" (por defecto), "equal" o un número. Un 2 hará que la distancia entre los puntos del eje x sea el doble que la de y

relación de tamaño entre una unidad del eje x y una unidad del eje y

Python matplotlib Cheat Sheet

by julenx via cheatography.com/168626/cs/35539/

Estilos

plt.style.available

Consultar estilos

plt.style.use(style)

Cambiar estilo

plt.style.use("default")

Estilo default que no aparece en la lista de estilos

mpl.rcParams["figure.dpi"] = 72

una vez que has usado otro estilo, aun volviendo al estilo "default" verás que los tamaños por defecto de las figuras ha aumentado. los puntos por pulgada (dpi) de las figuras, que por defecto toma el valor 72, pasa a valer 100 al activar el estilo "default". Esta es la solución.

Mapas de color

https://matplotlib.org/stable/tutorials/colors/colormaps.html

Adición de otros elementos gráficos

ax.text(x, y, "text")

x e y son las coordenadas según los ejes usados. Admite los atributos de title.

transform = ax.transAxes

Parámetro que cambia el tipo de coordenadas. 0,0 será la esquina inferior izquierda y 1,1 será la esquina superior derecha de la gráfica en la que aparece el texto

Adición de otros elementos gráficos (cont)

transform = fig.transFigure

Parámetro que cambia el tipo de coordenadas. 0,0 será la esquina inferior izquierda y 1,1 será la esquina superior derecha de la figura

ax.annotate("Máximo local", (x1, y1), (x2, y2), arrowprops = dict())

Mostrar un texto en x2 y2 con una flecha que apunta a x1 y1. El objeto dict almacena las propiedades de la flecha y es necesario crearlo para que aparezca.

plt.axhline(n)

Añade una línea recta horizontal. n es la posición según el eje y usado en la gráfica. Admite los mismos parámetros que plot.

plt.axvline(42);

Añade una línea recta vertical. n es la posición según el eje x usado en la gráfica. Admite los mismos parámetros que plot.

from matplotlib.lines import Line2D

l = Line2D([x1, y1], [x2, y2])

ax.add_line(l)

Añade una línea recta que va desde un punto a otro según el eje usado en el gráfico. Admite los mismos parámetros que plot

from matplotlib.patches import Rectangle, Circle, Ellipse, Polygon

rect = Rectangle((x, y), width, height)

ax.add_patch(rect)

Crear un rectángulo. tupla con las coordenadas x e y de la esquina inferior izquierda, la anchura y la altura del rectángulo.

Adición de otros elementos gráficos (cont)

circ = Circle((x, y), width, height)

ax.add_patch(circ)

Crear círculo/elipse. El ancho y alto son el radio.

polygon = Polygon([[x1, y1], [x2, y2], [x3, y3]])

ax.add_patch(polygon)

Crear polígono. En el ejemplo se crear un triángulo donde las tres tuplas son la posición de los tres vértices.

2D Histogram

plt.style.use("default")

plt.hist2d(x, y)

bins = (x,y)

If int, the number of bins for the two dimensions (nx=ny=bins). - If `[int, int]`, the number of bins in each dimension (nx, ny = bins).

plt.colorbar()

Añadir barra de colores

Gráficos de barras

fig, ax = plt.subplots()

ax.bar(x, y)

plt.show()

y es la altura de cada barra

ax.bar(x, y1, label = "Producto A")

ax.bar(x, y2, bottom = y1, label = "Producto B")

Cheatography

Python matplotlib Cheat Sheet

by julenx via cheatography.com/168626/cs/35539/

Gráficos de barras (cont)

ax.bar(x, y3, bottom = y1 + y2, label = "Producto C")

> Con el parámetro bottom se especifica que una barra se pondrá encima de la otra. Si no se especifica, saldrá una cubriendo a la otra.

width = 0.3

> Cambia el ancho de las barras

ax.barh(y, x)

> Gráfico de barras horizontales

ax.barh(y, x1, label = "Producto A")

ax.barh(y, x2, left = x1, label = "Producto B")

ax.barh(y, x3, left = x1 + x2, label = "Producto C")

ax.legend(loc = (1.1, 0.8))

plt.show()

> Apilar barras en gráfico horizontal

ax.barh(y, x, height = 1)

> Cambiar ancho de barras en gráfico horizontal

Boxplot

plt.boxplot(data, labels = tips.day.unique())

Violinplot

plt.violinplot(data)

bw_method=1

sns.kdeplot(data)

Curvas de nivel

plt.style.use("default")

mpl.rcParams["figure.dpi"] = 72

```
def f(x, y):
  return np.sin(x) * 2 + np.cos(5 + x y) + 2 * np.cos(x)
```

x = np.linspace(0, 5, 100)

Curvas de nivel (cont)

y = np.linspace(0, 5, 100)

X, Y = np.meshgrid(x, y)

plt.contour(X, Y, Z)

> Curvas de nivel sin rellenar

plt.contourf(X, Y, Z)

> Curvas de nivel rellenas

plt.colorbar()

> Mostrar barra de colores

plt.show()

levels = 15

> determina el número de curvas a mostrar

cmap = "coolwarm"

> mapa de color a usar

alpha

> grado de transparencia

linewidths = 4

> ancho de los contornos

linestyles = "dotted"

> estilo: solid, dashed, dashdot o dotted

```
fig, ax = plt.subplots()
g = ax.contourf(X, Y, Z, levels = 15)
fig.colorbar(g)
plt.show()
```

> Ejemplo con fig y ax

fraction = 0.20

> Porcentaje de la figura que se dedicará a la barra de color

aspect = 3.7

> Proporción alto-ancho de la barra de colores

orientation = "horizontal"

> Cambiar orientación barra de colores