

Matplotlib Cheat Sheet

EPFL CS 328
Numerical Methods for Visual Computing

(Version 1)

Importing Matplotlib Packages

Common magic commands for Jupyter:

```
# Enable inline backend (plots within notebooks).
%matplotlib inline
# Enable interactive inline backend.
%matplotlib notebook
```

Common import statements:

```
# Package used for state-machine usage of Matplotlib
from matplotlib import pyplot as plt
# Reading images.
import matplotlib.image
# Plotting 3D plots.
import mpl_toolkits.mplot3d
```

Plotting Lines Pipeline

Preparing the data:

```
x = np.linspace(-2.0 * np.pi, 2.0 * np.pi, 1000)
y = np.sin(x)
```

Limiting displayed axes ranges:

```
plt.xlim([-3.0, 3.0]) # [from, to]
plt.ylim([-1.5, 1.5])
```

Adding title, axes labels, grid:

```
plt.title('My wonderful plot')
plt.xlabel('T')
plt.ylabel('amplitude')
plt.grid()
```

Plotting:

```
plt.plot(x, y, label='sine wave') # Linear axes.
plt.semilogx(x, y) # Logarithmic X axis.
plt.semilogy(x, y) # Logarithmic Y axis.
plt.loglog(x, y) # Logarithmic both axes.
```

Legends:

```
# Legend uses labels set in plotting statements.
plt.legend(loc='lower right')
# loc={'best', 'upper center', ...}
```

Saving plotted image to file:

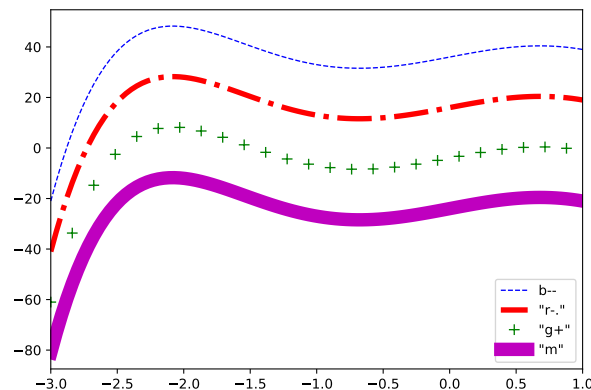
```
# File format inferred from extension (.pdf, .png, ...).
plt.savefig('file.png', dpi=200)
# 'dpi' can be used to set custom resolution.
```

Line Styles

Using (optional) plotting parameters:

linestyle, linewidth, color, marker, markersize, label

```
plt.plot(x, y, color='b', linewidth=1, linestyle='--',
         label='b--')
plt.plot(x, y, color='r', linewidth=4, linestyle='-.',
         label='r-.')
plt.plot(x, y, color='g', linestyle='', marker='+',
         markersize=8, label='g+')
plt.plot(x, y, color='m', linewidth=10, label='m')
```



Histograms

Draw the histogram of an array x :

```
plt.hist(x)
```

Returns three arrays of..

n	histogram values
bins	x-positions of bin edges
patches	patches/rectangle objects drawn in the figure

Common optional parameters:

bins	number of bins to use
normed	if True, mimics a <i>probability density</i> . (The histogram will integrate to 1.)
cumulative	if True, mimics a <i>cumulative distribution</i> .
orientation	{ 'vertical', 'horizontal' }
color	specifies color to be used for bars.
log	if True, histogram axis is set to log scale.

Images

Read image from disk:

```
img = matplotlib.image.imread('path/to/image.png')
```

Plot the image:

```
plt.imshow(img)
```

Optionally show colorbar:

```
plt.colorbar()
```

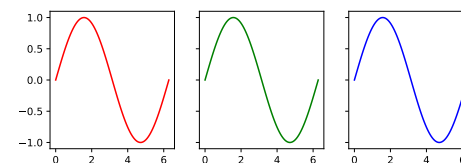
Common optional parameters:

cmap	colormap used for grayscale images. { 'gray', 'hot', 'plasma', ... }
interpolation	{ 'nearest', 'bilinear', ... }

Subplots

Three plots in a 1 x 3 matrix with shared y-axis:

```
fig, ax = plt.subplots(1, 3, figsize=(9, 3), sharey=True)
ax[0].plot(x, y, color='r')
ax[1].plot(x, y, color='g')
ax[2].plot(x, y, color='b')
```



3D Plots

Create 3D figure:

```
fig = plt.figure()
ax = fig.gca(projection='3d')
```

Preparing the data:

```
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
Z = np.sin(np.sqrt(X**2 + Y**2))
```

Plotting the surface:

```
ax.plot_surface(X, Y, Z)
```

Other common plot styles:

ax.plot(..)	3D Line plot
ax.scatter(..)	3D Scatter plot
ax.plot_trisurf(..)	Triangulated mesh data

Interactivity

Easily add interactivity with sliders in Jupyter notebooks:

```
from ipywidgets import interact

@interact(omega=(0, 10, 1)) # min, max, step
def plotSin(omega = 1):
    x = np.linspace(0.0, 2*np.pi, 1000)
    y = np.sin(omega * x)
    plt.plot(x, y)
```