Python for scientific research

Pattern matching and text manipulation

Bram Kuijper

University of Exeter, Penryn Campus, UK

March 3, 2020

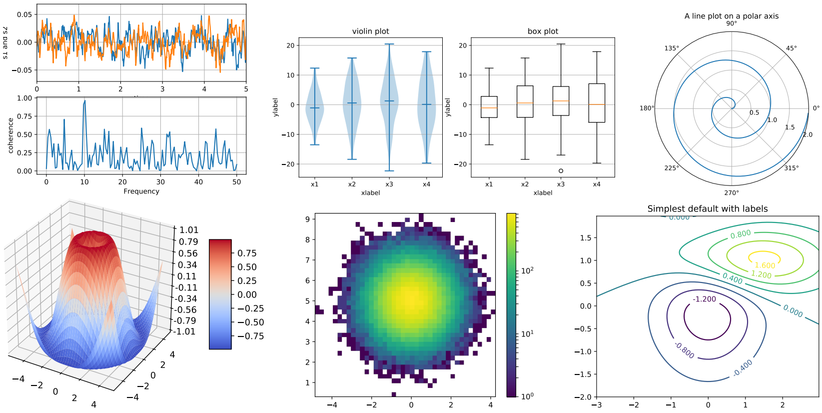
University of Exeter Doctoral College — Researcher Development

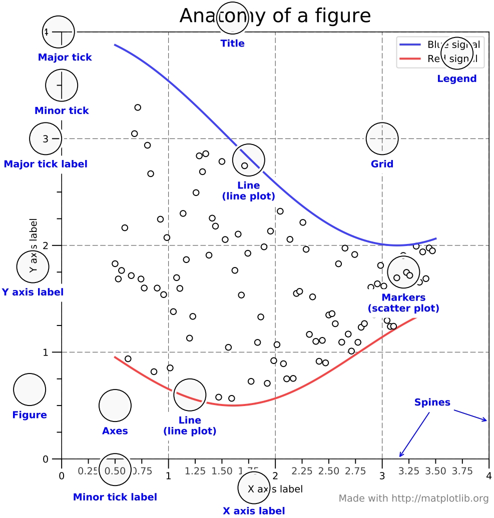
What we've done so far

1. Declare variables using built-in data types and execute operations on them
2. Use flow control commands to dictate the order in which commands are run and when
3. Encapsulate programs into reusable functions, modules and packages
4. Use string manipulation and regex to work with textual data
5. Interact with the file system
6. Number crunching using `NumPy/SciPy`
7. **Next**: Introducing `Matplotlib`, Python's plotting library

Introduction

- Matplotlib is a 2D and 3D plotting library that produces publication-ready scientific figures in most formats (e.g png, eps, svg)

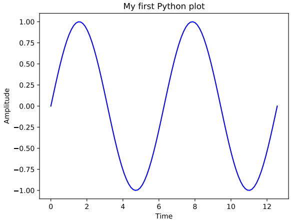




Anatomy of a figure
Title
Major tick
Minor tick
Major tick label
Y axis label
Figure
Axes
Line
(line plot)
Line
(line plot)
Grid
Legend
Blue signal
Red signal
Markers
(scatter plot)
Spines
Minor tick label
X axis label
Y axis label
X axis label
0
1
2
3
4
0.25
0.50
0.75
1.25
1.50
1.75
2.25
2.50
2.75
3.25
3.50
3.75
Made with http://matplotlib.org

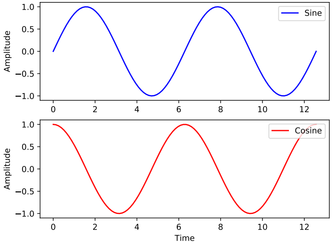
My first plot

```python
import numpy as np
import matplotlib.pyplot as plt

# Generate sinusoidal data
x = np.linspace(0, 4*np.pi, 100)
y = np.sin(x)

# Plot sinusoidal curve
plt.plot(x, y, color="blue")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.title("My first Python plot")
```



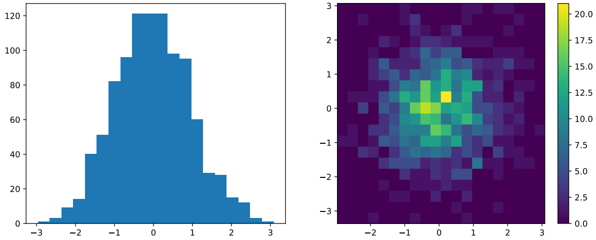
Subplots

```python
plt.subplot(2, 1, 1) # 2 rows, 1 column, first plot
plt.plot(x, np.sin(x), color="b", label="Sine")
plt.ylabel("Amplitude")
plt.legend(loc="upper right")

plt.subplot(2, 1, 2) # 2 rows, 1 column, second plot
plt.plot(x, np.cos(x), color="r", label="Cosine")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.legend(loc="upper right")
```



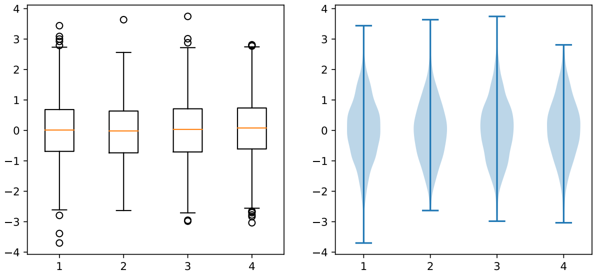
Histograms

```
# Generate 1000 normally distributed numbers
x1 = np.random.randn(1000)
x2 = np.random.randn(1000)

# 1D histogram (x1)
plt.subplot(1, 2, 1)
plt.hist(x1, bins=20)

# 2D histogram (x1 vs x2)
plt.subplot(1, 2, 2)
plt.hist2d(x1, x2, bins=20)
plt.colorbar()
```



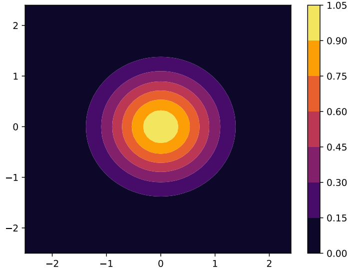
Boxplots and violin plots

```
# Generate four sets of random numbers
data = [np.random.randn(1000) for i in range(4)]

# Boxplot
plt.subplot(1, 2, 1)
plt.boxplot(data)

# Violin plot
plt.subplot(1, 2, 2)
plt.violinplot(data)
```



Contour plot

```python
# import colormaps
from matplotlib import cm

# Generate some x, y, z data
x = np.arange(-2.5, 2.5, 0.1)
y = np.arange(-2.5, 2.5, 0.1)
x, y = np.meshgrid(x, y)
z = np.exp(-x**2 - y**2)

# Contour plot
plt.contourf(x, y, z, cmap=cm.inferno)
plt.colorbar()
```



- Typical structure of 3D data

column_x	column_y	column_z
0.897238	100000	4.06333241929155
0.916328	500	5.1788870872331
0.954509	100909	5.1788870872331
....	...	...

- Data needs to be transformed to a pivot table before plotting. Example:

```
print(z)
# [[3.72665317e-06 6.08307642e-06 9.73288695e-06 ... 1.52642052e-05
#   9.73288695e-06 6.08307642e-06]
#  ...
#  [6.08307642e-06 9.92950431e-06 1.58871492e-05 ... 2.49160097e-05
#   1.58871492e-05 9.92950431e-06]]
```

Working with data and matplotlib

- Typical structure of 3D data

column_x	column_y	column_z
0.897238	100000	4.06333241929155
0.916328	500	5.1788870872331
0.954509	100909	5.1788870872331
....	...	...

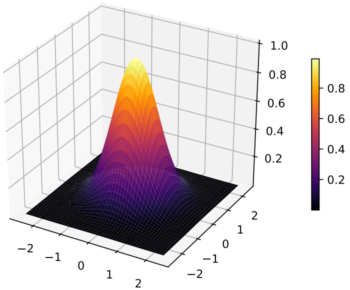
- Data needs to be transformed to a pivot table before plotting. Example:

```
print(z)
# [[3.72665317e-06 6.08307642e-06 9.73288695e-06 ... 1.52642052e-05
#   9.73288695e-06 6.08307642e-06]
#  ...
#  [6.08307642e-06 9.92950431e-06 1.58871492e-05 ... 2.49160097e-05
#   1.58871492e-05 9.92950431e-06]]
```

- Transform to pivot table with `pandas.pivot_table()` (see later)

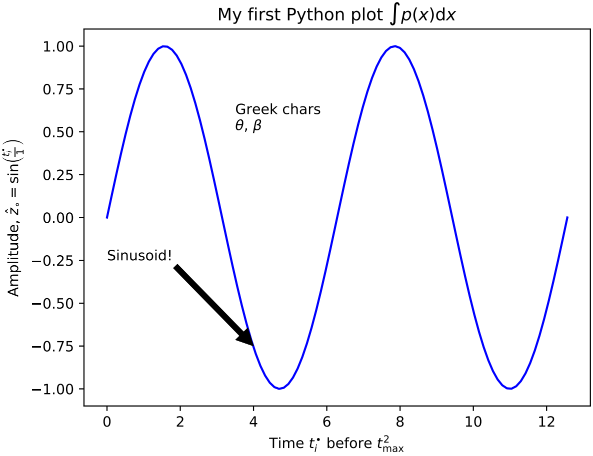
3D surface

```python
from mpl_toolkits.mplot3d import Axes3D

# Create figure amenable for 3D plotting
hFig = plt.figure()
hAx = hFig.gca(projection="3d")

# Plot the surface
hSurf = hAx.plot_surface(x, y, z, cmap=cm.inferno)
plt.colorbar(hSurf, shrink=0.5)
```



Mathematical annotations in graphs

- Broad support for text annotation in matplotlib



Mathematical annotations in graphs

```python
# Generate sinusoidal data
x = np.linspace(0, 4*np.pi, 100)
y = np.sin(x)

# Plot sinusoidal curve
plt.plot(x, y, color="blue")

# labels using latex commands
plt.xlabel(r"Time $t_{i}^{\bullet}$ before $t_{\mathrm{max}}^{2}$")
plt.ylabel(r"Amplitude, $\hat{z}_{\circ} = \sin \left(\frac{t_{i}^{\bullet}}{1} \right)$")

# plot title
plt.title(r"My first Python plot $\int p(x) \mathrm{d}x$")

# text label with arrow using pyplot.annotate
plt.annotate(s = "Sinusoid!", xy=(4.0, -0.75), xytext=(0,-0.25), arrowprops=dict(facecolor="black",linewidth=0.01))

# text annotation within the plot using pyplot.text
plt.text(x=4.7, y = 0.5, s = r"A label $\theta$, $\beta$")
```

Saving figures to files

- One can automate saving figures to files using pyplot.savefig

```
# at the end of the figure

# store figure in file (png, eps, pdf, svg, etc)
plt.savefig(fname="lineplot.svg")
plt.close() # plot stays in memory unless closed
```

- Note that svg format can easily be edited using vector graphics software like Illustrator or Inkscape

Finer control over multipanel figures using `gridspec`

```
import matplotlib.gridspec as gridspec

# set up a 2 x 2 grid with lower row half the size of top row
gs = gridspec.GridSpec(
        nrows=2
        ,ncols=2
        ,height_ratios=[1,0.5]
        ,width_ratios=[1,1])

plt.figure()
# first subplot top left corner, return Axes object for control over axes
ax = plt.subplot(gs[0,0])
ax.plot(...)

# work with axes, e.g., remove x-tick labels
ax.set_xticklabels([])

# second subplot top right corner
plt.subplot(gs[0,1])
plt.plot(...)

# third subplot bottom left corner
plt.subplot(gs[1,0])
plt.plot(...)

# second subplot bottom right corner
plt.subplot(gs[0,1])
plt.plot(...)

plt.show()
```