Lecture 19
Jupyter + plotting

Course: Practical Bioinformatics (BIOL 4220)
Instructor: Michael Landis
Email: michael.landis@wustl.edu

Lecture 19 outline

Last time: protein evolution

This time: plotting

- Jupyter
- matplotlib

https://jupyter.org

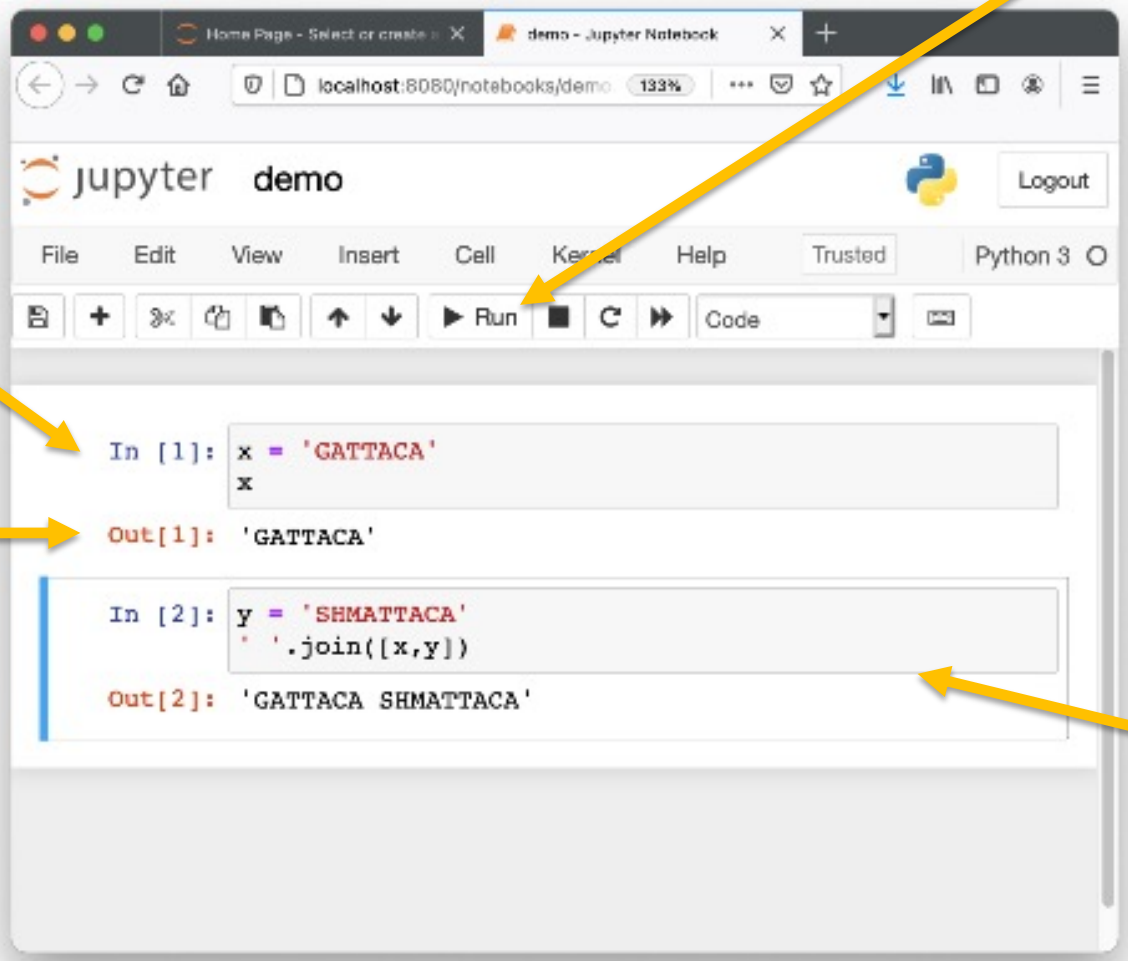
Jupyter is a framework for creating interactive computational ***notebooks***

Jupyter notebooks are organized into a series of ***cells***

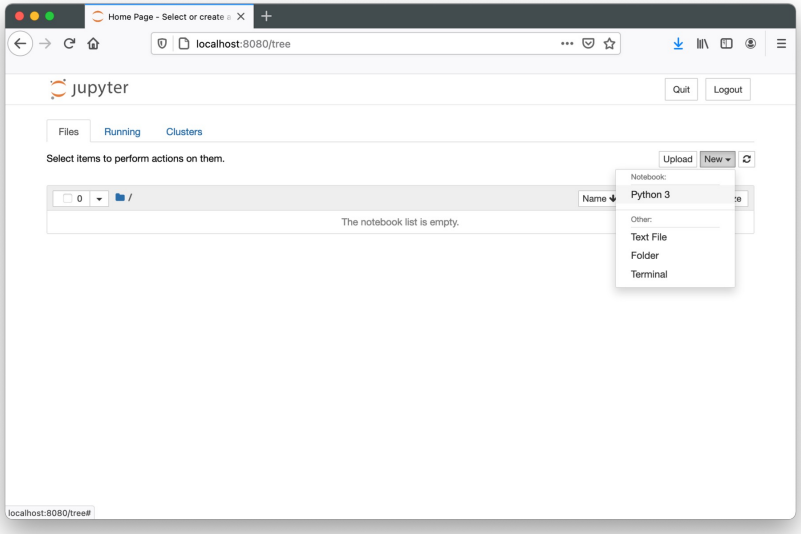
Each cell can contain executable code, richly formatted text, and more

Promising platform for open and reproducible science

Jupyter notebook



Using Jupyter with SSH (will cover in lab)



Jupyter browser

Remote computer (Jupyter host)

1. Connect to VPN
2. SSH into remote computer
3. Launch Jupyter server:
   jupyter notebook --no-browser --port=8080

Local computer (Jupyter client)

4. Create SSH tunnel from port 8080 on remote machine into port 8080 on local workstation:
   ssh -N -L 8080:localhost:8080 snoopy@12.34.56.78
5. Access Jupyter browser page (will require "token"):
   https://localhost:8080

Home Page - Select or create | Untitled - Jupyter Notebook

localhost:8080/notebooks/Untitled.ipynb?kernel_name=p

jupyter Untitled Last Checkpoint: an hour ago (unsaved changes)

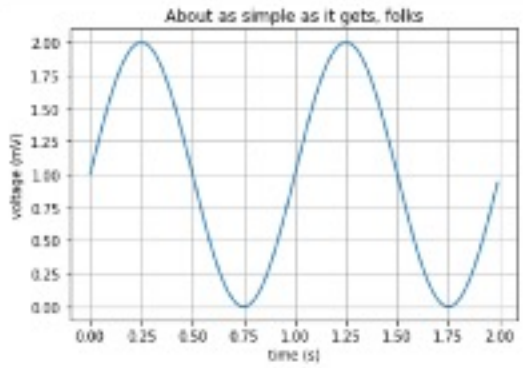
Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

Run Code

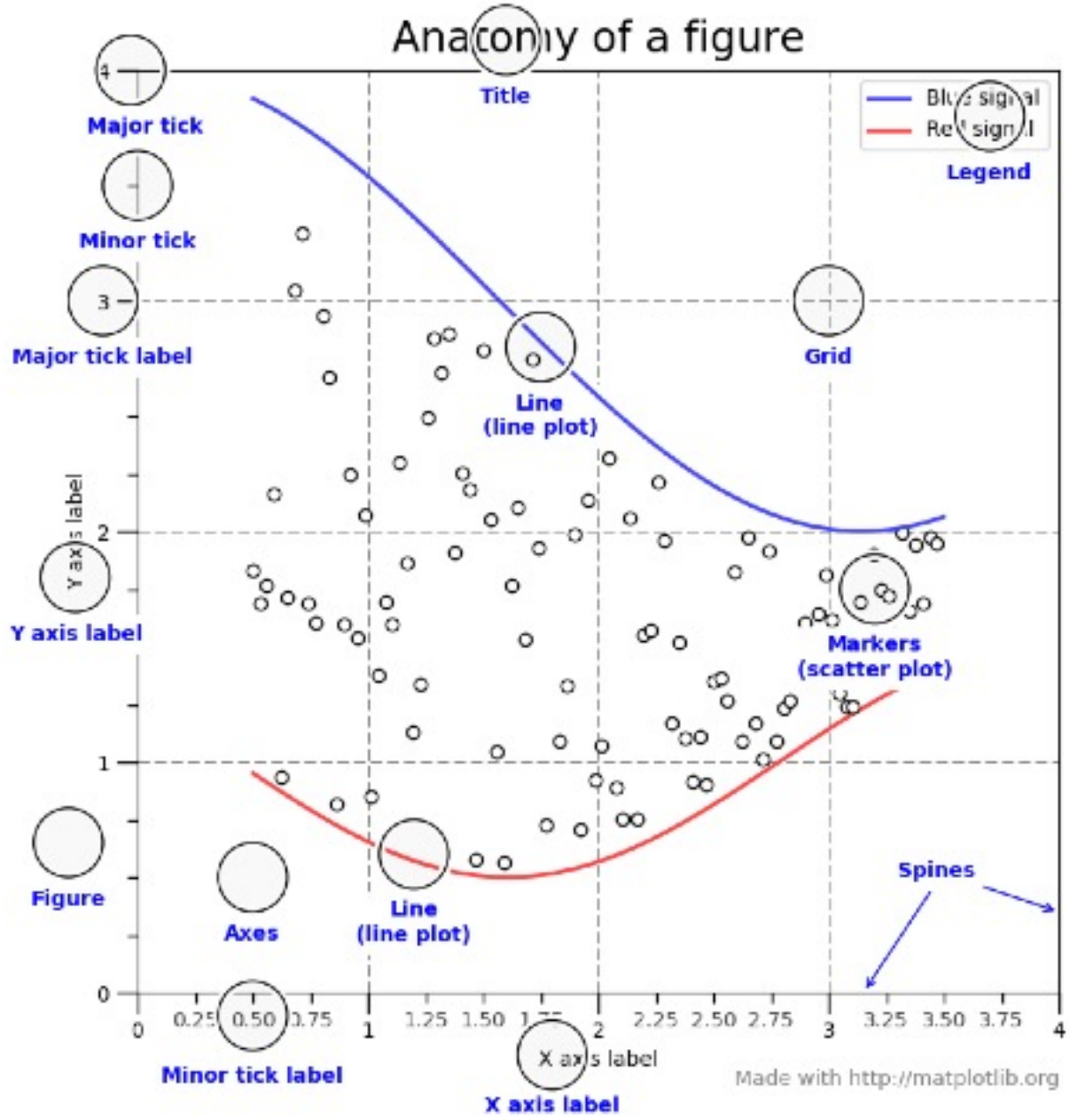
In [12]:
```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

# Data for plotting
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2 * np.pi * t)

fig, ax = plt.subplots()
ax.plot(t, s)

ax.set(xlabel='time (s)', ylabel='voltage (mV)',
       title='About as simple as it gets, folks')
ax.grid()

fig.savefig('test.png')
plt.show()
```



Matplotlib is a library for visualizing data

Supports a wide range of customizable plots from simpler scatterplots, to contoured heatmaps, to interactive 3D plots

Detailed examples for how to use Matplotlib are published through the user guide and gallery



https://matplotlib.org/3.1.1/gallery/showcase/anatomy.html

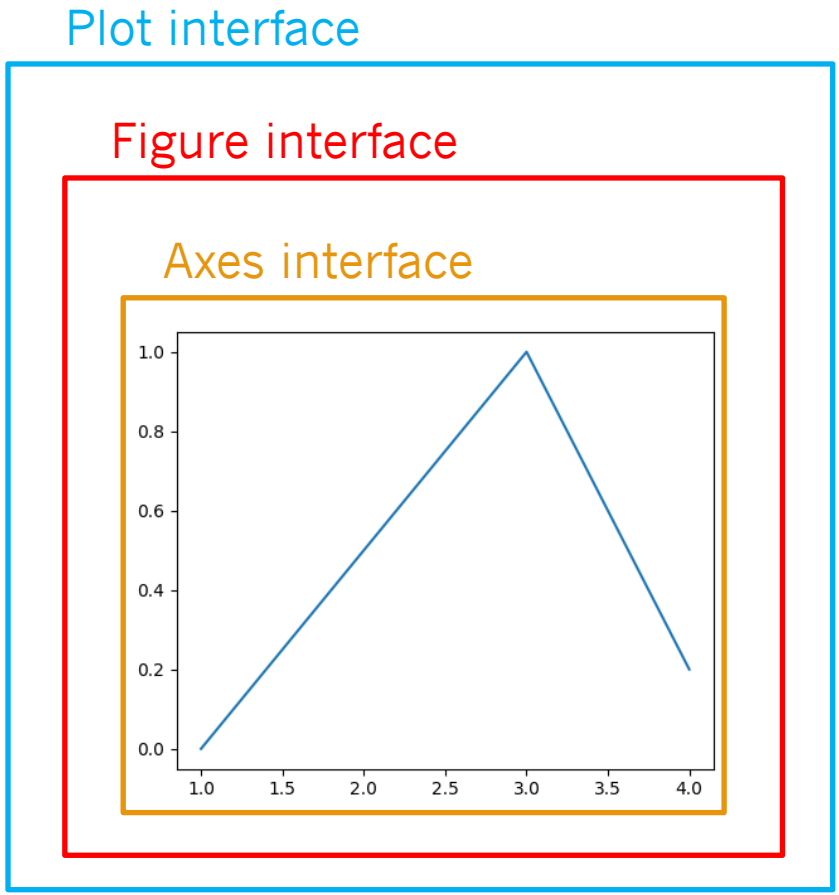
matplotlib drawing interfaces

Explicit interface
- verbose
- allows customization

```
# import plot interface
import matplotlib.pyplot as plt
# make figure object from plot
fig = plt.figure()
# make axis object from figure
ax = fig.subplots()
# plot data within axis object
ax.plot([1, 2, 3, 4], [0, 0.5, 1, 0.2])
```

Implicit interface:
- simpler
- no customization

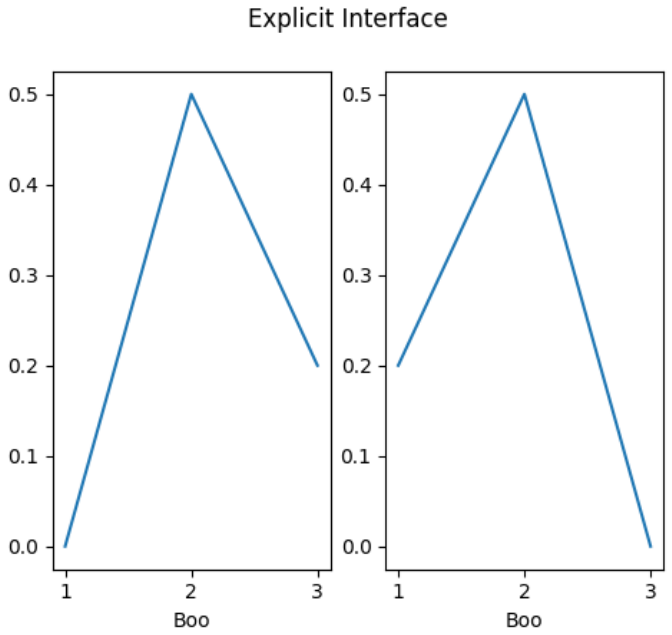
```
# import matplotlib
import matplotlib.pyplot as plt
# draw directly through plot interface
# (internally handles figure and axis)
plt.plot([1, 2, 3, 4], [0, 0.5, 1, 0.2])
```



https://matplotlib.org/stable/users/explain/api_interfaces.html

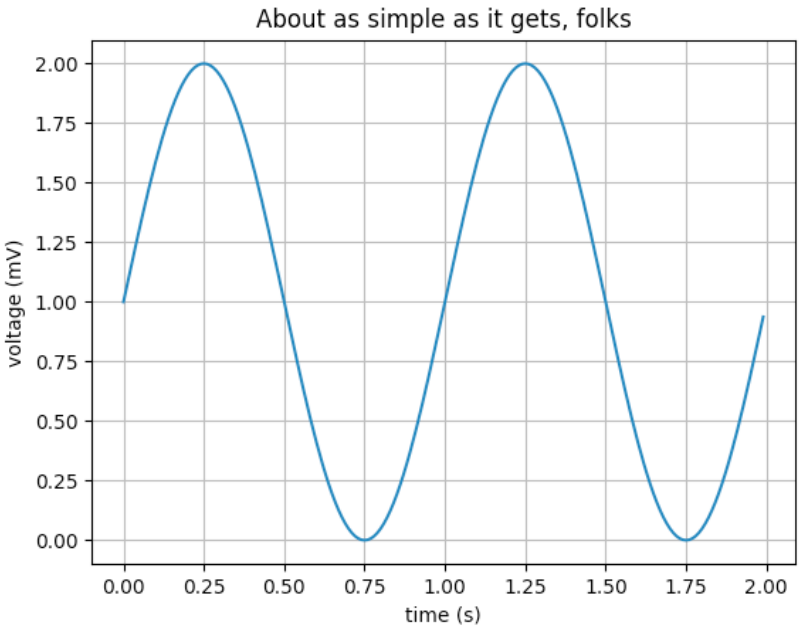
Gallery example: multiple panels

```
# import matplotlib
import matplotlib.pyplot as plt
# construct a matrix of (1,2) subplots
# save figure and axes (list) interfaces
fig, axs = plt.subplots(1, 2)
# create lineplot it the 0th axis interface
axs[0].plot([1, 2, 3], [0, 0.5, 0.2])
# create lineplot it the 1st axis interface
axs[1].plot([3, 2, 1], [0, 0.5, 0.2])
# customize title for the figure interface that
# contains both axis interfaces
fig.suptitle('Explicit Interface')
# customize x-axis label for each axis interface
for i in range(2):
  axs[i].set_xlabel('Boo')
```



Gallery example: lineplot

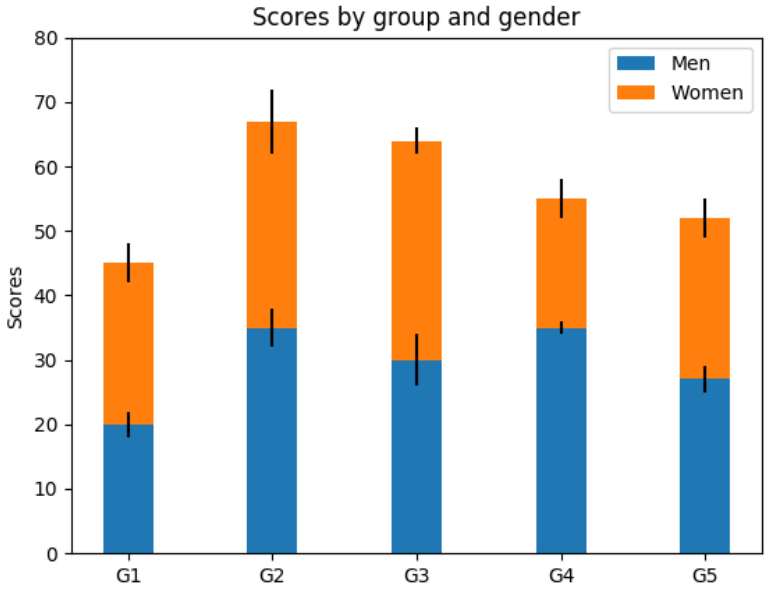
```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np

# Data for plotting
t = np.arange(0.0, 2.0, 0.01)
s = 1 + np.sin(2 * np.pi * t)

fig, ax = plt.subplots()
ax.plot(t, s)

ax.set(xlabel='time (s)',
       ylabel='voltage (mV)',
       title='About as simple as it gets,
              folks')
ax.grid()

fig.savefig("test.png")
plt.show()
```



https://matplotlib.org/3.1.0/gallery/lines_bars_and_markers/simple_plot.html

Gallery example: barplot

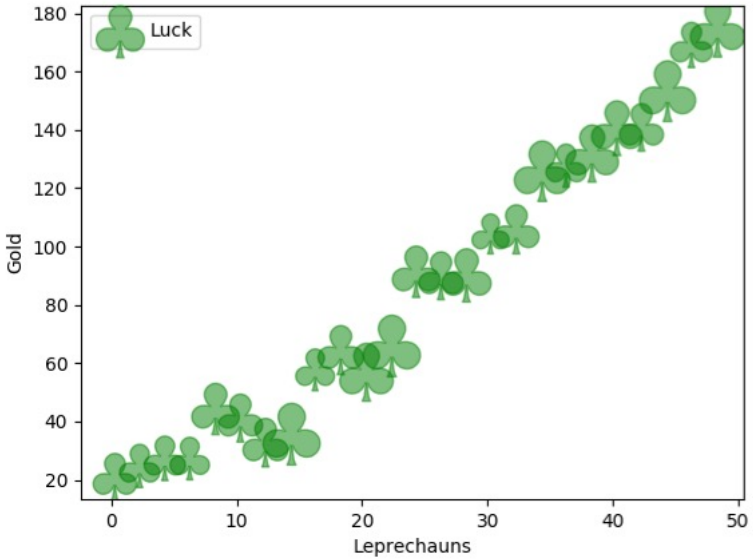
```
import numpy as np
import matplotlib.pyplot as plt


N = 5
menMeans = (20, 35, 30, 35, 27)
womenMeans = (25, 32, 34, 20, 25)
menStd = (2, 3, 4, 1, 2)
womenStd = (3, 5, 2, 3, 3)
ind = np.arange(N) # x-locations for groups
width = 0.35 # the width of the bars:

p1 = plt.bar(ind, menMeans, width, yerr=menStd)
p2 = plt.bar(ind, womenMeans, width,
bottom=menMeans, yerr=womenStd)

plt.ylabel('Scores')
plt.title('Scores by group and gender')
plt.xticks(ind, ('G1', 'G2', 'G3', 'G4', 'G5'))
plt.yticks(np.arange(0, 81, 10))
plt.legend((p1[0], p2[0]), ('Men', 'Women'))

plt.show()
```



https://matplotlib.org/3.1.1/gallery/lines_bars_and_markers/bar_stacked.html

Gallery example: scatterplot

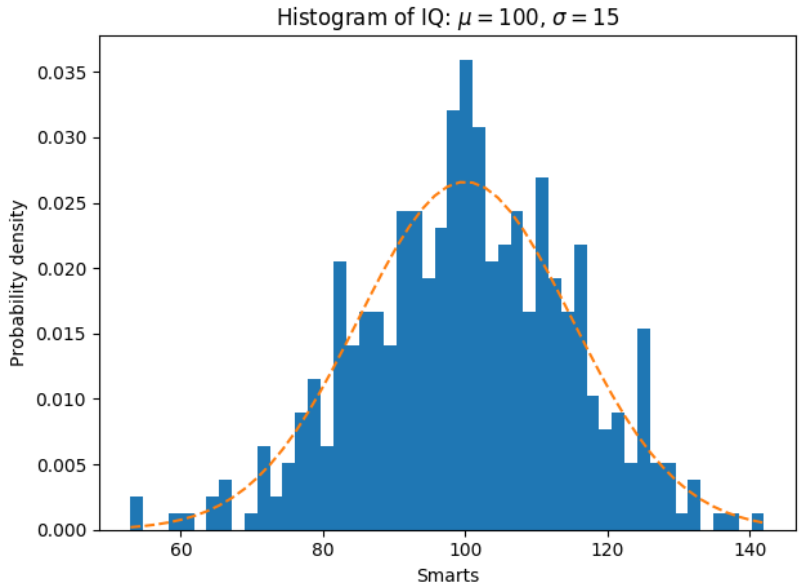
```
import matplotlib.pyplot as plt
import numpy as np

# Fixing random state for reproducibility
np.random.seed(19680801)


x = np.arange(0.0, 50.0, 2.0)
y = x ** 1.3 + np.random.rand(*x.shape) * 30.0
s = np.random.rand(*x.shape) * 800 + 500

plt.scatter(x, y, s, c="g",
    alpha=0.5,
    marker=r'$\clubsuit$',
    label="Luck")
plt.xlabel("Leprechauns")
plt.ylabel("Gold")
plt.legend(loc='upper left')
plt.show()
```



https://matplotlib.org/3.1.1/gallery/lines_bars_and_markers/scatter_symbol.html

Gallery example: histogram

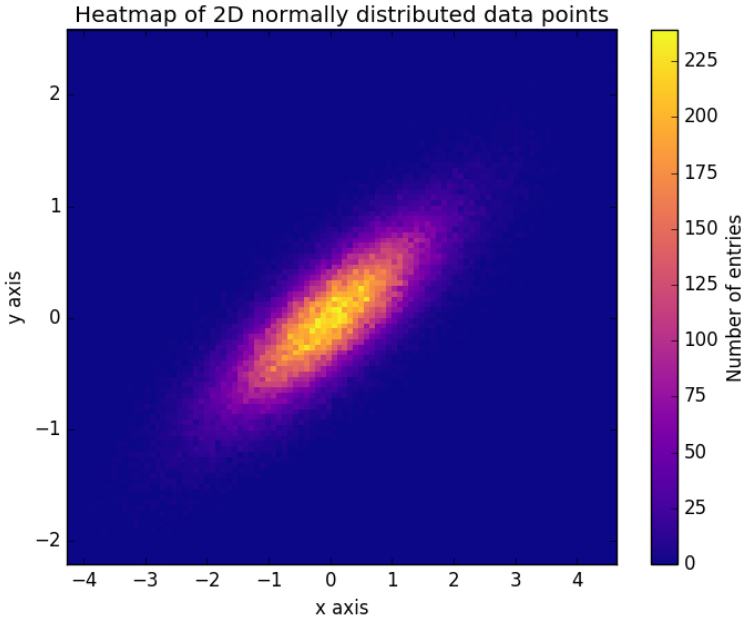
```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(19680801)

# example data
mu = 100 # mean of distribution
sigma = 15 # standard deviation of distribution
x = mu + sigma * np.random.randn(437)
num_bins = 50

# the histogram of the data
fig, ax = plt.subplots()
n, bins, patches = ax.hist(x, num_bins, density=1)

# add a 'best fit' line
y = ((1 / (np.sqrt(2 * np.pi) * sigma)) *
np.exp(-0.5 * (1 / sigma * (bins - mu))**2))
ax.plot(bins, y, '--')
ax.set_xlabel('Smarts')
ax.set_ylabel('Probability density')
title=r'Histogram of IQ: $\mu=100$, $\sigma=15$'
ax.set_title(title)

# Tweak spacing to prevent clipping of ylabel
fig.tight_layout()
plt.show()
```



https://matplotlib.org/3.1.1/gallery/statistics/histogram_features.html

Gallery example: heatmap

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

# Define numbers of data points and bins per axis.
N_numbers = 100000
N_bins = 100

# set random seed
np.random.seed(0)

# Generate 2D normally distributed numbers.
x, y = np.random.multivariate_normal(
  mean=[0.0, 0.0],   # mean
  cov=[[1.0, 0.4],
       [0.4, 0.25]], # covariance matrix
  size=N_numbers).T  # transpose into columns

# Construct 2D histogram using the 'plasma' colormap
plt.hist2d(x, y, bins=N_bins, cmap='plasma')

# Plot a colorbar with label.
cb = plt.colorbar()
cb.set_label('Number of entries')

# Add title and labels to plot.
title='Heatmap of 2D normally distributed data points'
plt.title(title)
plt.xlabel('x axis')
plt.ylabel('y axis')
plt.show()
```



https://riptutorial.com/matplotlib/example/17254/heatmap

OPEN ACCESS Freely available online

PLOS COMPUTATIONAL BIOLOGY

Editorial

Ten Simple Rules for Better Figures

Nicolas P. Rougier[1,2,3]*, Michael Droettboom[4], Philip E. Bourne[5]

1 INRIA Bordeaux Sud-Ouest, Talence, France, **2** LaBRI, UMR 5800 CNRS, Talence, France, **3** Institute of Neurodegenerative Diseases, UMR 5293 CNRS, Bordeaux, France, **4** Space Telescope Science Institute, Baltimore, Maryland, United States of America, **5** Office of the Director, The National Institutes of Health, Bethesda, Maryland, United States of America

Paper linked in course schedule:

1. Know your audience
2. Identify your message
3. Adapt figure to support medium
4. Captions are not optional
5. Do not trust the defaults
6. Use color effectively
7. Do not mislead the reader
8. Avoid "chart junk"
9. Message trumps beauty
10. Get the right [plotting] tool

Overview for Lab 19