W4995 Applied Machine Learning

Visualization and Matplotlib

01/25/17

Andreas Müller

Principles of data visualization

Why?

Explore

Communicate

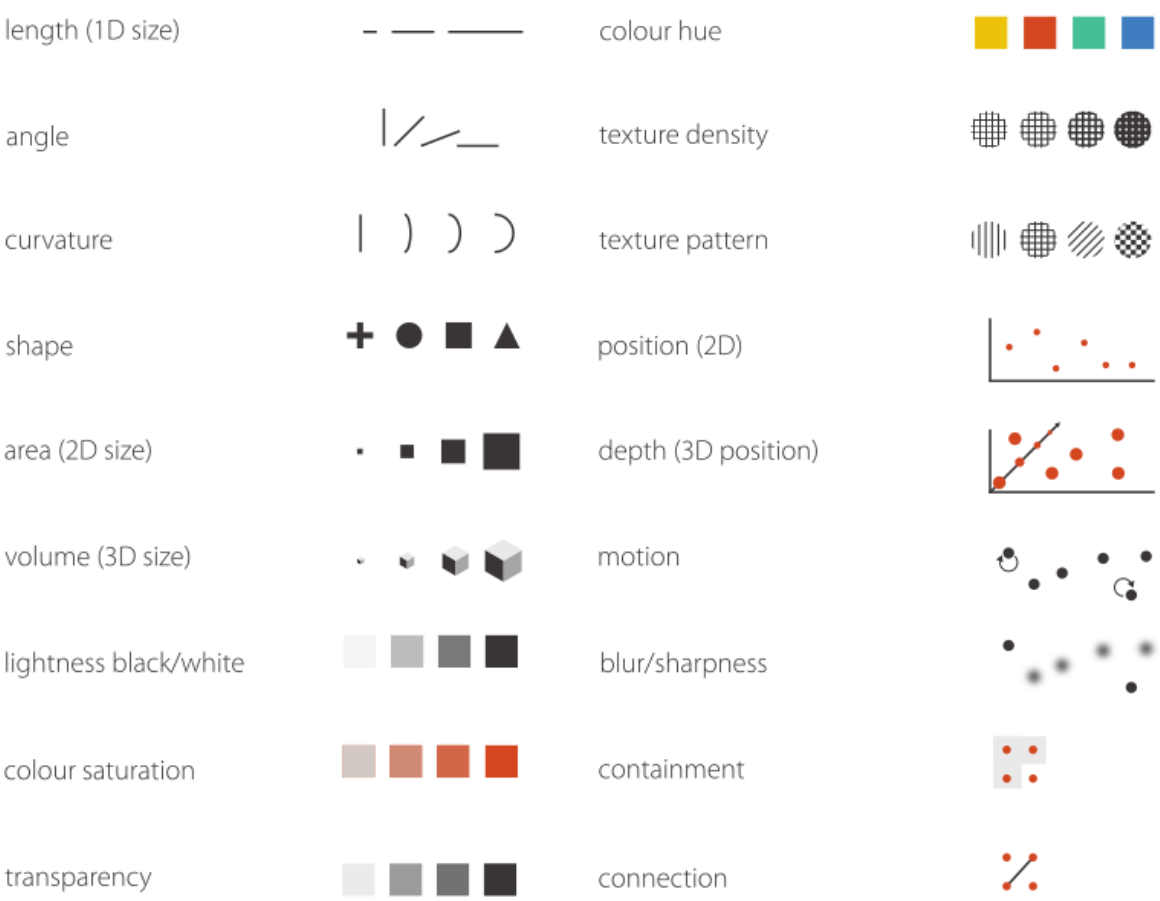
Above else, show the data.
Maximize the data-ink ratio.

- E. Tufte

Tools matter.

- W. S. Cleveland

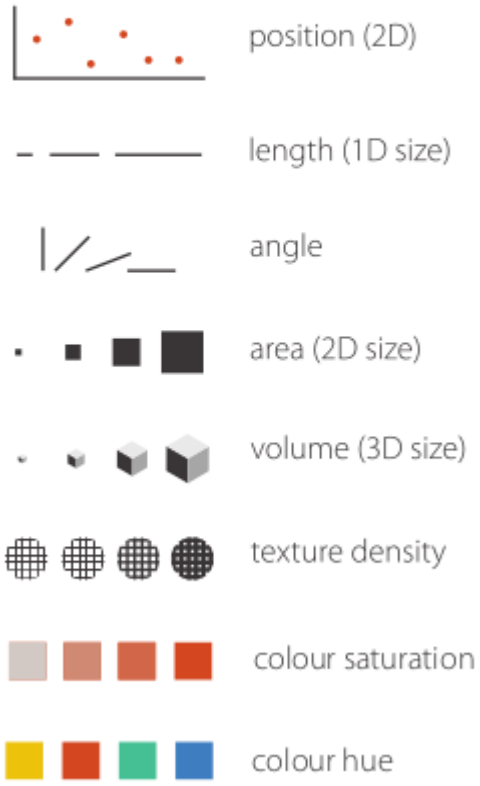
Visual Channels



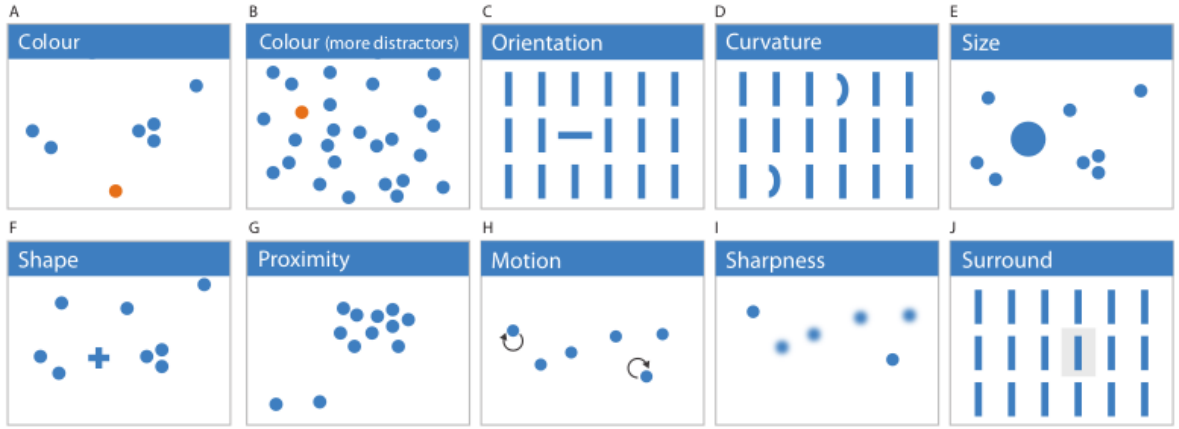
Picking Channels

Quantitative validated

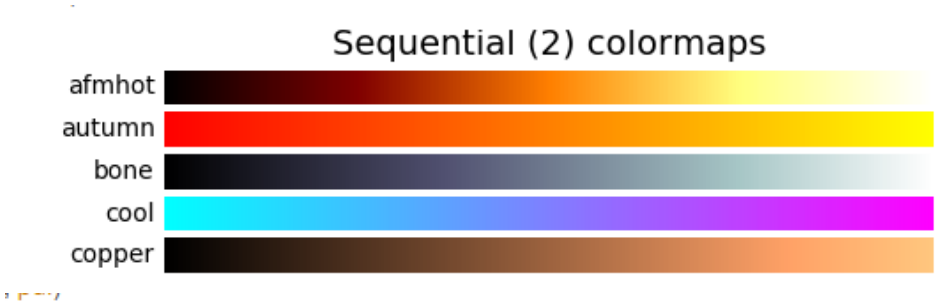
Cleveland and McGill, 1983
Heer and Bostock, 2010
MacKinley, 1986

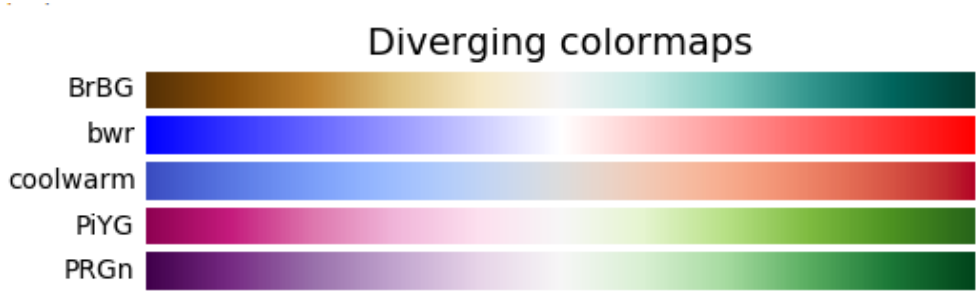


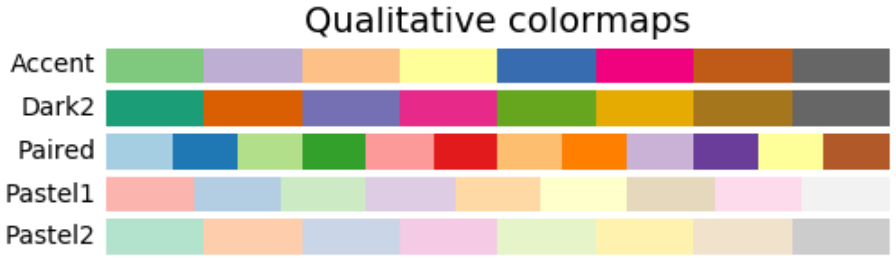
Pop-out Channels

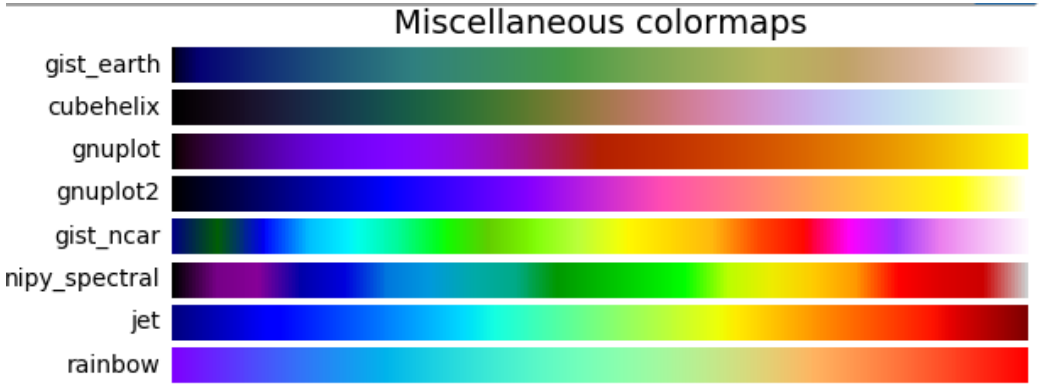


Colormaps


Sequential (2) colormaps
afmhot
autumn
bone
cool
copper

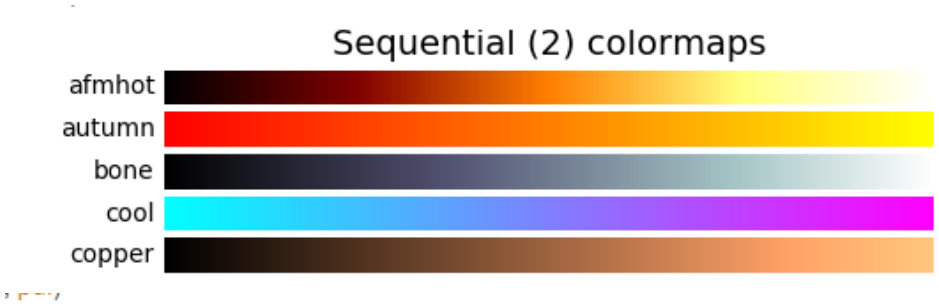

Diverging colormaps
BrBG
bwr
coolwarm
PiYG
PRGn

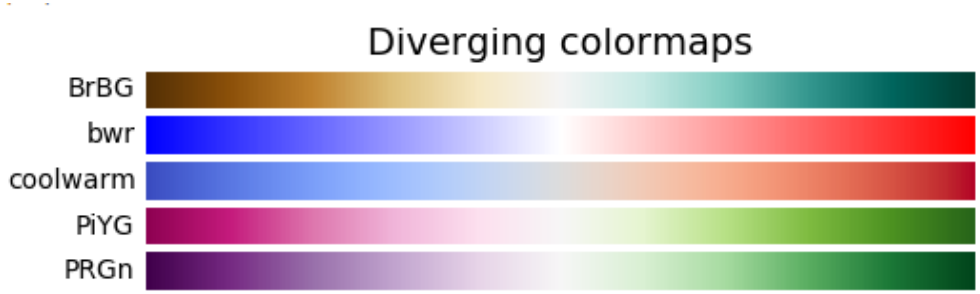

Qualitative colormaps
Accent
Dark2
Paired
Pastel1
Pastel2

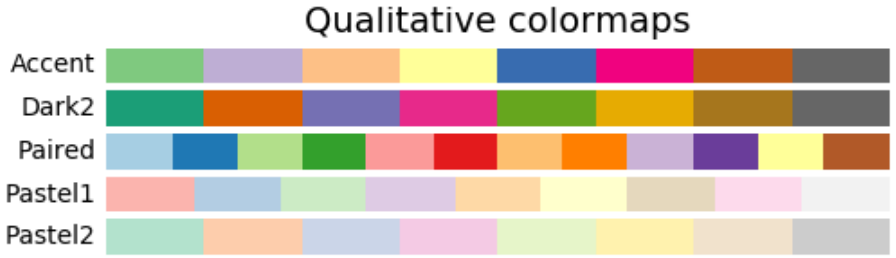

Miscellaneous colormaps
gist_earth
cubehelix
gnuplot
gnuplot2
gist_ncar
nipy_spectral
jet
rainbow

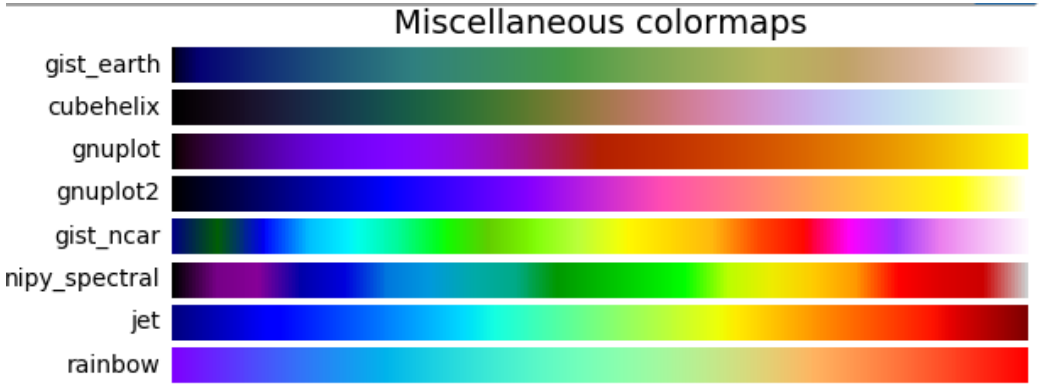

Perceptually Uniform Sequential colormaps
viridis
inferno
plasma
magma

Colormaps


Sequential (2) colormaps
afmhot
autumn
bone
cool
copper

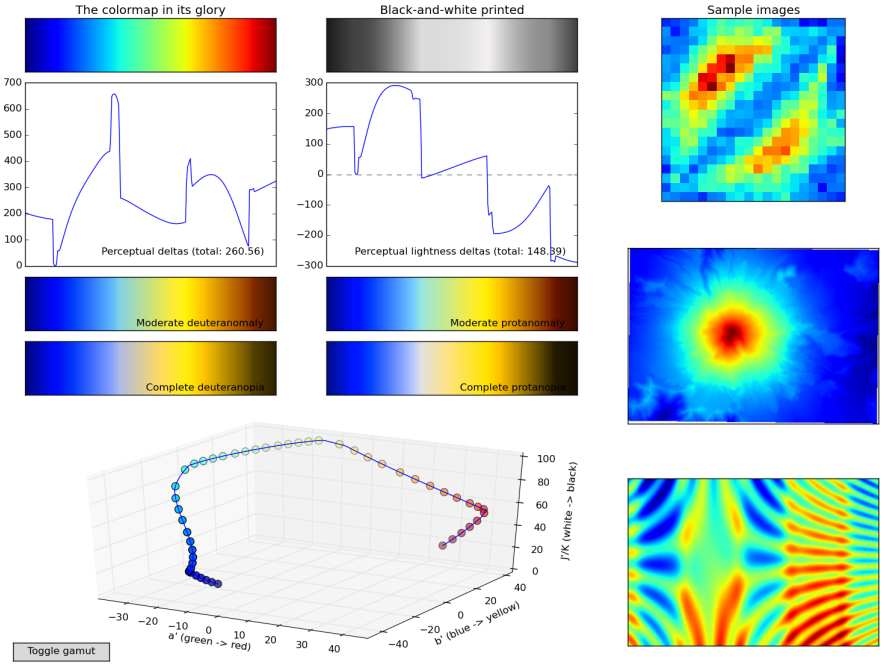

Diverging colormaps
BrBG
bwr
coolwarm
PiYG
PRGn


Qualitative colormaps
Accent
Dark2
Paired
Pastel1
Pastel2

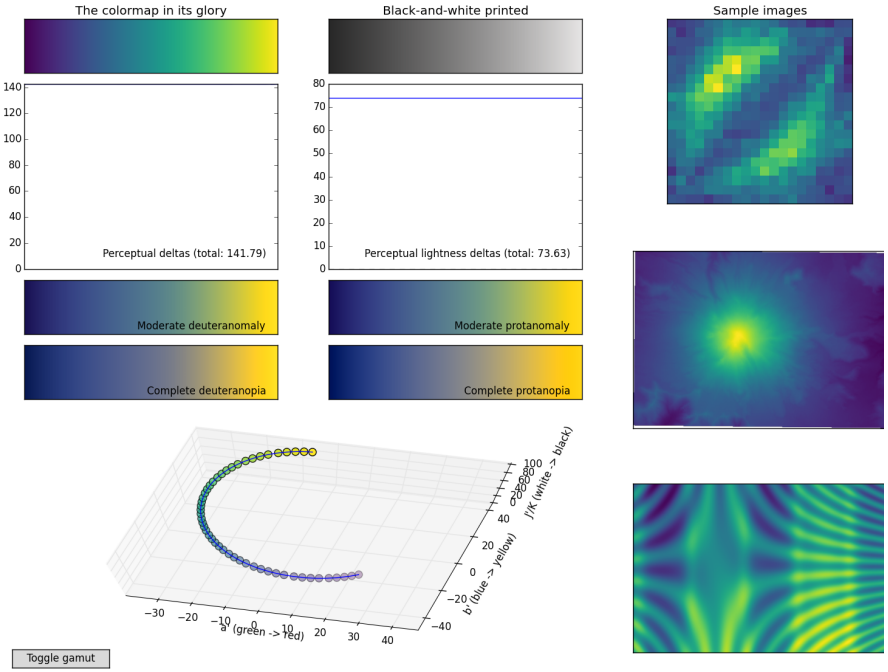

Miscellaneous colormaps
gist_earth
cubehelix
gnuplot
gnuplot2
gist_ncar
nipy_spectral
jet
rainbow


Perceptually Uniform Sequential colormaps
viridis
inferno
plasma
magma

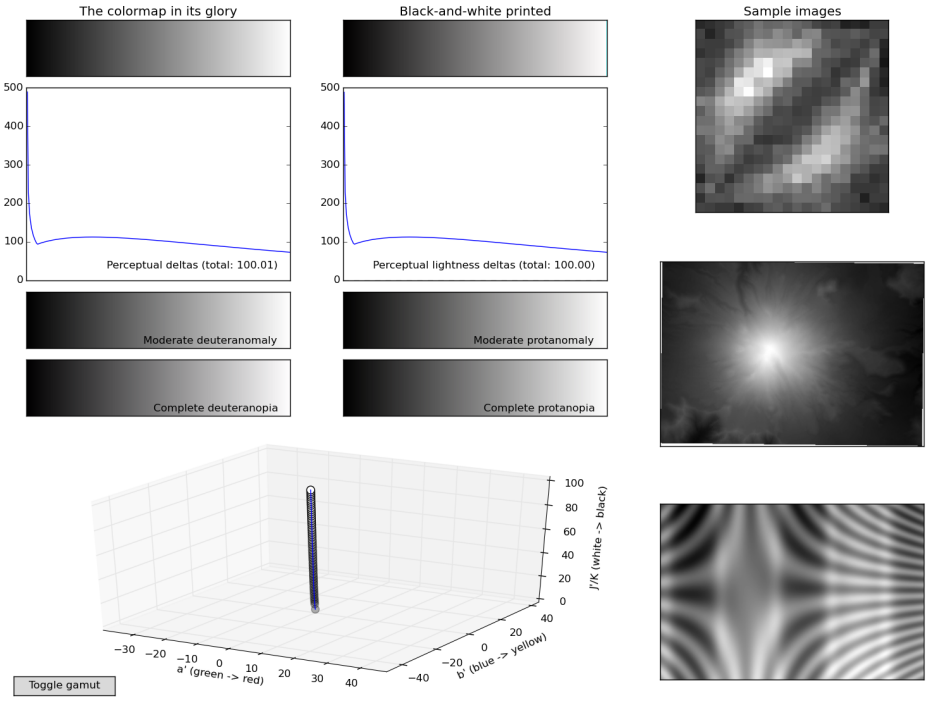
Colormap evaluation: jet



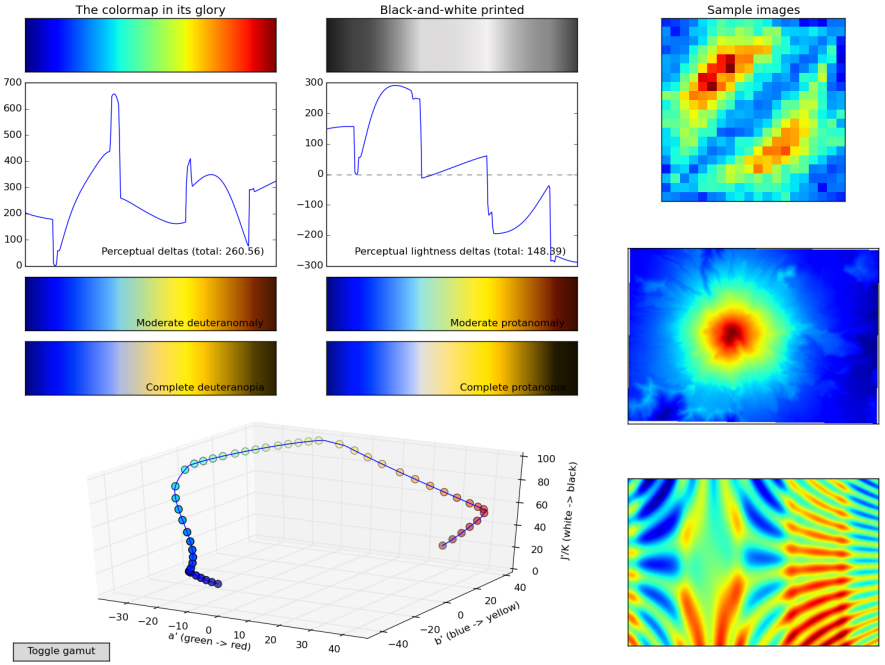
Colormap evaluation: option_d.py



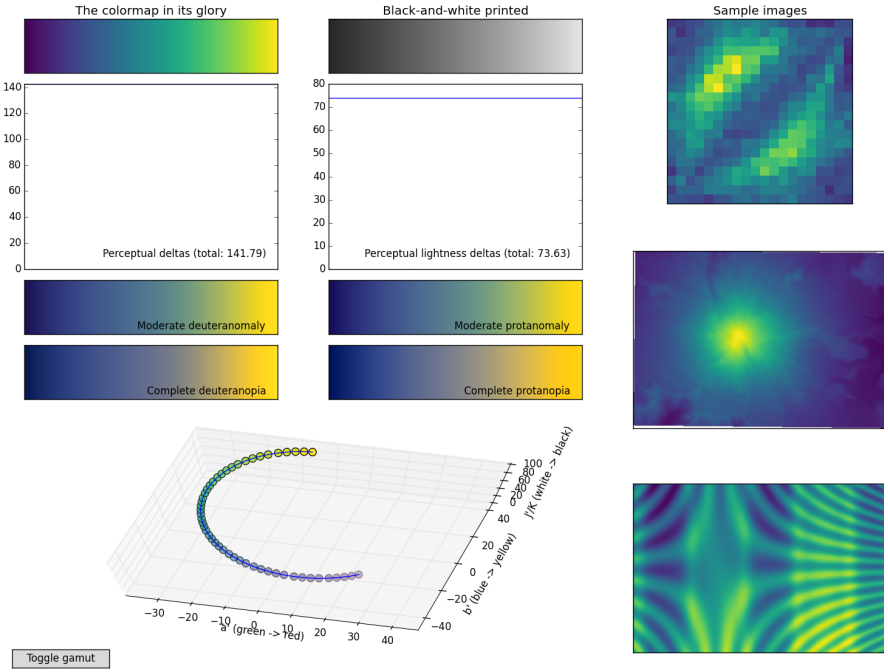
Colormap evaluation: gray



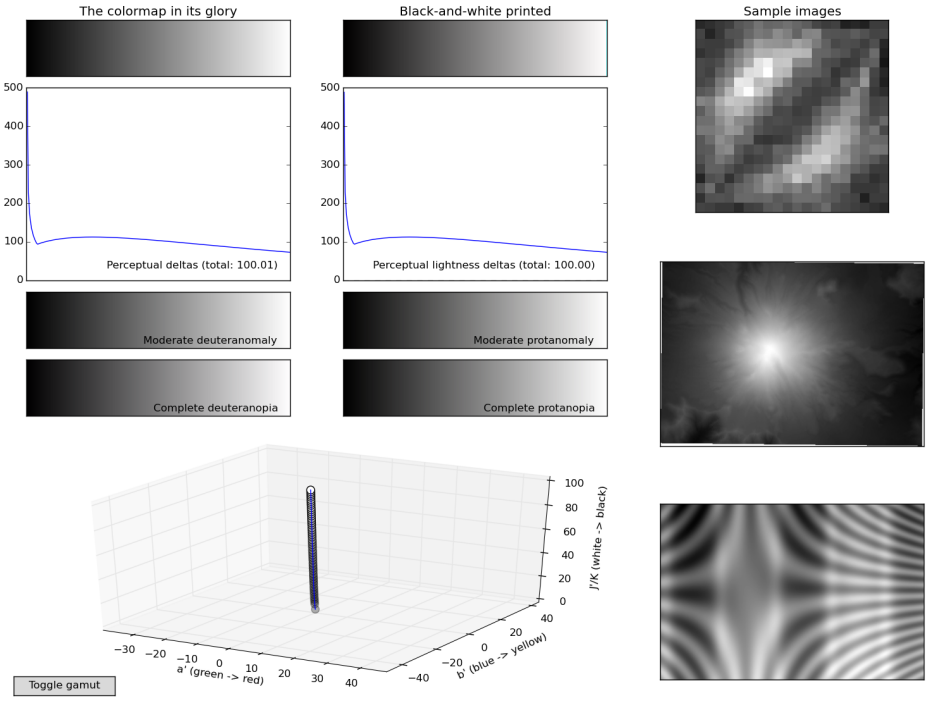
Colormap evaluation: jet



Colormap evaluation: option_d.py



Colormap evaluation: gray



matplotlib

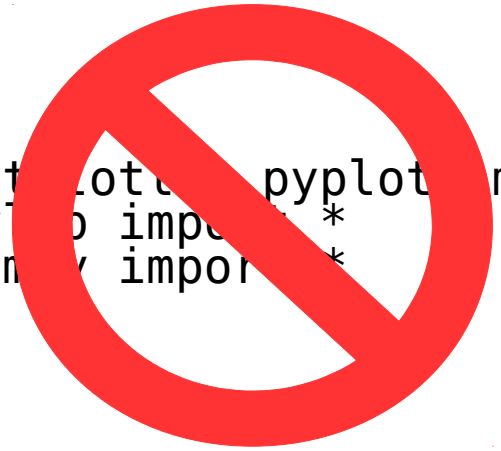
matplotlib v2

update now!
(you can enable classic style if your really want)

Other libs

- pandas plotting - convenience
- seaborn - ready-made stats plots
- bokeh - alternative to matplotlib for in-browser
- several ggplot translations / interfaces

Imports



```
from matplotlib.pyplot import *
from pylab import *
from numpy import *
```

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

matplotlib & Jupyter

`% matplotlib inline`

- sends png to browser
- no panning or zooming
- new figure for each cell
- no changes to previous figures

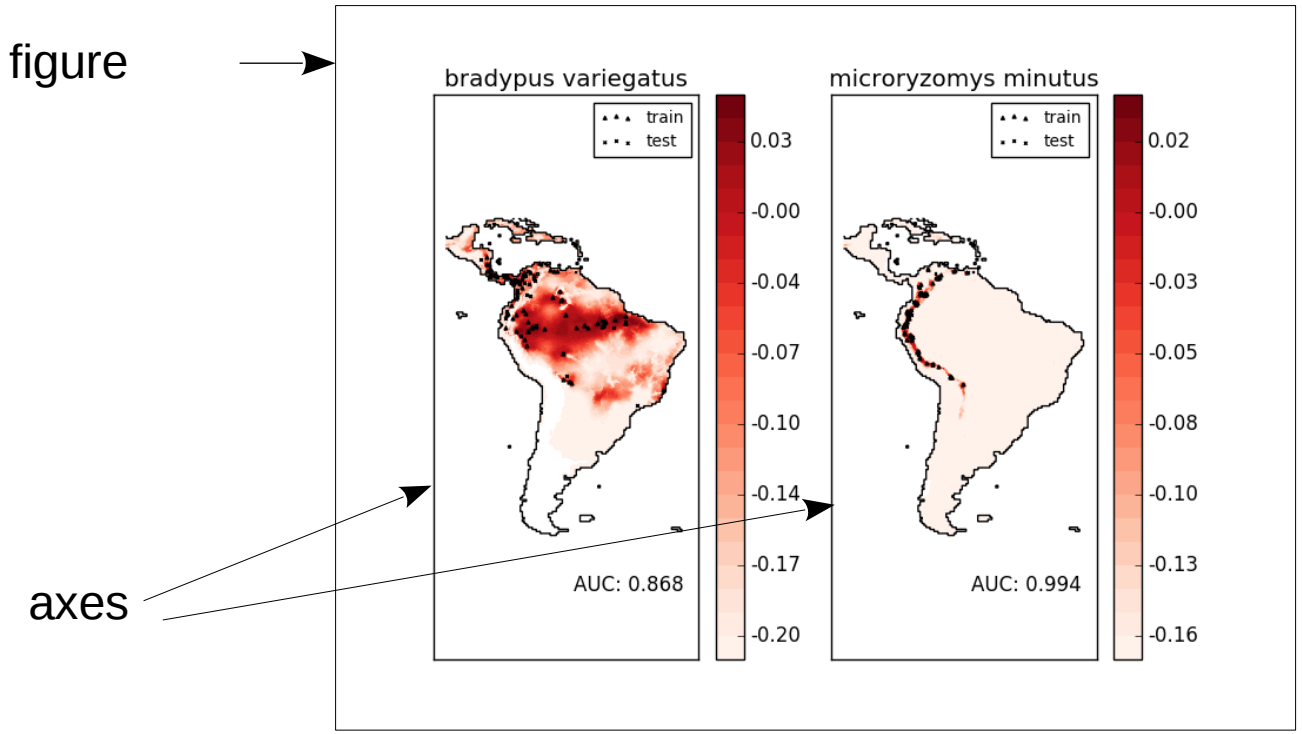
`% matplotlib notebook`

- interactive widget
- all figure features
- need to create figures explicitly
- ability to update figures

Figures and axes

figure = one window or one image file

axes = one drawing area with coordinate system



by default: each figure has one axis

Creating Figures and axes

1st way: don't.

Creates figure with axes on plot command

2nd way: **`fig = plt.figure()`**

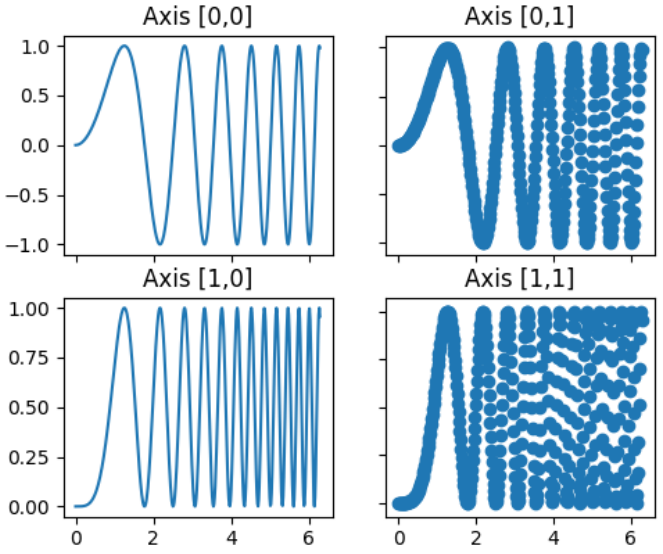
Creates a figure with axes, sets current figure.

Can add more / different axes later.

3rd way: **`fig, ax = plt.subplots(n, m)`**

Creates a figure with a regular grid

of n x m axes.

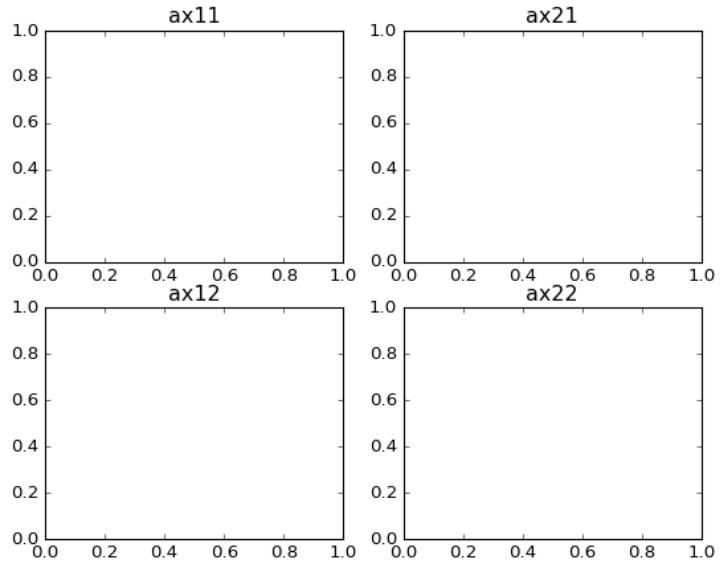


More axes via subplots

`ax = plt.subplot(n, m, i)` `# or plt.subplot(nmi)`

places ax at position “i” in n x m grid (1-based index)

```
ax11 = plt.subplot(2, 2, 1)
ax21 = plt.subplot(2, 2, 2)
ax12 = plt.subplot(2, 2, 3)
ax22 = plt.subplot(2, 2, 4)
```



equivalent:

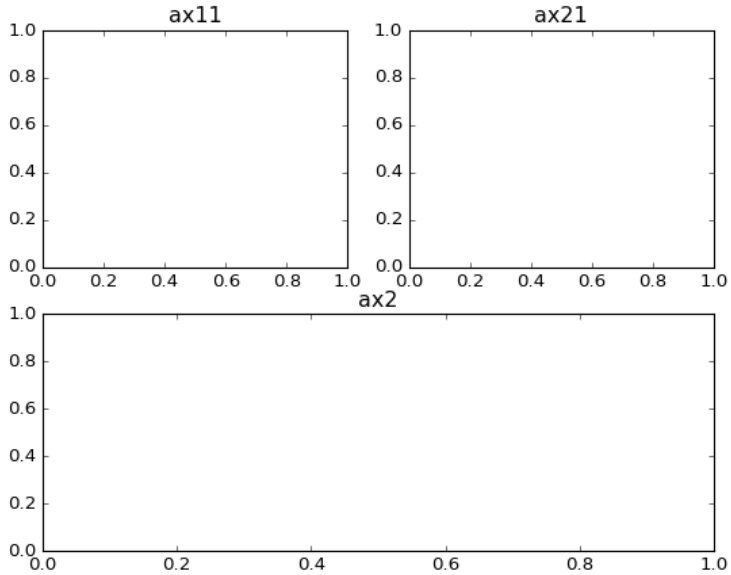
```
fig, axes = plt.subplots(2, 2)
ax11, ax21, ax12, ax22 = axes.ravel()
```

More axes via subplots

`ax = plt.subplot(n, m, i)` `# or plt.subplot(nmi)`

places ax at position “i” in n x m grid (1-based index)

```
ax11 = plt.subplot(2, 2, 1)
ax21 = plt.subplot(2, 2, 2)
ax2 = plt.subplot(2, 1, 2)
```

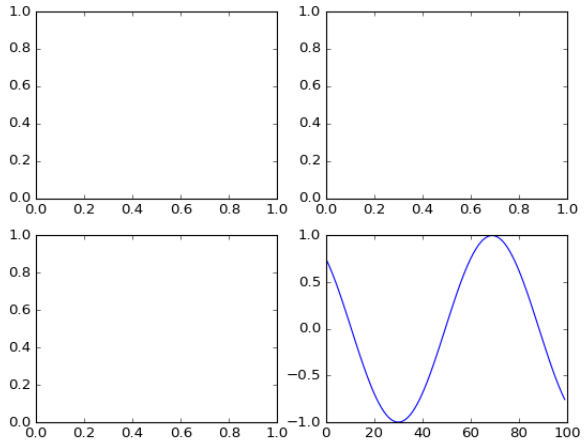


Two interfaces

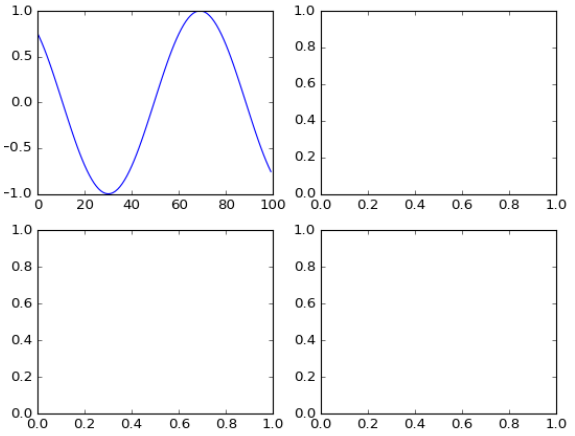
Stateful interface - applies to current figure and axes

object oriented interface - explicitly use object

```
sin = np.sin(np.linspace(-4, 4, 100))
fig, axes = plt.subplots(2, 2)
plt.plot(sin)
```



```
fig, axes = plt.subplots(2, 2)
axes[0, 0].plot(sin)
```



Differences between interfaces

Stateful	Object oriented
`plt.title`	`ax.set_title`
`plt.xlim, plt.ylim`	`ax.set_xlim, ax.set_ylim`
`plt.xlabel, plt.ylabel`	`ax.set_xlabel, ax.set_ylabel`
`plt.xticks, plt.yticks`	`ax.set_xticks, ax.set_yticks` (& `ax.set_xtick_labels`)

```
ax = plt.gca()   # get current axes
fig = plt.gcf()  # get current figure
```

Plotting commands

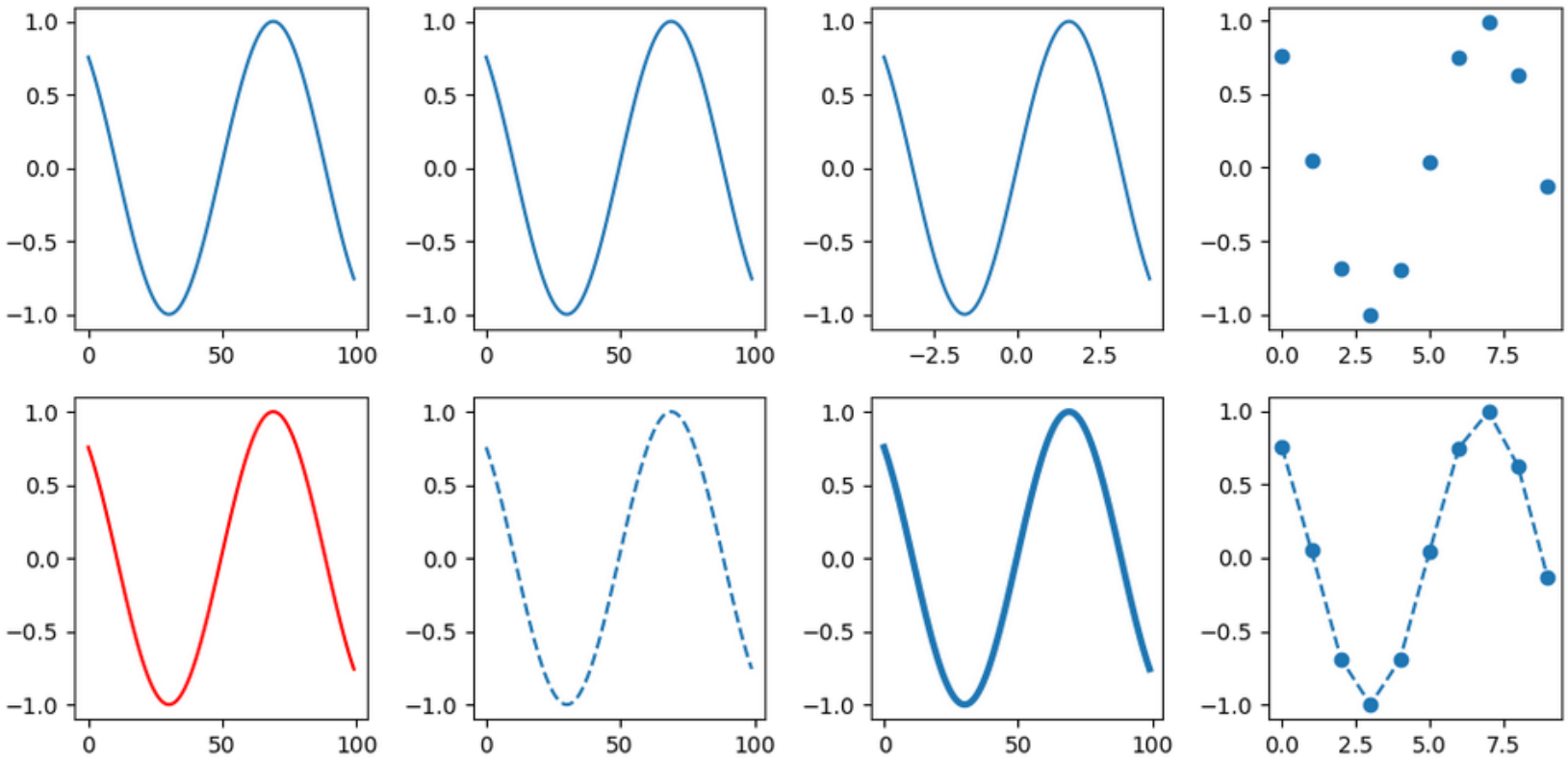
- Gallery:
  http://matplotlib.org/gallery.html

- Plotting commands summary
  http://matplotlib.org/api/pyplot_summary.html

plot

```
fig, ax = plt.subplots(2, 4, figsize=(10, 5))
ax[0, 0].plot(sin)
ax[0, 1].plot(range(100), sin)  # same as above
ax[0, 2].plot(np.linspace(-4, 4, 100), sin)
ax[0, 3].plot(sin[::10], 'o')
ax[1, 0].plot(sin, c='r')
ax[1, 1].plot(sin, '--')
ax[1, 2].plot(sin, lw=3)
ax[1, 3].plot(sin[::10], '--o')
plt.tight_layout() # makes stuff fit - usually works
```

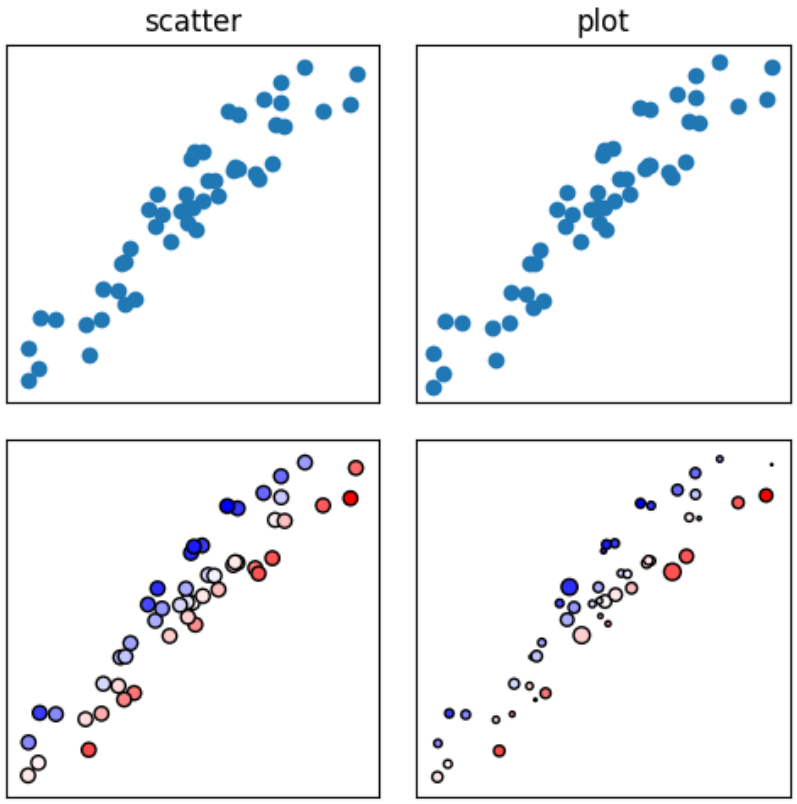
why?



scatter

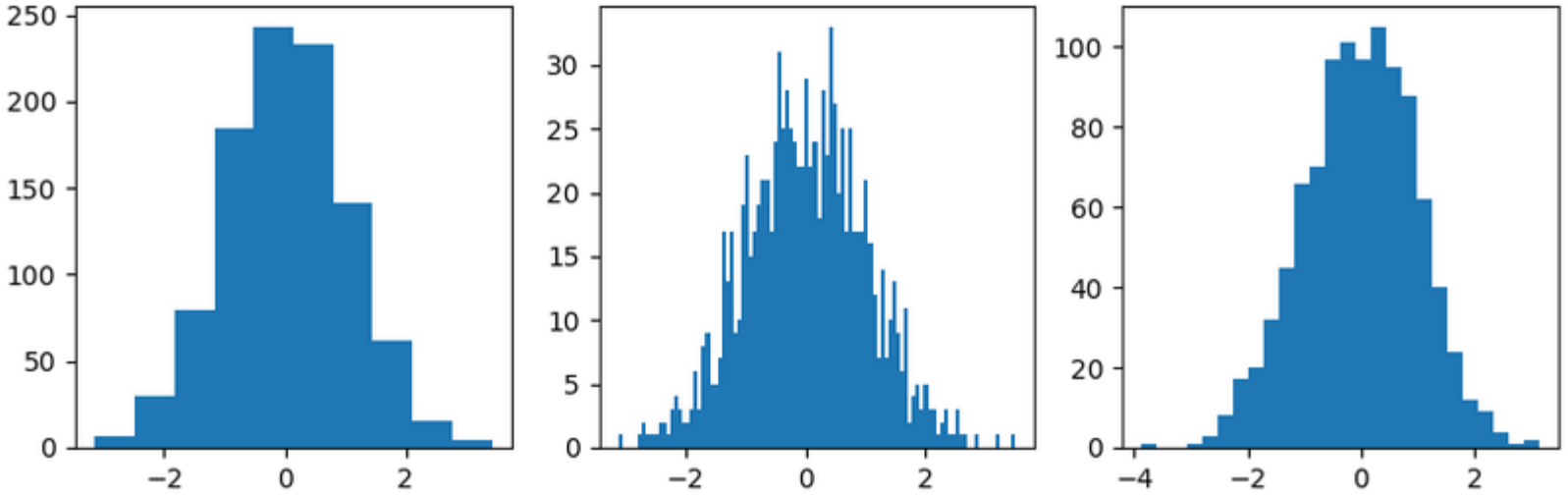
```
x = np.random.uniform(size=50)
y = x + np.random.normal(0, .1, size=50)

fig, ax = plt.subplots(2, 2, figsize=(5, 5),
                       subplot_kw={'xticks': (), 'yticks': ()})
ax[0, 0].scatter(x, y)
ax[0, 0].set_title("scatter")
ax[0, 1].plot(x, y, 'o')
ax[0, 1].set_title("plot")
ax[1, 0].scatter(x, y, c=x-y, cmap='bwr', edgecolor='k')
ax[1, 1].scatter(x, y, c=x-y, s=np.abs(np.random.normal(scale=20, size=50)), cmap='bwr', edgecolor='k')
plt.tight_layout()
```



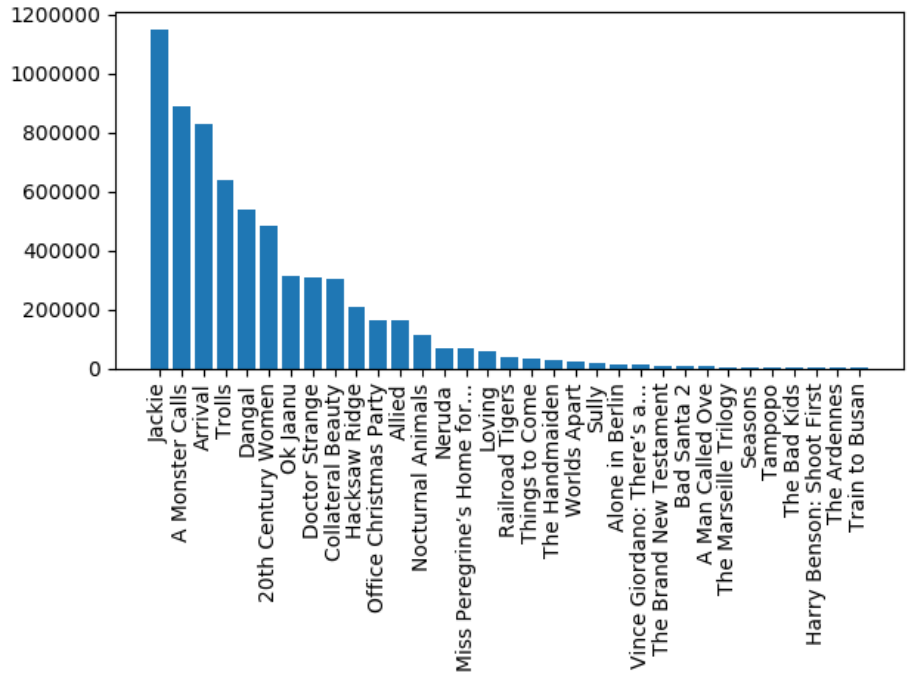
histogram

```
fig, ax = plt.subplots(1, 3, figsize=(10, 3))
ax[0].hist(np.random.normal(size=1000))
ax[1].hist(np.random.normal(size=1000), bins=100)
ax[2].hist(np.random.normal(size=1000), bins="auto")
```

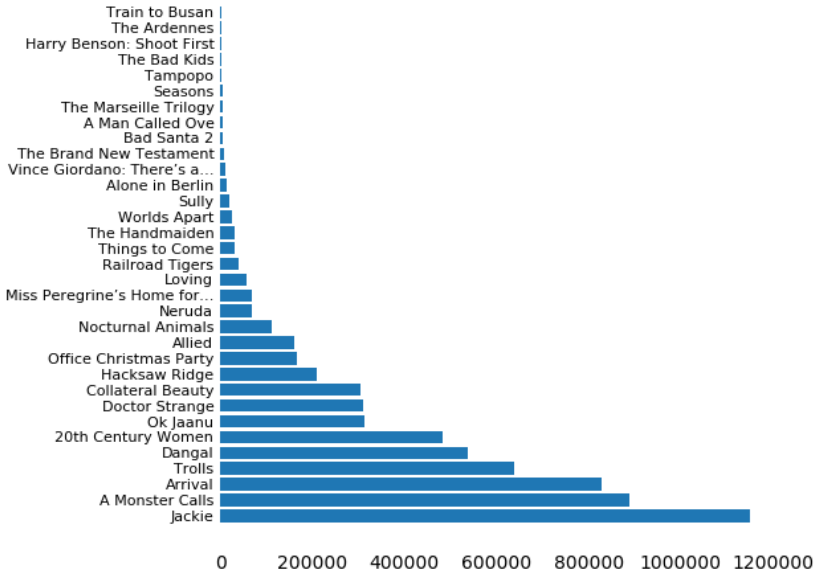


bars

```
plt.figure()
plt.bar(range(len(gross)), gross)
plt.xticks(range(len(gross)), movie, rotation=90)
plt.tight_layout()
```

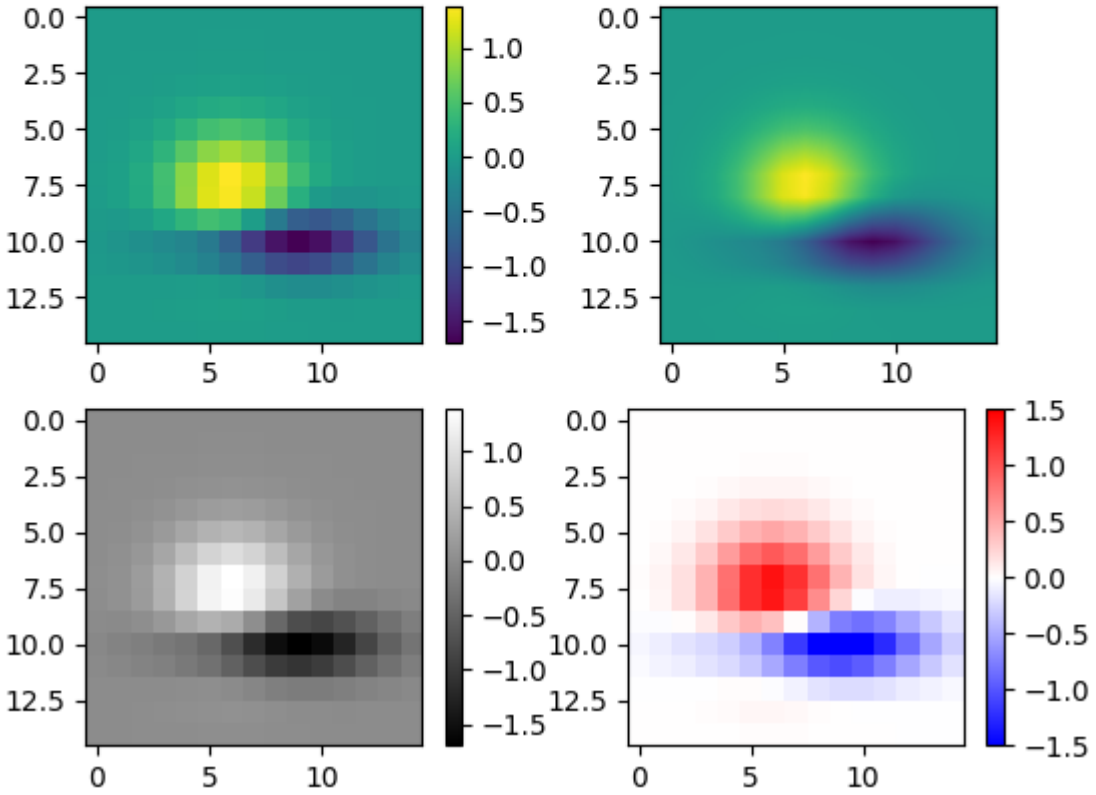



```
plt.figure()
plt.barh(range(len(gross)), gross)
plt.yticks(range(len(gross)), movie, fontsize=8)
ax = plt.gca()
ax.set_frame_on(False)
ax.tick_params(length=0)
plt.tight_layout()
```



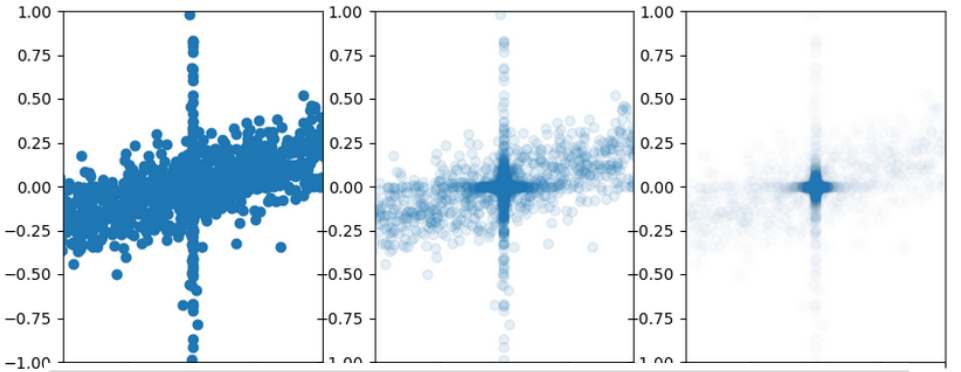
heatmaps

```
fig, ax = plt.subplots(2, 2)
im1 = ax[0, 0].imshow(arr)
ax[0, 1].imshow(arr, interpolation='bilinear')
im3 = ax[1, 0].imshow(arr, cmap='gray')
im4 = ax[1, 1].imshow(arr, cmap='bwr', vmin=-1.5, vmax=1.5)
plt.colorbar(im1, ax=ax[0, 0])
plt.colorbar(im3, ax=ax[1, 0])
plt.colorbar(im4, ax=ax[1, 1])
```

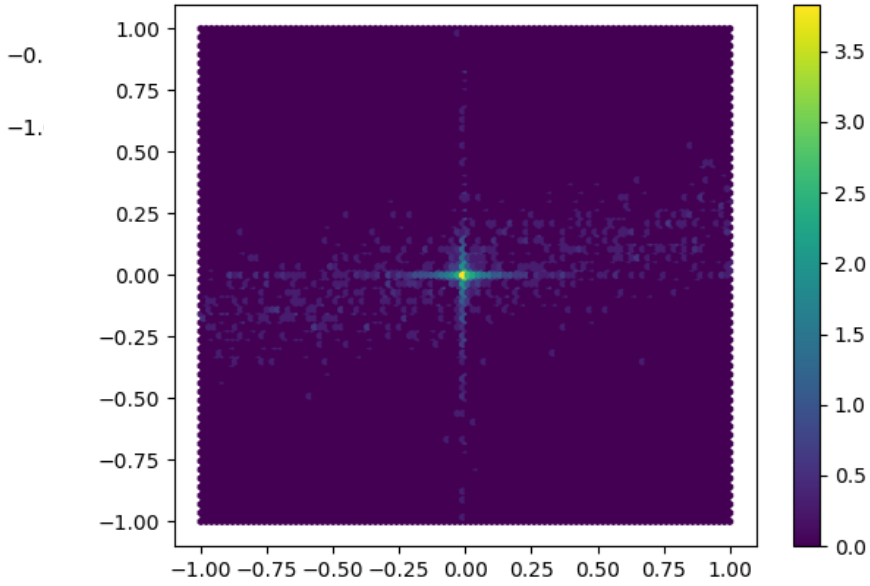


hexgrids

```
fig, ax = plt.subplots(1, 3, figsize=(10, 4),
                       subplot_kw={'xlim': (-1, 1),
                                   'ylim': (-1, 1)})
ax[0].scatter(x, y)
ax[1].scatter(x, y, alpha=.1)
ax[2].scatter(x, y, alpha=.01)
```




```
plt.figure()
plt.hexbin(x, y, bins='log', extent=(-1, 1, -1, 1))
plt.colorbar()
```



Twin x / twiny

```
plt.figure()
ax1 = plt.gca()
line1, = ax1.plot(years, phds)
ax2 = ax1.twinx()
line2, = ax2.plot(years, revenue, c='r')
plt.legend((line1, line2), ("math PhDs awarded", "revenue by arcades"))
ax1.set_ylabel("Math PhDs awarded")
ax2.set_ylabel("revenue by arcades")
```