W4995 Applied Machine Learning

Visualization and Matplotlib

01/30/19

Andreas C. Müller

Principles of Data Visualization

Why?

Why?

Explore

Why?

Explore

Communicate

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

E. Tufte

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

E. Tufte

Tools matter.

W. S. Cleveland

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

E. Tufte

Tools matter.

W. S. Cleveland

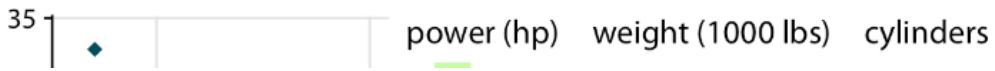
Label stuff.
Spend more time.

me 8 / 49

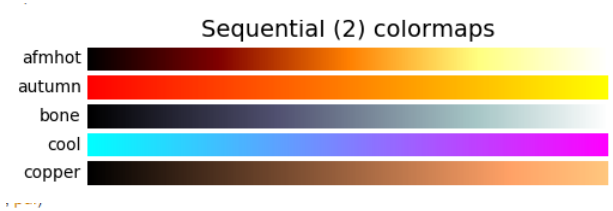
Visual Channels

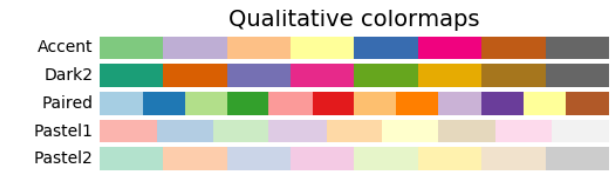
Picking Channels

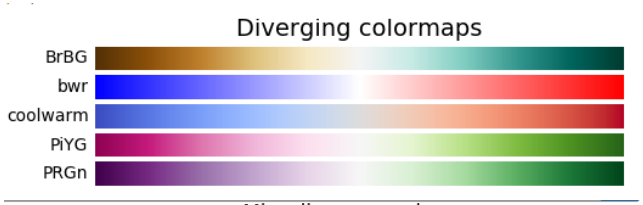
Quantitative validated
Cleveland and McGill, 1983

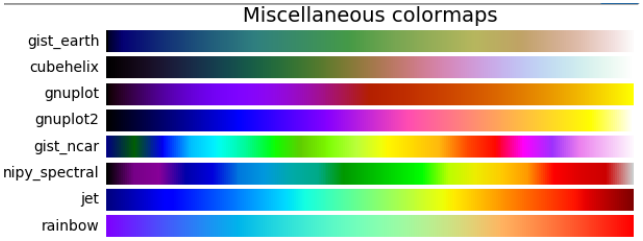

35
power (hp)
weight (1000 lbs)
cylinders

Colormaps


Sequential (2) colormaps
afmhot
autumn
bone
cool
copper

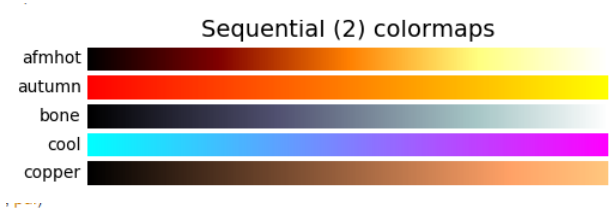

Qualitative colormaps
Accent
Dark2
Paired
Pastel1
Pastel2

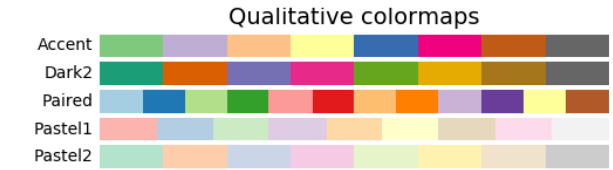

Diverging colormaps
BrBG
bwr
coolwarm
PiYG
PRGn

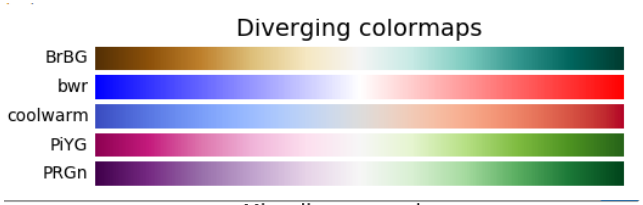

Miscellaneous colormaps
gist_earth
cubehelix
gnuplot
gnuplot2
gist_ncar
nipy_spectral
jet
rainbow

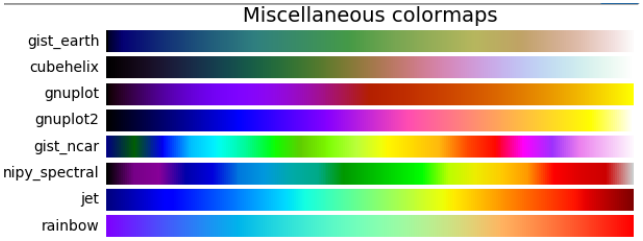

Perceptually Uniform Sequential colormaps
viridis

Colormaps

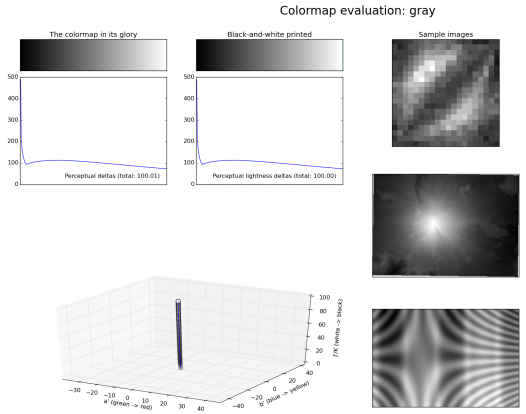

Sequential (2) colormaps
afmhot
autumn
bone
cool
copper

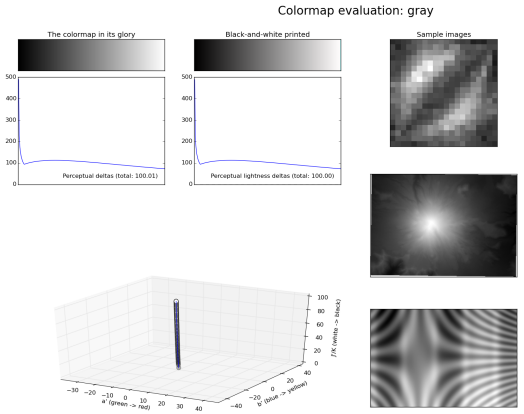

Qualitative colormaps
Accent
Dark2
Paired
Pastel1
Pastel2

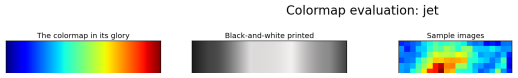

Diverging colormaps
BrBG
bwr
coolwarm
PiYG
PRGn


Miscellaneous colormaps
gist_earth
cubehelix
gnuplot
gnuplot2
gist_ncar
nipy_spectral
jet
rainbow

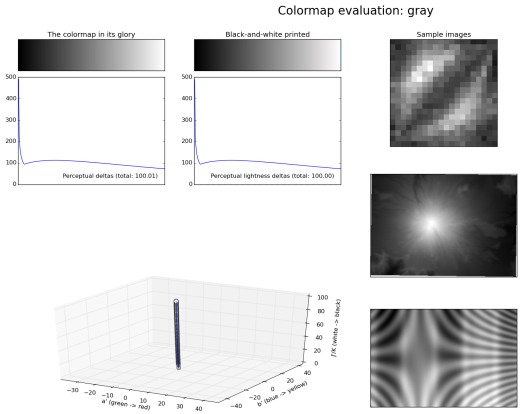

Perceptually Uniform Sequential colormaps
viridis

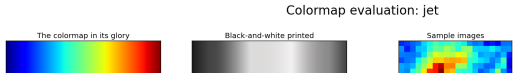

Colormap evaluation: gray
The colormap in its glory
Black-and-white printed
Sample images
Perceptual deltas (total: 100.01)
Perceptual lightness deltas (total: 100.00)
J/K (white -> black)
a* (green -> red)
b* (blue -> yellow)


Colormap evaluation: gray
The colormap in its glory
Black-and-white printed
Sample images
Perceptual deltas (total: 100.01)
Perceptual lightness deltas (total: 100.00)
J/K (white -> black)
a (green -> red)
b (blue -> yellow)


Colormap evaluation: jet
The colormap in its glory
Black-and-white printed
Sample images


Colormap evaluation: option_d.py
The colormap in its glory
Black-and-white printed
Sample images


Colormap evaluation: gray
The colormap in its glory
Black-and-white printed
Sample images
Perceptual deltas (total: 100.01)
Perceptual lightness deltas (total: 100.00)
J/K (white -> black)
a (green -> red)
b (blue -> yellow)


Colormap evaluation: jet
The colormap in its glory
Black-and-white printed
Sample images


Colormap evaluation: option_d.py
The colormap in its glory
Black-and-white printed
Sample images

matplotlib

matplotlib v2 (also v3 now)

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
- bqplot
- plotly
- altair (the cool new kid)
- yellowbrick (plotting for sklearn)

Imports

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

Imports

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

NO!

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

YES!

matplotlib & Jupyter

```
%matplotlib inline
```

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

matplotlib & Jupyter

`%matplotlib inline`

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

`%matplotlib notebook`

- interactivity in old notebook
- doesn't work in jupyter lab
- need to create separate figures
- ability to update figures

matplotlib & Jupyter

`%matplotlib inline`

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

`%matplotlib notebook`

- interactivity in old notebook
- doesn't work in jupyter lab
- need to create separate figures
- ability to update figures

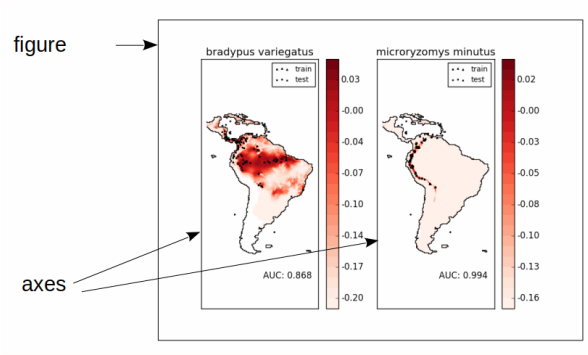
`%matplotlib widget`

- interactive widget
- all figure features
- need to create separate figures explicitly
- ability to update figures
- installation instructions: https://github.com/matplotlib/jupyter-matplotlib

Figures and Axes

figure = one window or one image file

axes = drawing areas with coordinate system



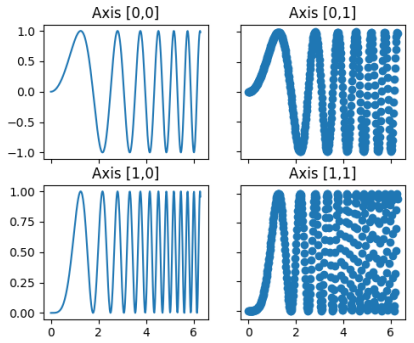
by default: each figure has one axis

Creating Figures and Axes

1st way: don't.
Creates figure with axes on plot command

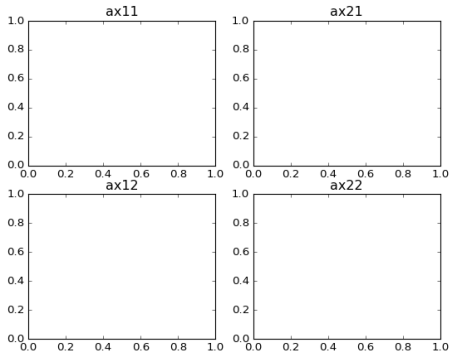
2nd way: `fig = plt.figure()`
Creates a figure with axes, sets current figure. Can add more / different axes later.

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)
```



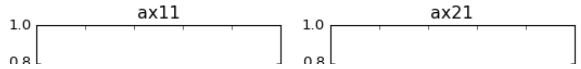
equivalent:

```
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)
```

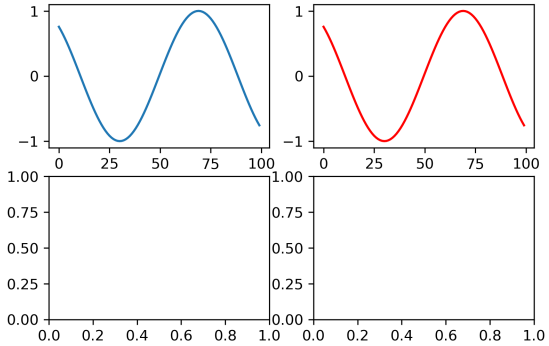


Two Interfaces

Stateful interface - applies to current figure and axes

object oriented interface - explicitly use object

```
sin = np.sin(np.linspace(-4, 4, 100))
plt.subplot(2, 2, 1)
plt.plot(sin)
plt.subplot(2, 2, 2)
plt.plot(sin, c='r')


fig, axes = plt.subplots(2, 2)
axes[0, 0].plot(sin)
axes[0, 1].plot(sin, c='r')
```



Differences Between the Interfaces

`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)`

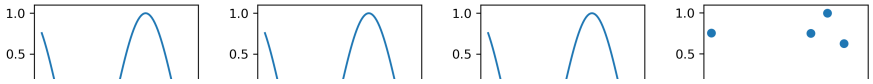
```
ax = plt.gca()   # get current axes
fig = plt.gcf()  # get current figure
```

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
```



plot

```
fig, ax = plt.subplots(2, 4, figsize=(10, 5))
ax[0, 0].plot(sin)
```

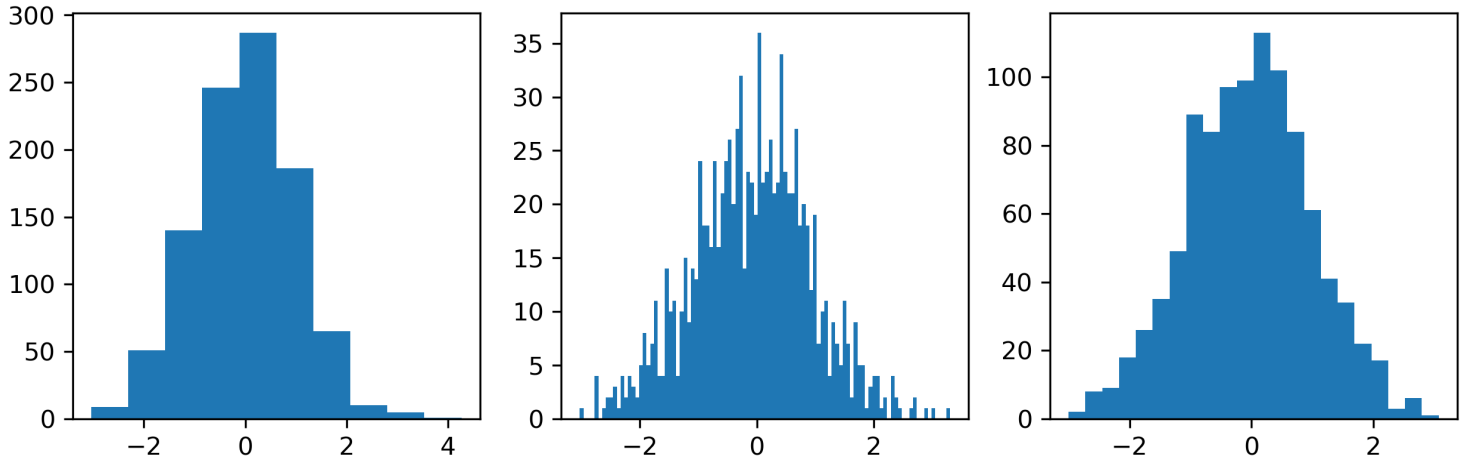
why?

scatter

```
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')
```

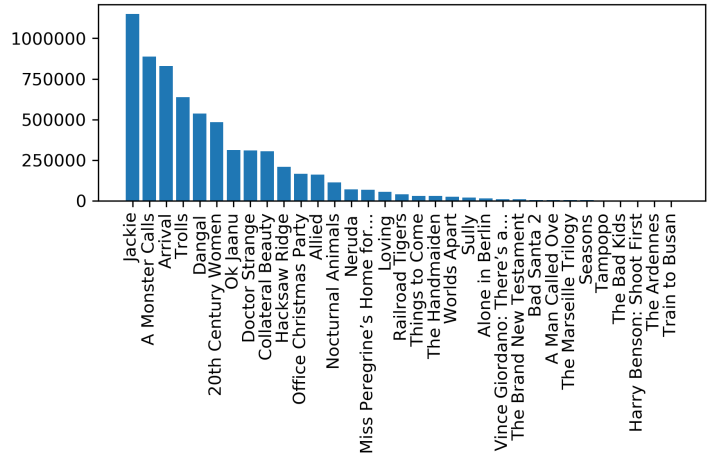
histogram

```
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")
```

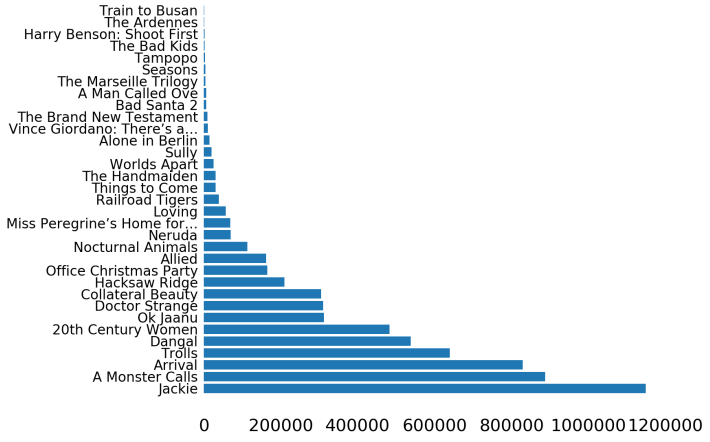


bars

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

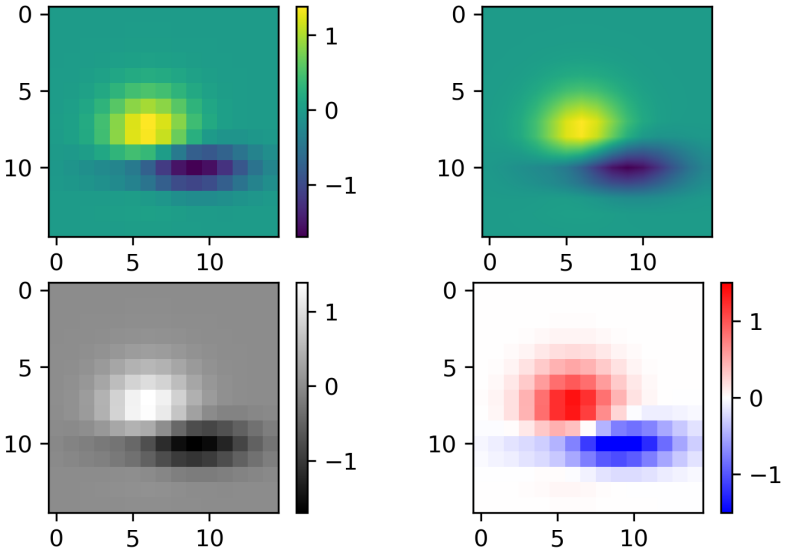



```
plt.barh(range(len(gross)), gross)
plt.yticks(range(len(gross)), movie, fontsize=8)
ax = plt.gca()
ax.set_frame_on(False)
ax.tick_params(length=0)
```

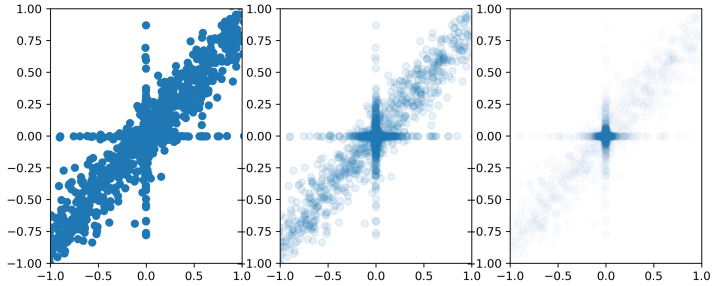


heatmaps

```
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])
```

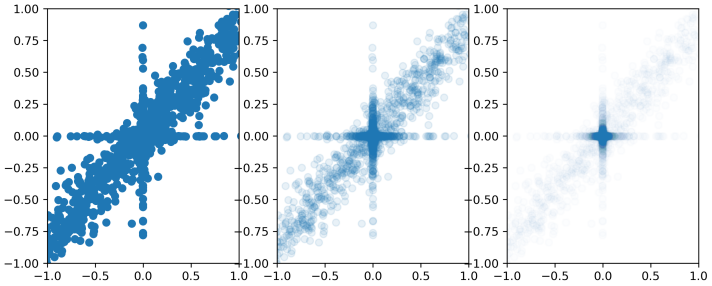


```
fig, ax = plt.subplots(1, 3, figsize=(10, 4),
                       subplot_kw={'xlim': (-1, 1),
                                   'ylim': (-1, 1)})
ax[0].scatter(x, y)
ax[1].scatter(x, y, alpha=.1)
ax[2].scatter(x, y, alpha=.01)
```

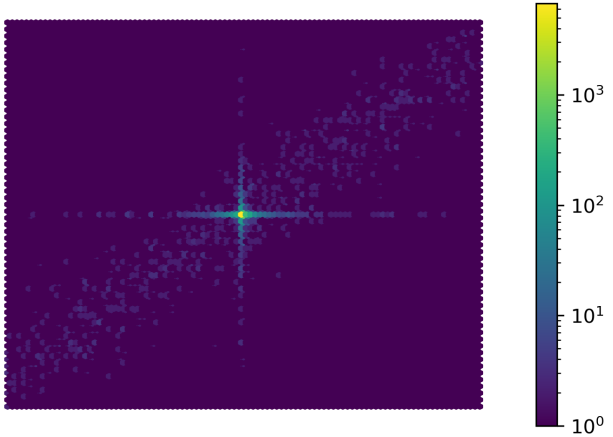


hexgrids

```
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)
```

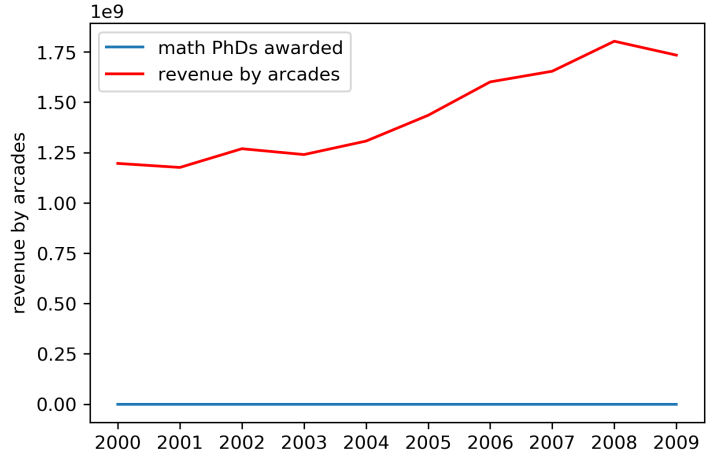



```
plt.figure()
plt.hexbin(x, y, bins='log', extent=(-1, 1, -1, 1))
plt.colorbar()
plt.axis("off")
```

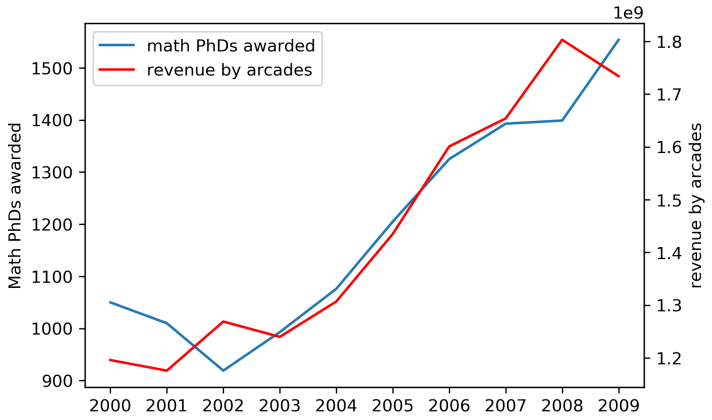


twinx, twiny

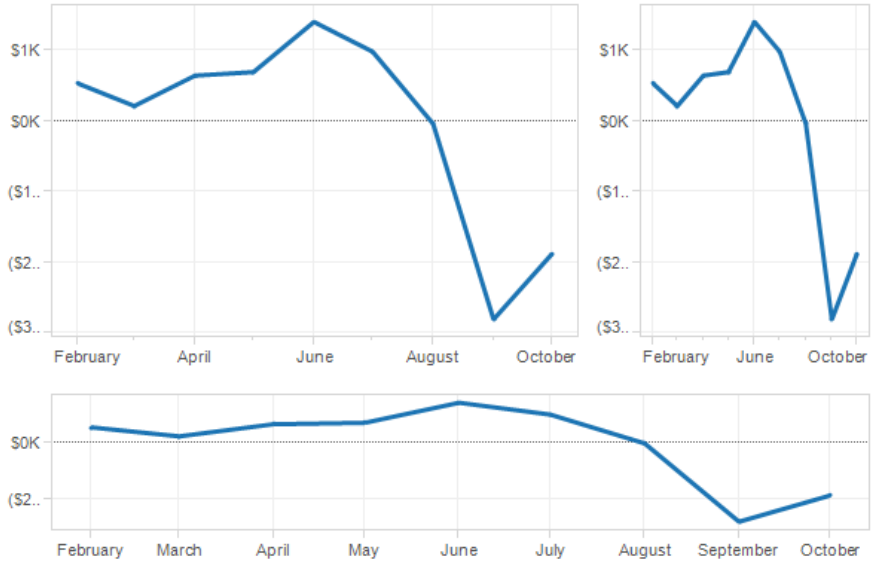
```
ax1 = plt.gca()
ax.plot(years, phds, label="math PhDs awarded")
ax.plot(years, revenue, c='r', label="revenue by arcades")
ax.set_ylabel("Math PhDs awarded")
ax.set_ylabel("revenue by arcades")
ax.legend()
```




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

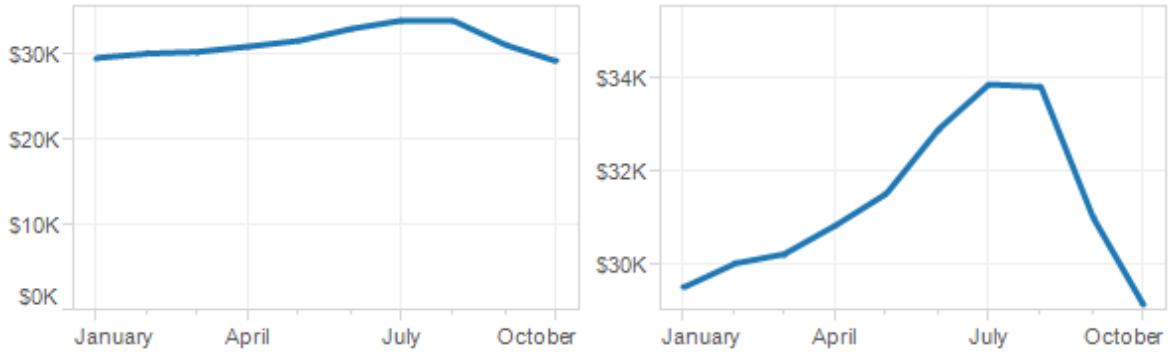


Aspect Ratios



https://eagereyes.org/basics/banking-45-degrees

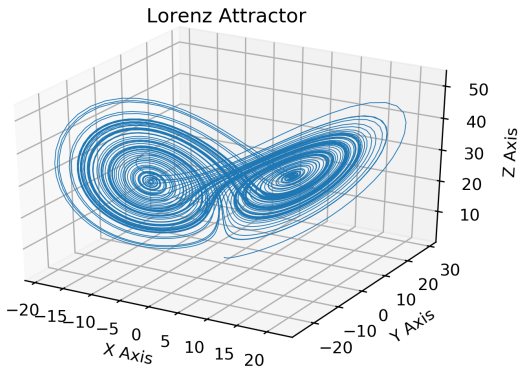
Baselines



https://eagereyes.org/basics/baselines

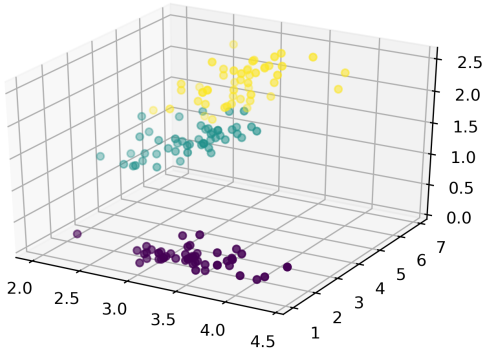
3D plotting

```
from mpl_toolkits.mplot3d import Axes3D

import matplotlib.pyplot as plt
import numpy as np

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

ax.plot(xs, ys, zs, lw=0.5)
```



3D Scatters

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

iris = load_iris()
X, y = iris.data, iris.target
fig = plt.figure()
ax = fig.add_subplot(
    111, projection='3d')
ax.scatter(X[:, 1], X[:, 2], X[:, 3],
           c=y)
```



Just Don't

Do's and Don't summarized

Do

- take your time
- label everything
- thing about questions and story

Do's and Don't summarized

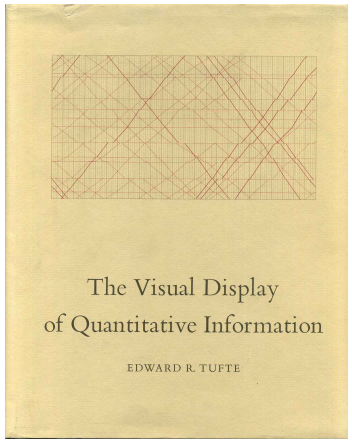
Do

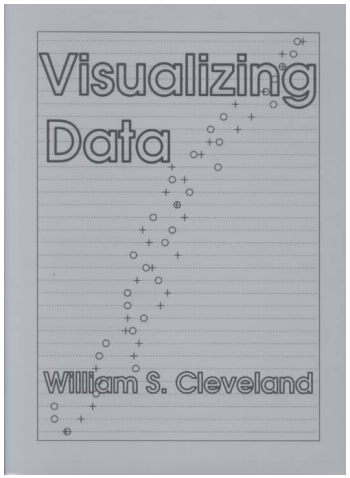
- take your time
- label everything
- thing about questions and story

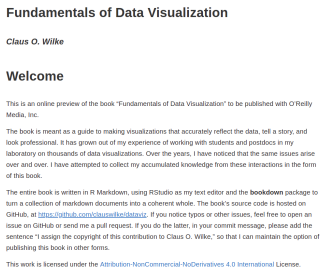
Don't

- Use 3D unless really necessary
- Use piecharts (usually)
- have non-varying "ink"
- Edit figure manually

More Resources






Fundamentals of Data Visualization

Claus O. Wilke

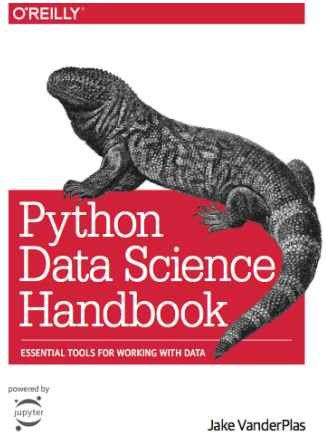
Welcome

This is an online preview of the book "Fundamentals of Data Visualization" to be published with O'Reilly Media, Inc.

The book is meant as a guide to making visualizations that accurately reflect the data, tell a story, and look professional. It has grown out of my experience of working with students and postdocs in my laboratory on thousands of data visualizations. Over the years, I have noticed that the same issues arise over and over. I have attempted to collect my accumulated knowledge from these interactions in the form of this book.

The entire book is written in R Markdown, using RStudio as my text editor and the **bookdown** package to turn a collection of markdown documents into a coherent whole. The book's source code is hosted on GitHub, at https://github.com/clauswilke/dataviz. If you notice typos or other issues, feel free to open an issue on GitHub or send me a pull request. If you do the latter, in your commit message, please add the sentence "I assign the copyright of this contribution to Claus O. Wilke," so that I can maintain the option of publishing this book in other forms.

This work is licensed under the Attribution-NonCommercial-NoDerivatives 4.0 International License.



Also: https://matplotlib.org/tutorials/

Questions ?