University of Zurich[UZH]

Department of Physics

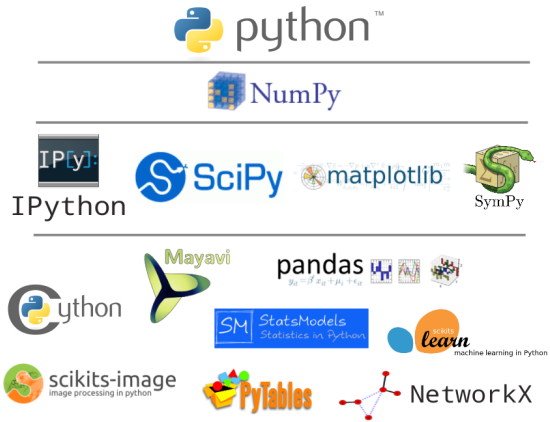
Scientific Programming: Data Visualisation and more

Scientific Programming with Python

Andreas Weiden

Based partially on a talk by Stéfan van der Walt

The Ecosystem of Homo Python Scientificus



[Ondřej Čertík/LANL]

Table of Contents

A Few Technical Remarks

If you want to follow directly the code used in the lecture

- Download the code from the course homepage (Lecture ?)
- Start the virtual environment
  `$ . venv/bin/activate` (from the home directory)
- Create a kernel for the notebook with the virtual environment
  `$ python3 -m ipykernel install --user --name=ve3`
- Unzip the file
  `$ tar zxvf material_visualization_lec.tar.gz`
- Enter the created directory
  `$ cd material_visualization_lec`
- ... and start the notebook
  `$ ipython3 notebook`

Visualisation

Visualisation as well as Content Matters

Visualisation Options in Python

Matplotlib

- Started as emulation for MATLAB
- Basic plotting also in more than one dimension

Seaborn

- Collection of more complex plots
- Based on Matplotlib

bokeh

- Web publishable graphics
- Large variety of usable interactions

Folium

- Python interface to leaflet (maps)
- Plotting of geo data

Color

Colour is a double-edged sword:

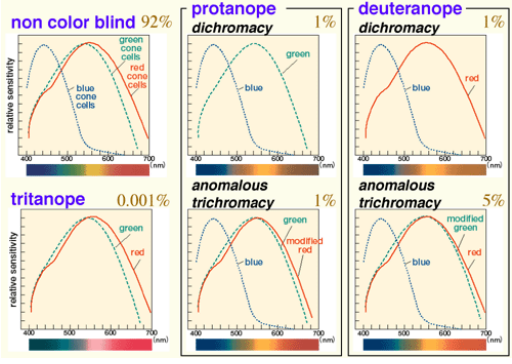
- Color can convey a lot of information
- But there are many forms of Color-blindness
- Many people will print your paper in black & white (for many reasons)

Two (non-exclusive) ways to deal with this:

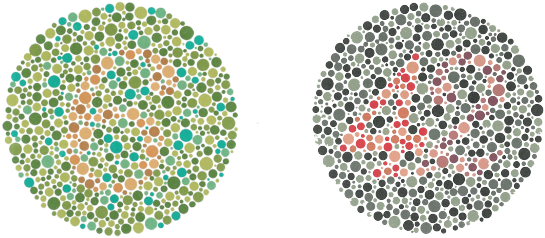
- Use Colors that are differentiable for all people and also in black & and white

Colorblindness

Colorblindness is not a total loss of color vision. Colorblind people can recognize a wide ranges of colors. But certain ranges of colors are hard to distinguish.



Colorblindness

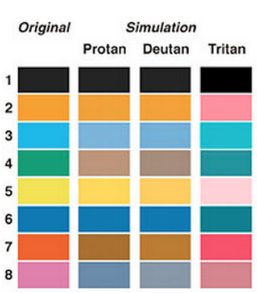


6 42

8% of Caucasian, 5% of Asian, and (4%) of African males are so-called "red-green" Colorblind. Chance to have at least one Colorblind reviewer out of three is up to $1 - (1 - 0.92)^3 = 22\%$!

Colorblindness

The way to deal with Colorblindess is to use redundant encoding of information
Most Colorblind people might not be able to distinguish certain colors, but are usually able to distinguish different brightness



	Original	Protan	Deutan	Tritan	Hue	for Photoshop, Illustrator, Freehand, etc. C,M,Y,K (%)	R,G,B (0-255)	for Word, Power Point, Canvas, etc. R,G,B (%)	Hex (0-f)
1					Black –	(0,0,0,100)	(0,0,0)	(0,0,0)	#000000
2					Orange 41°	(0,50,100,0)	(230,159,0)	(90,60,0)	#e69f00
3					Sky Blue 202°	(80,0,0,0)	(86,180,233)	(35,70,90)	#56b4e9
4					bluish Green 164°	(97,0,75,0)	(0,158,115)	(0,60,50)	#009e73
5					Yellow 56°	(10,5,90,0)	(240,228,66)	(95,90,25)	#f0e442
6					Blue 202°	(100,50,0,0)	(0,114,178)	(0,45,70)	#0072b2
7					Vermilion 27°	(0,80,100,0)	(213,94,0)	(80,40,0)	#d55e00
8					reddish Purple 326°	(10,70,0,0)	(204,121,167)	(80,60,70)	#cc799c

Use a color-palette taking advantage of this (either built-in or self-defined)

Defininig custom colors in matplotlib

Custom color palette

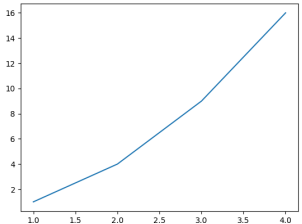
```
import matplotlib.pyplot as plt
from cycler import cycler

colors = ["#e69f00", "#56b4e9", "#009e73", "#d55e00", "#cc799c"]
plt.rc("axes", prop_cycle=cycler("color", colors))
```

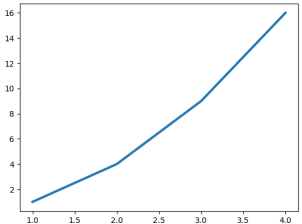
Texture

Use redundant coding. Not only Color, but also texture/patterns:

- Different markers
- Different line-styles
- Different filling-styles

Make plots visible enough using thick enough lines:

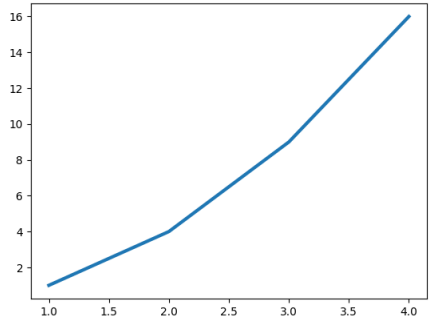




1D plotting

Line

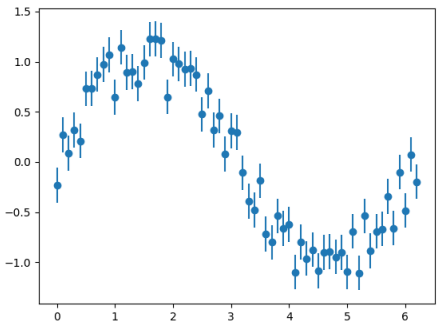
```
import matplotlib.pyplot as plt

x = np.array([1, 2, 3, 4])
y = x**2
plt.plot(x, y, linewidth=3)
plt.show()

df = pd.DataFrame({"x": x, "y":y})
df.plot("x", "y")
plt.show()
```



1D plotting

Error bars

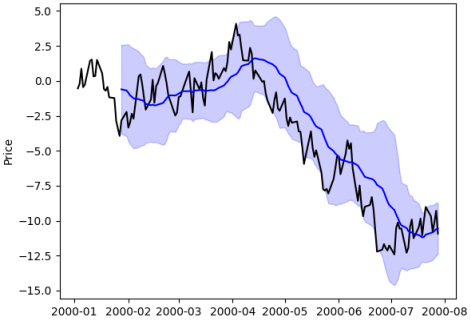
```
from numpy.random import random_sample

x = np.arange(0, 2*np.pi, 0.1)
a, b = -0.3, 0.3
noise = (b - a) * random_sample(x.shape) + a
y = np.sin(x) + noise
std = (b - a) / np.sqrt(12)
plt.errorbar(x, y, yerr=std, fmt="o")
```



1D plotting

Filling areas

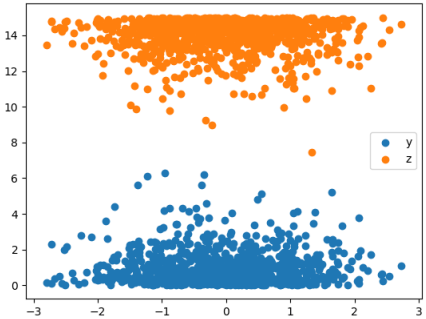
```
from numpy.random import randn

t = pd.date_range("2000-1-1", periods=150,
                  freq="B")
price = pd.Series(randn(150).cumsum(),
                  index=t)
avg = price.rolling(20).mean()
std = price.rolling(20).std()
plt.plot(price.index, price, "k")
plt.plot(avg.index, avg, "b")
plt.fill_between(std.index, avg-2*std,
                 avg+2*std, color="b",
                 alpha=0.2)
plt.ylabel("Price")
```



1D plotting

Scatter

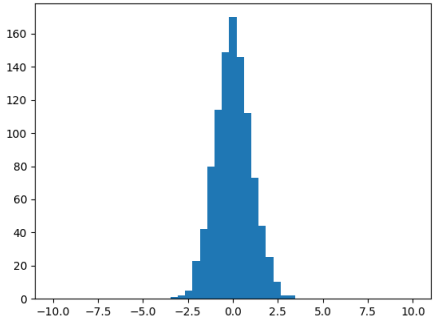
```
from numpy.random import normal
from numpy.random import exponential

x = randn(1000)
y = exponential(1, 1000)
z = 15 - exponential(1, 1000)
plt.scatter(x, y, label="y")
plt.scatter(x, z, label="z")
plt.legend()
plt.savefig("figs/plt_scatter.png")
```



1D plotting

Histograms

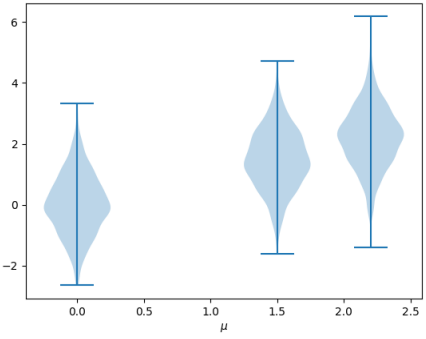
```
x = randn(1000)
bins = np.linspace(-10, 10)
plt.hist(x, bins=bins)
plt.show()
```



1D plotting

Violin plots

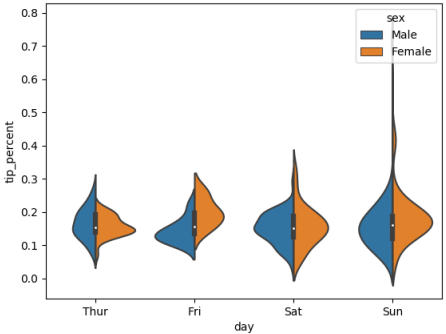
```
from numpy.random import normal

mus = 0, 1.5, 2.2
data = [normal(mu, 1, 1000) for mu in mus]
plt.violinplot(data, positions=mus)
plt.xlabel(r"$\mu$")
plt.show()
```



1D plotting

Split violin plots

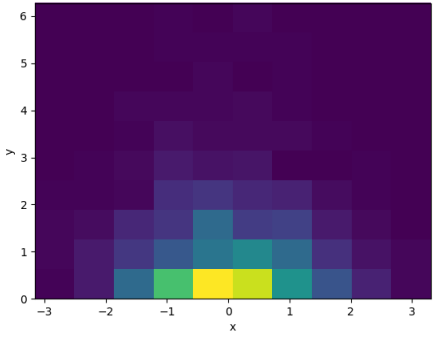
```
import seaborn as sns
tips = sns.load_dataset("tips")

tips["percent"] = tips.tip / tips.total_bill
sns.violinplot("day", "percent", "sex",
               data=tips, split=True)
```



2D plotting

2D histogram

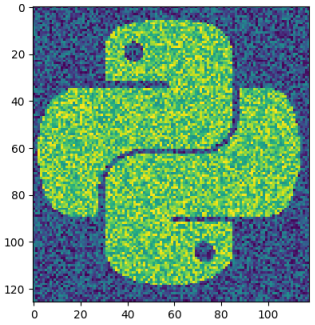
```
x = randn(1000)
y = exponential(size=1000)
plt.hist2d(x, y)
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```



2D plotting

Images

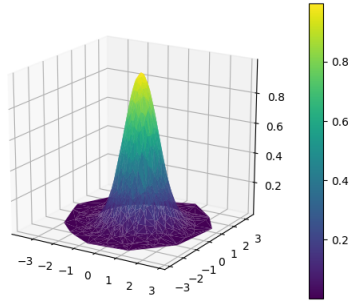
```
from numpy.random import rand
from scipy.ndimage import imread

path = "python.png"
img = imread(path, flatten=True)
data = rand(*img.shape)
data[img < 255] += 1
plt.imshow(data)
plt.show()
```



3D plotting

Surface plots

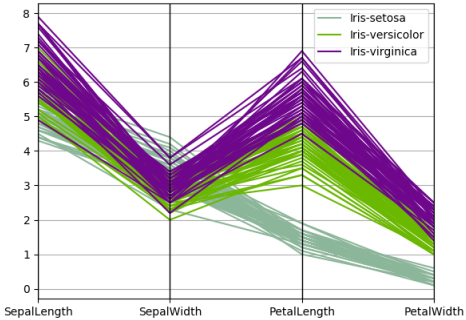
```
from mpl_toolkits.mplot3d import Axes3D

x = randn(1000)
y = randn(1000)
z = np.exp(-(x**2+y**2))
ax = plt.gca(projection="3d")
cmap = plt.cm.viridis
surf = ax.plot_trisurf(x, y, z, cmap=cmap)
plt.colorbar(surf)
```



ND plotting

Parallel coordinates

```
from pandas.plotting import parallel_coordin

data = pd.read_csv("data/iris.csv")
parallel_coordinates(data, "Name")
plt.show()
```



More tools

Argparse

Easy parsing of commandline options using argparse.

Argparse

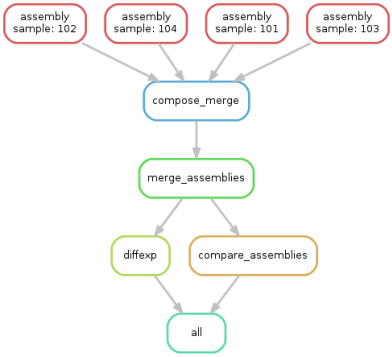
```
import argparse

parser = argparse.ArgumentParser(description="Process some integers.")
parser.add_argument("integers", metavar="N", type=int, nargs="+",
                    help="an integer for the accumulator")
parser.add_argument("--sum", dest="accumulate", action="store_const",
                    const=sum, default=max,
                    help="sum the integers (default: find the max)")

args = parser.parse_args()
print(args.accumulate(args.integers))

$ script.py --sum 1 2 3 4
10
```

Snakemake

Automate your analysis flow using snakemake.

Snakemake

```
rule targets:
    input:
        "plots/dataset1.pdf",
        "plots/dataset2.pdf"

rule plot:
    input:
        "raw/{dataset}.csv"
    output:
        "plots/{dataset}.pdf"
    shell:
        "somecommand {input} {output}"
```



Frameworks

Some fields have even created their own toolkits:

- Computational biology: https://biopython.org/
- Astronomy: http://www.astropy.org/
- High-energy particle physics: https://github.com/scikit-hep

Exercise

Take two of the graphs shown in this presentation and make them look good enough to be included in a scientific paper!

Ressources:

- Pyplot tutorial: https://matplotlib.org/users/pyplot_tutorial.html
- Matplotlib documentation: https://matplotlib.org/api/pyplot_summary.html
- Custom style-sheets: https://matplotlib.org/users/customizing.html
- Pandas plotting documentation: https://pandas.pydata.org/pandas-docs/stable/visualization.html
- Seaborn documentation: https://seaborn.pydata.org/

```
plt.style.use("./matplotlib_custom_style.mplstyle")
```