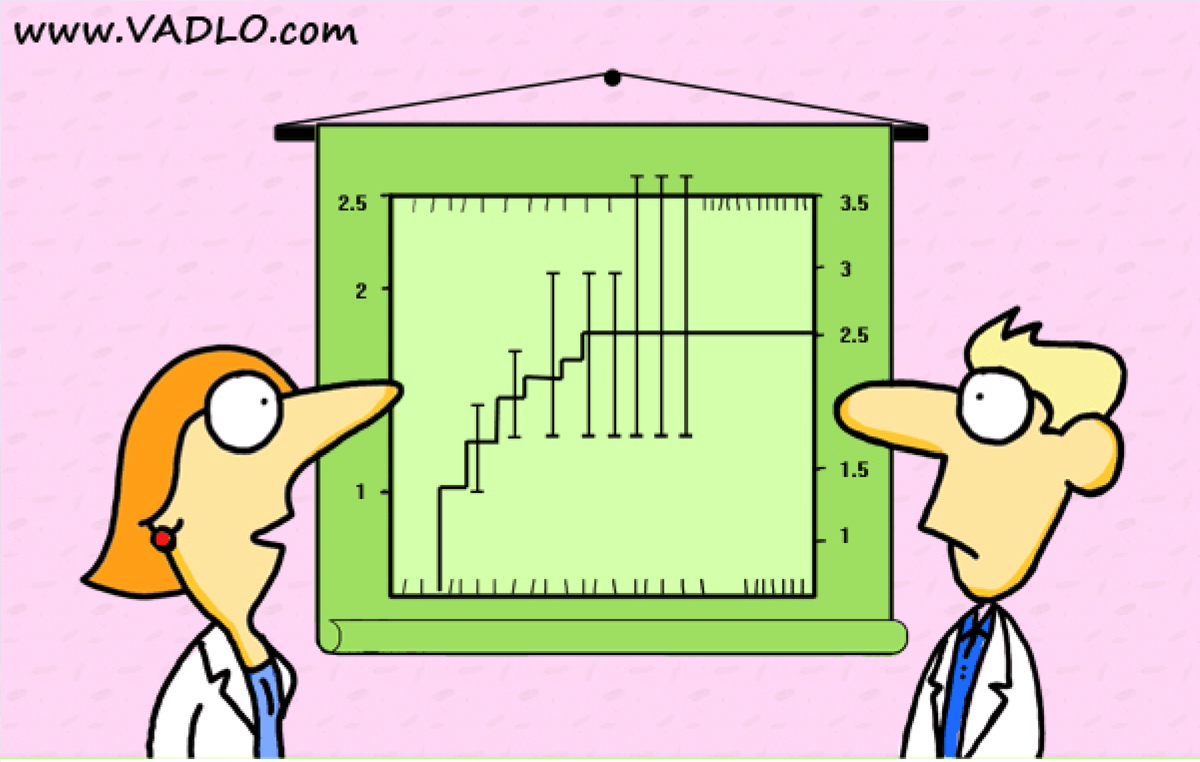
M. Hohle:

Physics 77: Introduction to Computational Techniques in Physics



“Did you really have to show the error bars?”

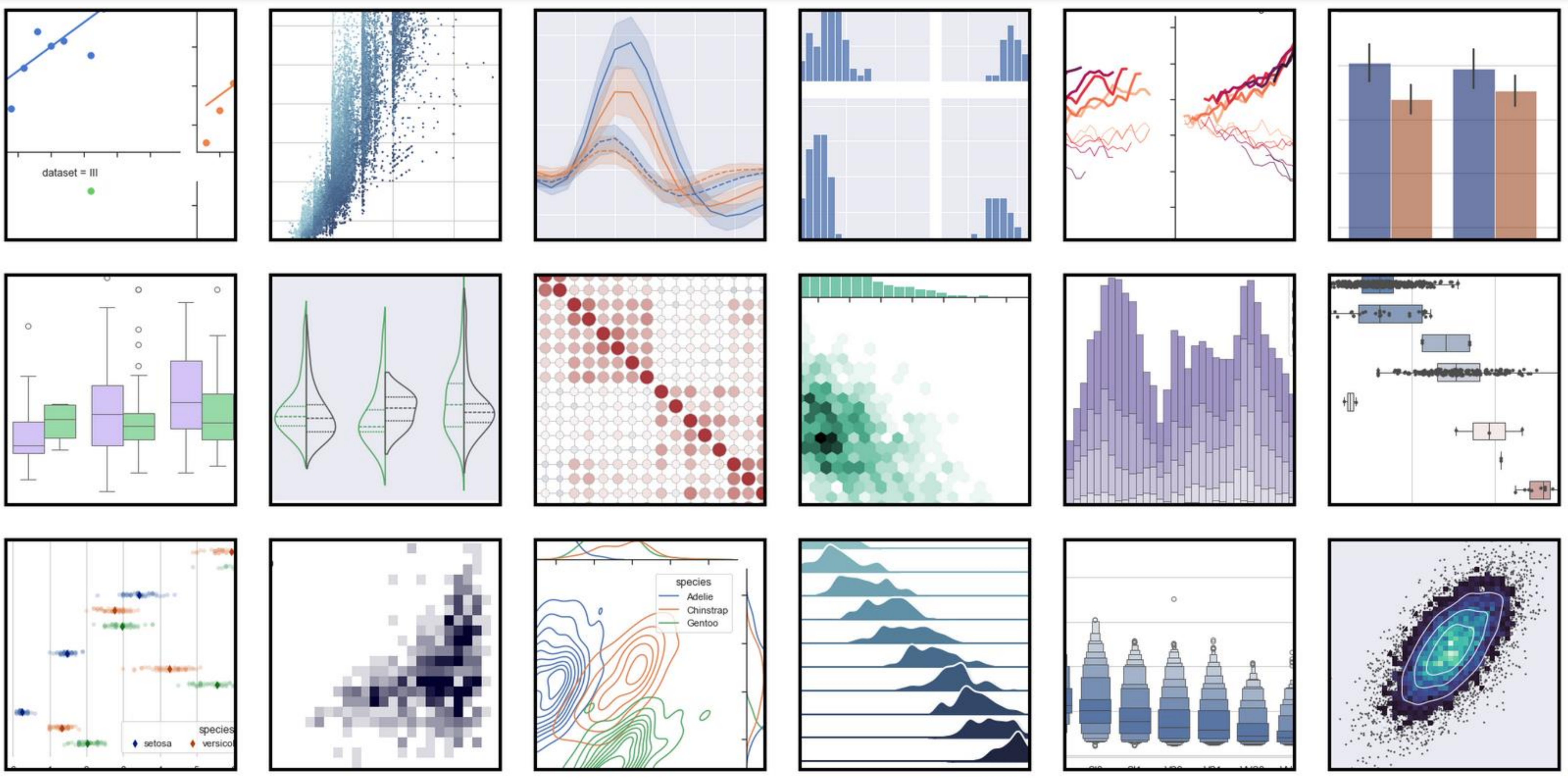
Introduction to Computational Techniques in Physics: syllabus

syllabus:

- Introduction to Unix & Python	(week 1 - 2)
- Functions, Loops, Lists and Arrays	(week 3 - 4)
- Visualization	**(week 5)**
- Parsing, Data Processing and File I/O	(week 6)
- Statistics and Probability, Interpreting Measurements	(week 7 - 8)
- Random Numbers, Simulation	(week 9)
- Numerical Integration and Differentiation	(week 10)
- Root Finding, Interpolation	(week 11)
- Systems of Linear Equations	(week 12)
- Ordinary Differential Equations	(week 13)
- Fourier Transformation and Signal Processing	(week 14)
- Capstone Project Presentations	(week 15)

Python Graph Gallery

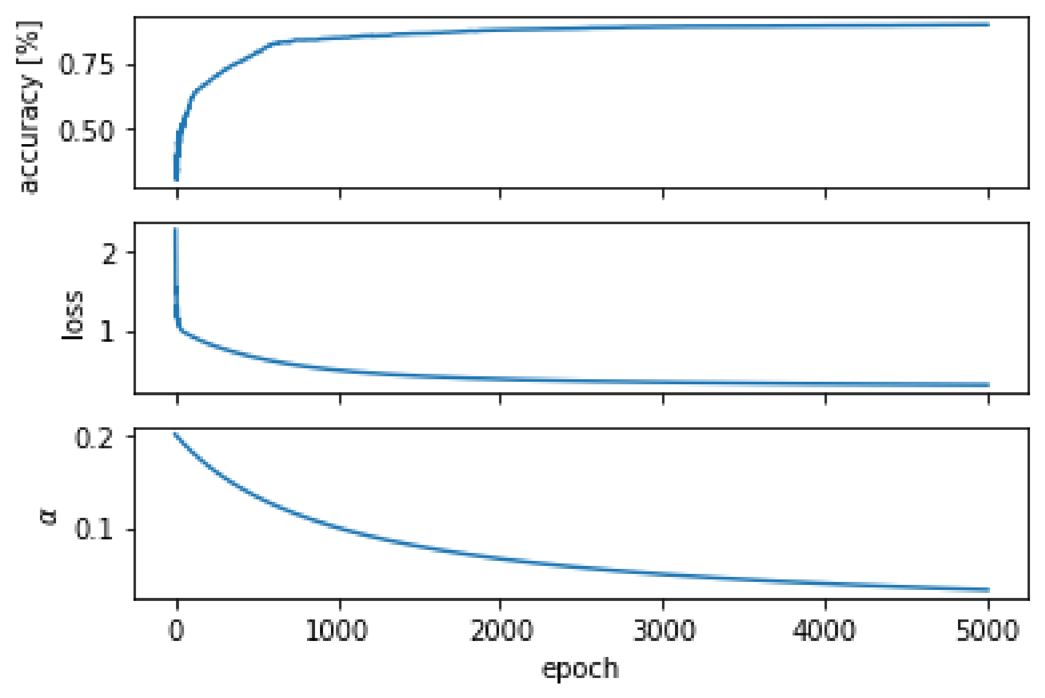
https://seaborn.pydata.org/examples/index.html

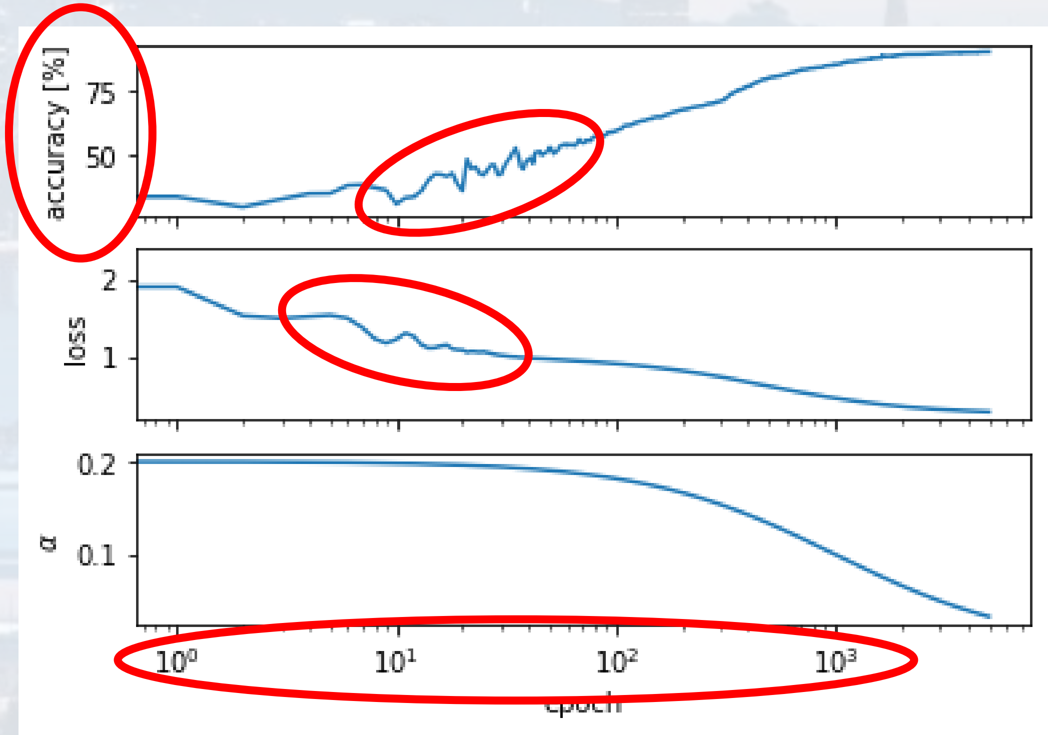


Introduction to Computational Techniques in Physics:

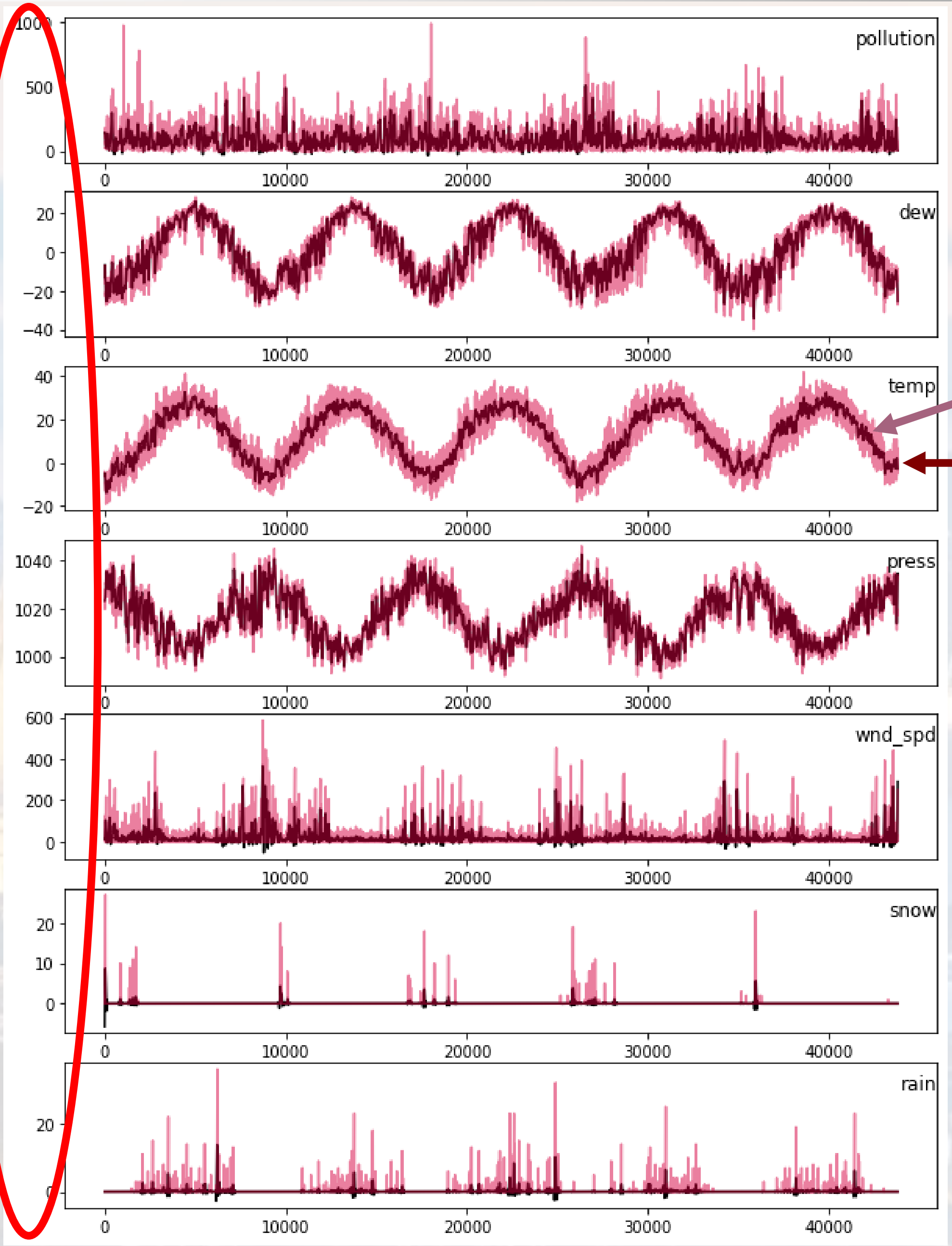
- **Matplotlib** → simple standard plots
- **Seaborn** → sophisticated plots

some rules:
- focus on what's **relevant**
- use **descent colors**
- axis **labels and units**
- scale **/ dynamic range**
- keep it as **simple** as possible





units?



actual data

moving average with exponential smoothing

transparency
alpha = 0….1

Let us create a random dataset first:

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

M = 25
N = 30

X = np.zeros((N,M))

for i in range(M):
    X[:,i] = np.random.normal(np.random.uniform(-10,10),\
                              np.random.uniform(0,10),(N,1))[:,0]

Y = np.cos(X)

x = np.sort(X[:,1])
y = np.sort(Y[:,1])
```

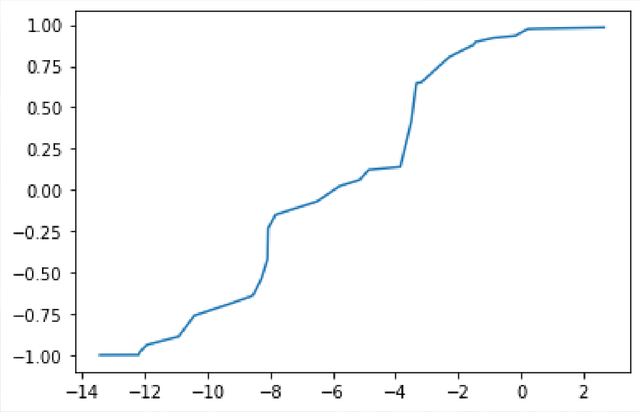
calling standard libraries

creating random data

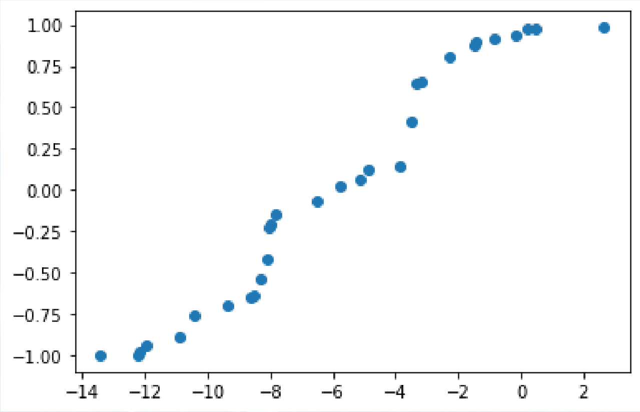
- **Matplotlib** → simple standard plots
- **Seaborn** → sophisticated plots

- **Matplotlib** → simple standard plots
- **Seaborn** → sophisticated plots

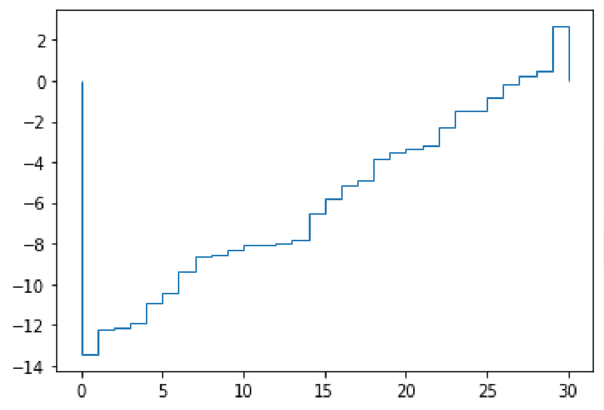
`plt.plot(x, y)`



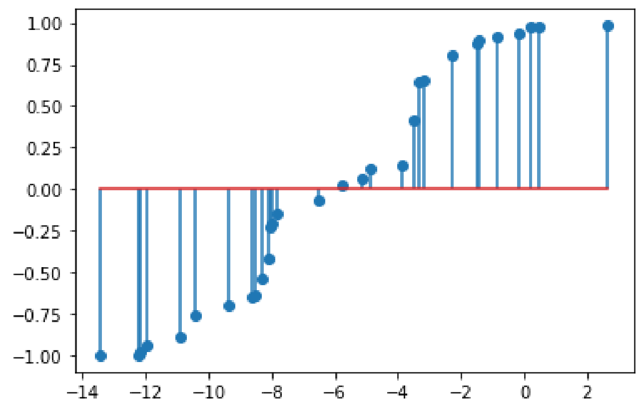
`plt.scatter(x, y)`



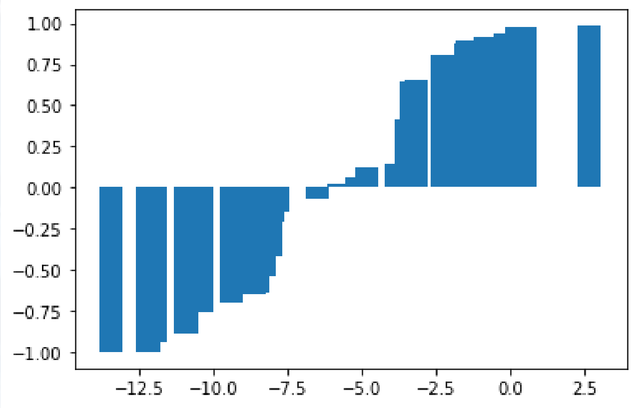
`plt.stairs(x)`



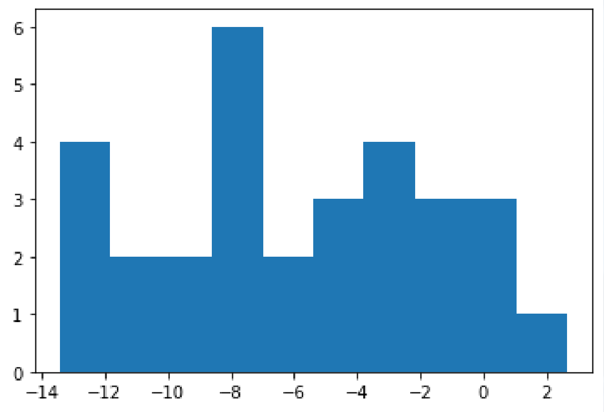
`plt.stem(x,y)`



`plt.bar(x, y)`



`plt.hist(x)`



```
plt.scatter(x, y, kwargs)
```

```
scatter(x: 'float | ArrayLike', y: 'float | ArrayLike', s:
        'float | ArrayLike | None'=None, c: 'ArrayLike |
        Sequence[ColorType] | ColorType | None'=None,
        marker: 'MarkerType | None'=None, cmap: 'str |
        Colormap | None'=None, norm: 'str | Normalize |
        None'=None, vmin: 'float | None'=None, vmax: 'float
        | None'=None, alpha: 'float | None'=None,
        linewidths: 'float | Sequence[float] | None'=None,
        *, edgecolors: "Literal['face', 'none'] | ColorType
        | Sequence[ColorType] | None"=None, plotnonfinite:
        'bool'=False, data=None, **kwargs)

A scatter plot of *y* vs. *x* with varying marker size
and/or color.

Parameters
----------
x, y : float or array-like, shape (n, )
The data positions.

s : float or array-like, shape (n, ), optional
The marker size in points**2 (typographic points are ...
```

```python
plt.scatter(x, y,
            marker = 'p',
            s = 135,
            color = [0.5, 0.1, 0.1],
            edgecolor = 'black',
            alpha = 0.3)
```

marker style: str

marker	symbol	description
"."	•	point
","	·	pixel
"o"	●	circle
"v"	▼	triangle_down
"^"	▲	triangle_up
"<"	◀	triangle_left
">"	▶	triangle_right
"1"	⋎	tri_down
"2"	⋏	tri_up
"3"	≺	tri_left
"4"	≻	tri_right
"8"	⯃	octagon
"s"	■	square
"p"	⬟	pentagon
"P"	✚	plus (filled)
"*"	★	star
"h"	⬢	hexagon1
"H"	⬣	hexagon2
"+"	+	plus
"x"	×	x
"X"	✖	x (filled)
"D"	◆	diamond
"d"	♦	thin_diamond
"|"	|	vline
"_"	—	hline
0 (TICKLEFT)	–	tickleft
1 (TICKRIGHT)	–	tickright
2 (TICKUP)	'	tickup
3 (TICKDOWN)	'	tickdown
4 (CARETLEFT)	◀	caretleft
5 (CARETRIGHT)	▶	caretright
6 (CARETUP)	▲	caretup
7 (CARETDOWN)	▼	caretdown
8 (CARETLEFTBASE)	◀	caretleft (centered at base)
9 (CARETRIGHTBASE)	▶	caretright (centered at base)
10 (CARETUPBASE)	▲	caretup (centered at base)
11	▼	caretdown (centered at base)

```
plt.scatter(x, y,
            marker = 'p',
            s = 135,
            color = [0.5, 0.1, 0.1],
            edgecolor = 'black',
            alpha = 0.3)
```

marker size in pixel: `int`

```
plt.scatter(x, y,
            marker = 'p',
            s = 135,
            color = [0.5, 0.1, 0.1],
            edgecolor = 'black',
            alpha = 0.3)
```

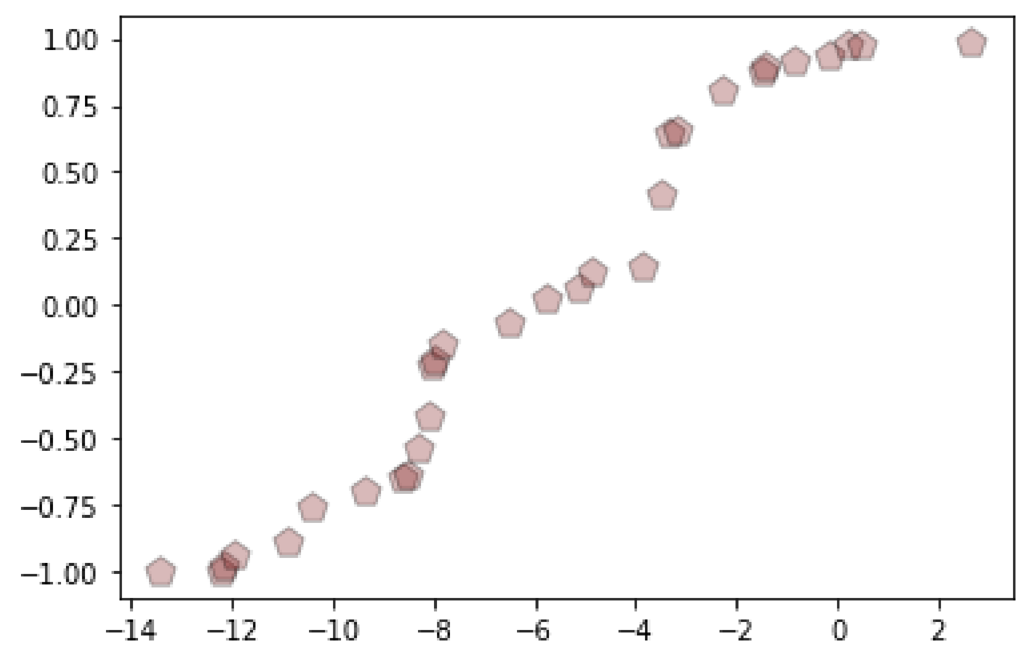
color:

`array`	RGB code if **three** values, RGB code plus alpha, if **four** values
`str`	full string: *'green'* *'yellow'* abbreviation: *'g'* *'y'* HEX code: *'#4b8333'* *'#fff8de'*

```
plt.scatter(x, y,
            marker = 'p',
            s = 135,
            color = [0.5, 0.1, 0.1],
            edgecolor = 'black',
            alpha = 0.3)
```

alpha (opaqueness/opacity): `int`

```python
plt.scatter(x, y,
            marker = 'p',
            s = 135,
            color = [0.5, 0.1, 0.1],
            edgecolor = 'black',
            alpha = 0.3)
```



```
plt.scatter(x, y, marker = 'p', s = 135, color = [0.5, 0.1, 0.1],
                   edgecolor = 'black', alpha = 0.3)

plt.xlabel(r'x values $\tau^{ij}_{def}$')
plt.ylabel('y values')
plt.title('first plot')
plt.legend(['data'])
#plt.xscale('log')
plt.savefig('new_plot.pdf')
plt.show()
```

Python speaks LaTeX, but needs full string

```
plt.scatter(x, y, marker = 'p', s = 135, color = [0.5, 0.1, 0.1],
                  edgecolor = 'black', alpha = 0.3)

plt.xlabel(r'x values $\tau^{ij}_{def}$')
plt.ylabel('y values')
plt.title('first plot')
plt.legend(['data'])
#plt.xscale('log')
plt.savefig('new_plot.pdf')
plt.show()
```

legend needs to be a `list`

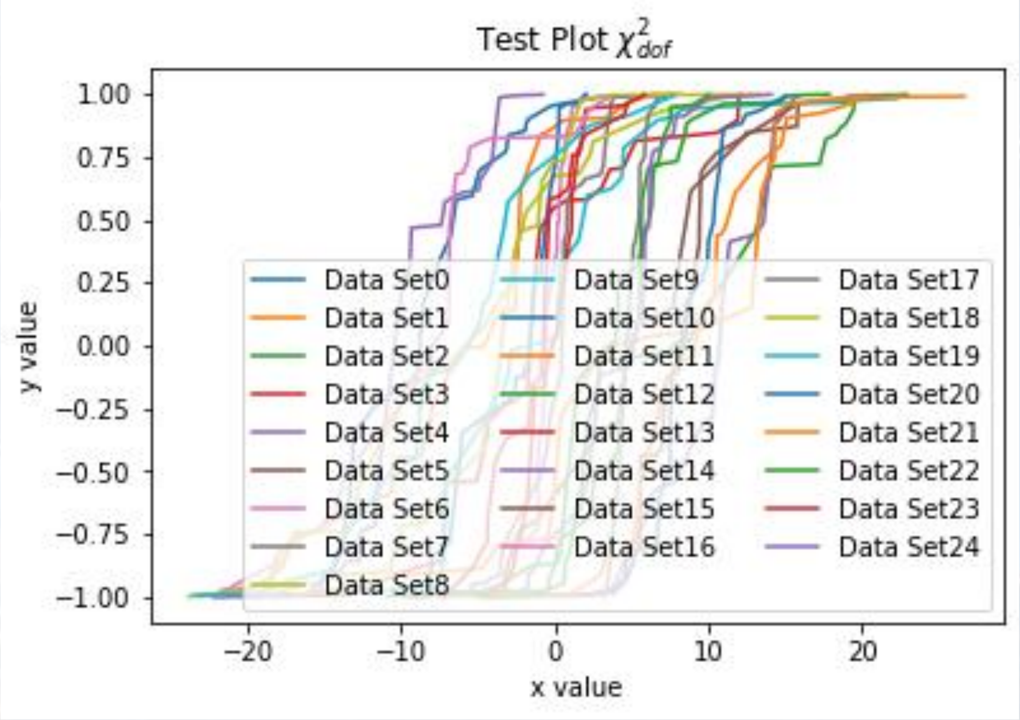
```
plt.scatter(x, y, marker = 'p', s = 135, color = [0.5, 0.1, 0.1],
                    edgecolor = 'black', alpha = 0.3)

plt.xlabel(r'x values $\tau^{ij}_{def}$')
plt.ylabel('y values')
plt.title('first plot')
plt.legend(['data'])
#plt.xscale('log')
plt.savefig('new_plot.pdf')
plt.show()
```

plots can be saved to any common format

```
plt.scatter(x, y, marker = 'p', s = 135, color = [0.5, 0.1, 0.1],
                  edgecolor = 'black', alpha = 0.3)

plt.xlabel(r'x values $\tau^{ij}_{def}$')
plt.ylabel('y values')
plt.title('first plot')
plt.legend(['data'])
#plt.xscale('log')
plt.savefig('new_plot.pdf')
plt.show()
```

sometims plots don‘t show up
(depending on settings)
type `plt.show()`
at the **very end**

```
for i in range(M):
    plt.plot(np.sort(X[:,i]), np.sort(Y[:,i]))

plt.xlabel("x value")
plt.ylabel("y value")
plt.title("Test Plot $\chi^2_{dof}$")
plt.legend(['Data Set ' + str(i) for i in range(M)],\
           loc = 'lower right', ncol = 3)
plt.show()
```

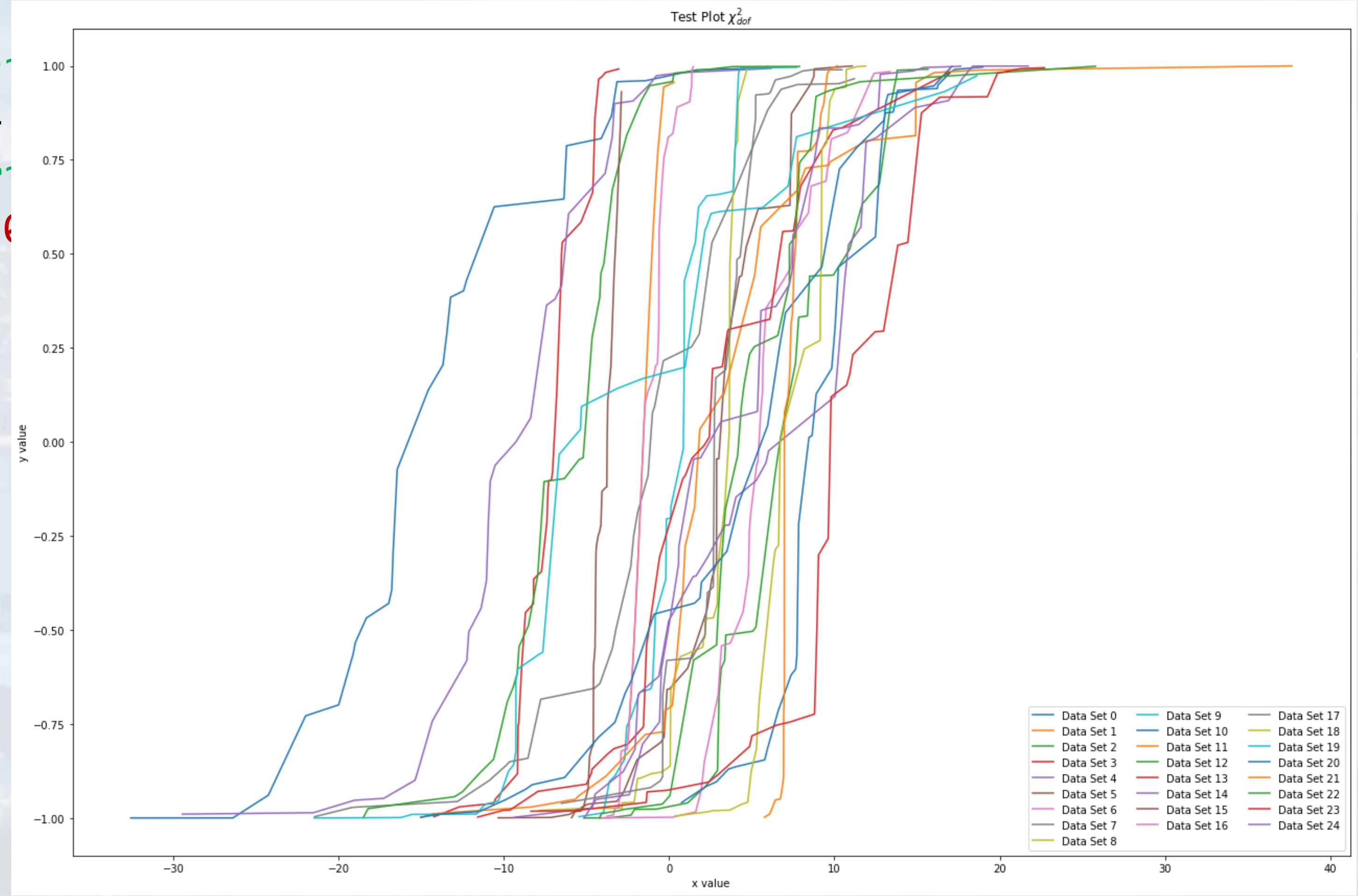


```
for i in range(M):
    plt.plot(np.sort(X[:,i]), np.sort(Y[:,i]))

plt.xlabel("x value")
plt.ylabel("y value")
plt.title("Test Plot $\chi^2_{dof}$")
plt.legend(['Data Set ' + str(i) for i in range(M)],\
           loc = 'lower right', ncol = 3)
plt.tight_layout(rect = [0, 0, 3, 3])
plt.show()
```

Berkeley UNIVERSITY OF CALIFORNIA

Introduction to Computational Techniques in Physics: Visualization

```
for i in range(M):
    plt.plot(np.sort(X[:,i]), np.sort(Y[:,i]))

plt.xlabel("x value")
plt.ylabel("y value")
plt.title("Test Plot $\ch
plt.legend(['Data Set ' +
              loc = 'lower r
plt.tight_layout(rect = [0
plt.show()
```

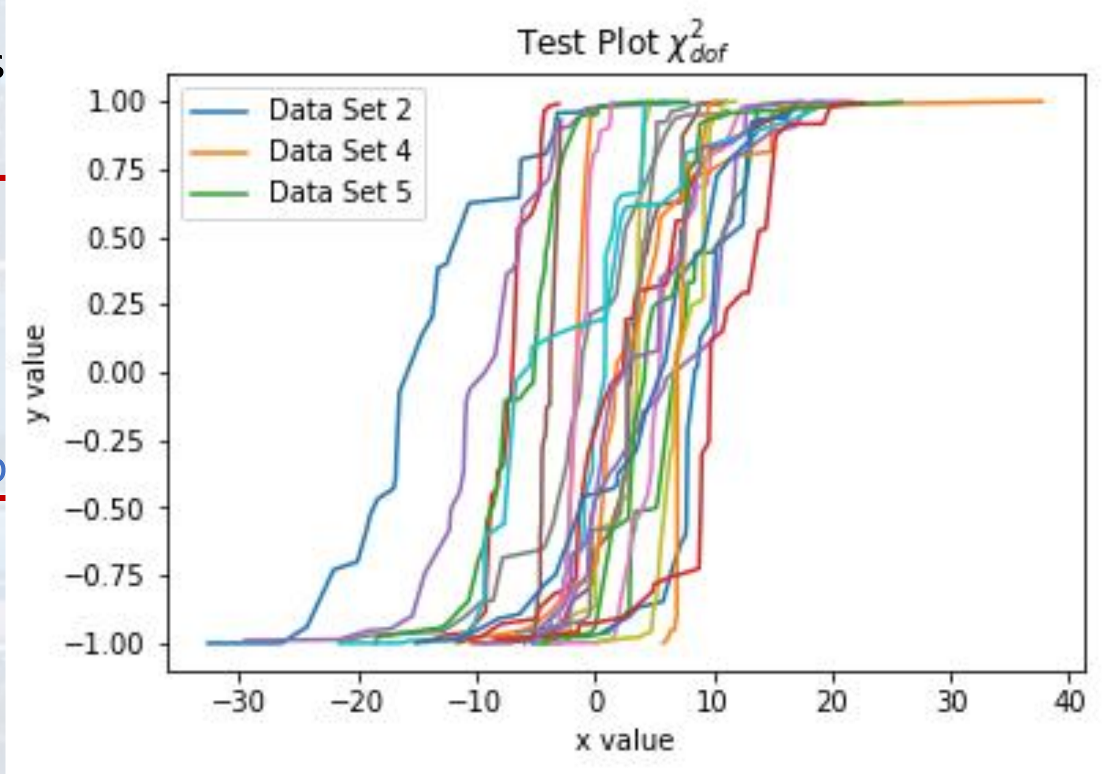


display only specific legends

```
H      = []
Filter = [2,4,5]


for i in range(M):
    h = plt.plot(np.sort(X[:,i]), np.sort(Y[:,i]))
    if i in Filter:
        H += h

plt.xlabel("x value")
plt.ylabel("y value")
plt.title("Test Plot $\chi^2_{dof}$")
plt.legend(H, ['Data Set ' + str(i) for i in Filter])
```

display only specific legends

```
H      = []
Filter = [2,4,5]


for i in range(M):
    h = plt.plot(np.sort(X[:,i]), np.s
    if i in Filter:
       H += h

plt.xlabel("x value")
plt.ylabel("y value")
plt.title("Test Plot $\chi^2_{dof}$")
plt.legend(H, ['Data Set ' + str(i) fo
```

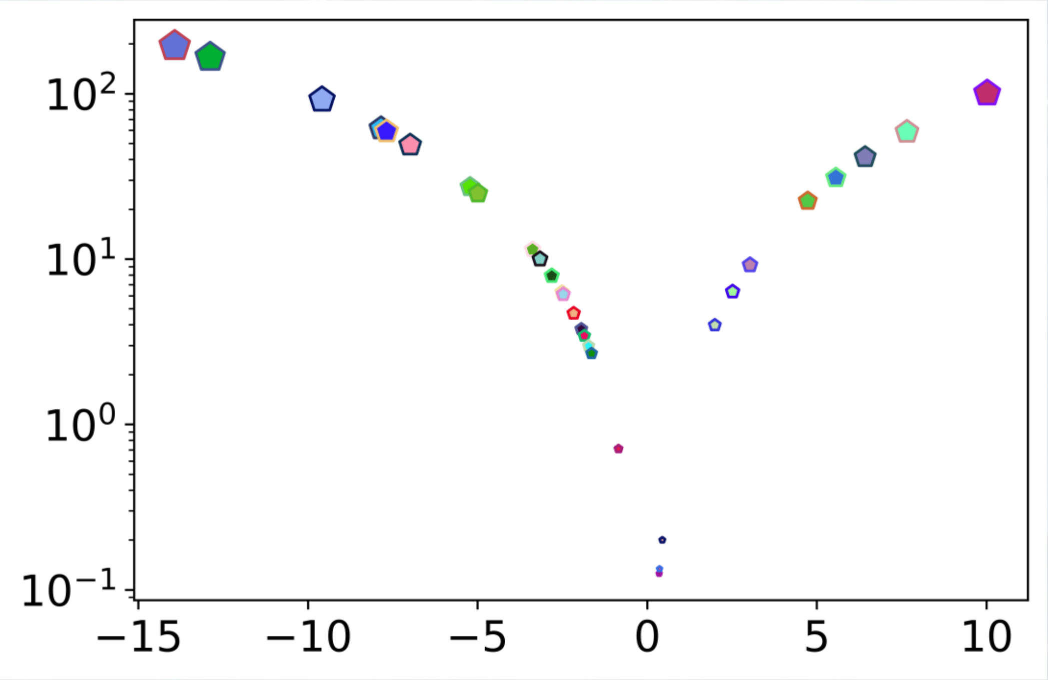


```
rgb_color      = np.random.uniform(0,1,(N,3))
x              = np.sort(X[:,1])
rgb_edge_color = np.random.uniform(0,1,(N,3))
size           = abs(10*x)
m              = 'p'

figX = plt.figure()

plt.scatter(x, x**2, c = rgb_color, edgecolors = rgb_edge_color, marker = m,\
            s = size)
plt.yscale('log')
plt.show()

figX.savefig('test.pdf')
```

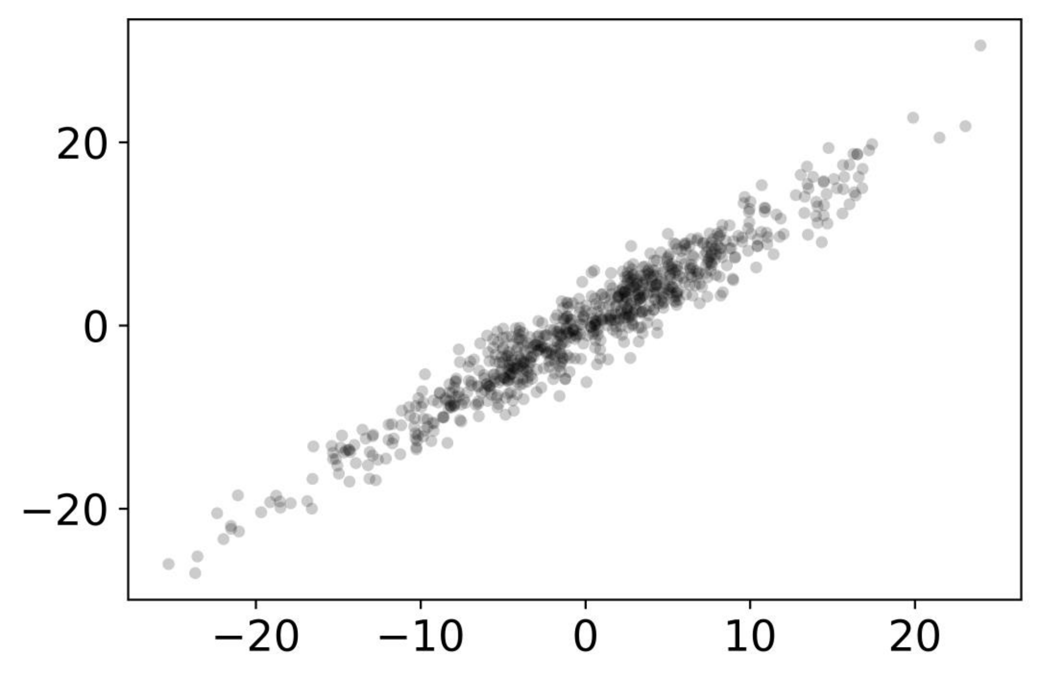
refering to a figure I'd like to save

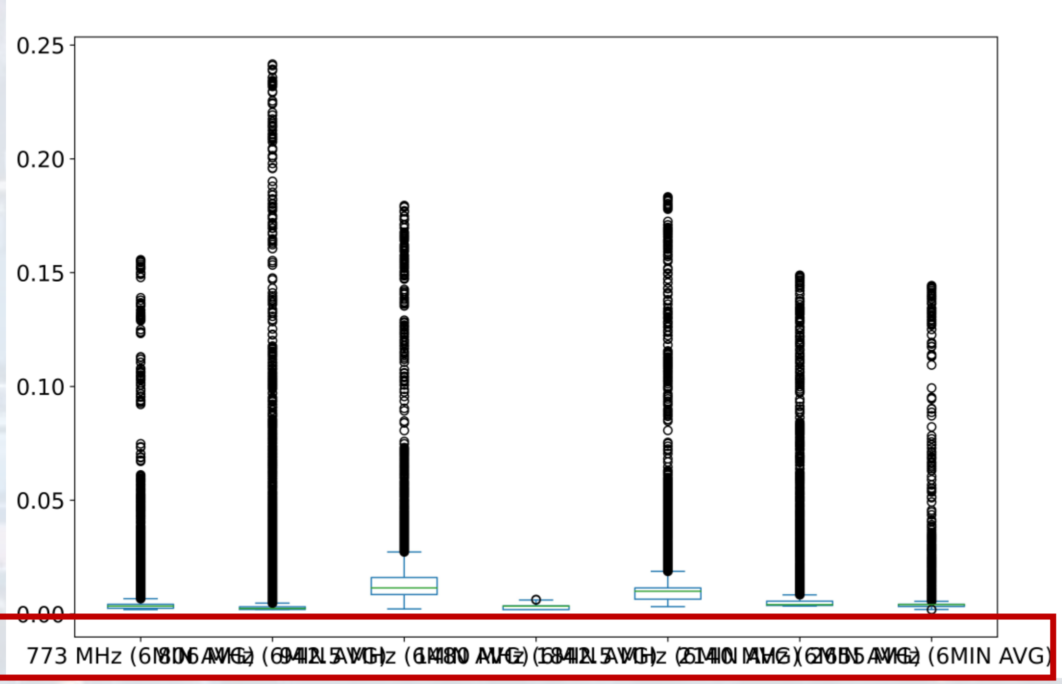
saving the figure

```
rgb_color      = np.random.uniform(0,1,(N,3))
x              = np.sort(X[:,1])
rgb_edge_color = np.random.uniform(0,1,(N,3))
size           = abs(10*x)
m              = 'p'

figX = plt.figure()

plt.scatter(x, x**2, c = rgb_color, edgecolors = rgb_edge_color, marker = m,\
            s = size)
plt.yscale('log')
plt.show()

figX.savefig('test.pdf')
```



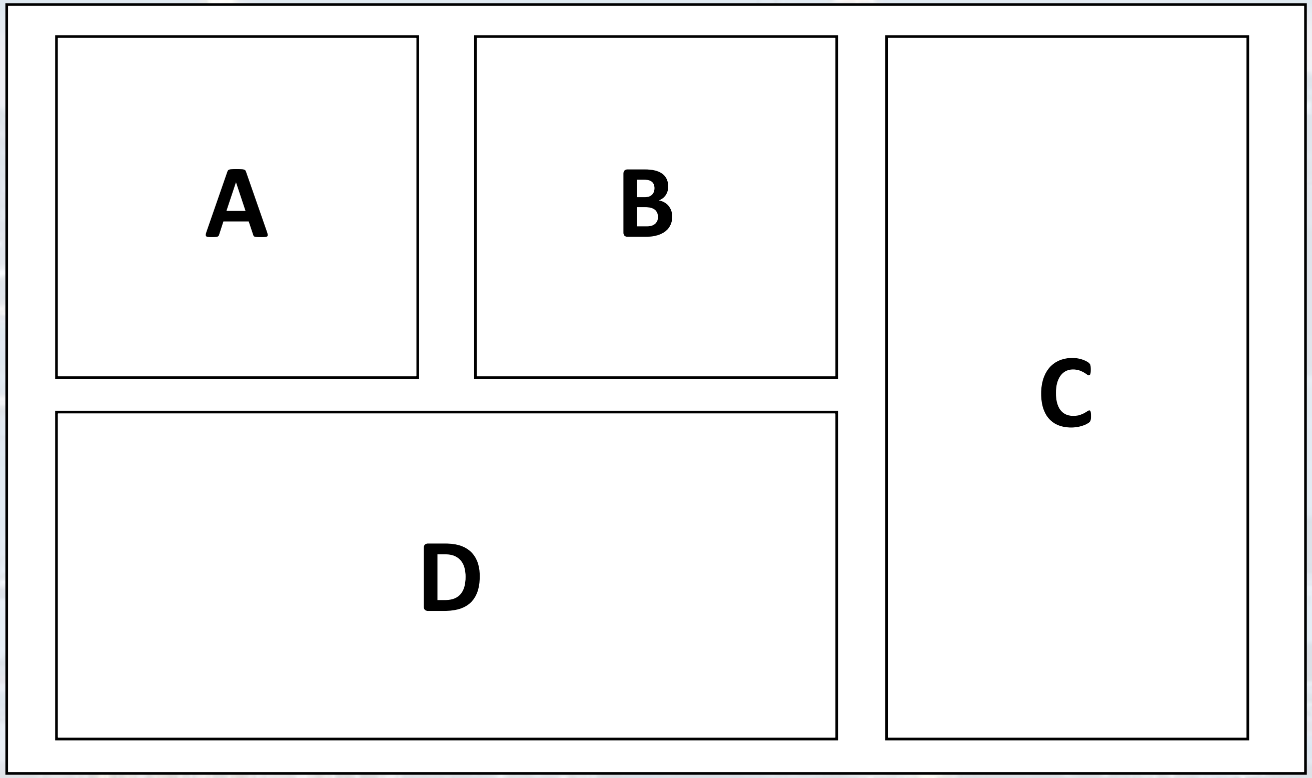
```
xr = X.reshape(N*M,1)
yr = xr + np.random.normal(0,2,(N*M,1))

plt.scatter(xr, yr, s = 20, c = 'k', alpha = 0.2, edgecolors = 'none')

plt.xticks(rotation = 45)
#labels = ['A', 'B'], ticks = [-20, -10])
```





subplots

```
figMo, axes = plt.subplot_mosaic([['A', 'B', 'C'],\
                                  ['D', 'D', 'C']], layout = "constrained")
```

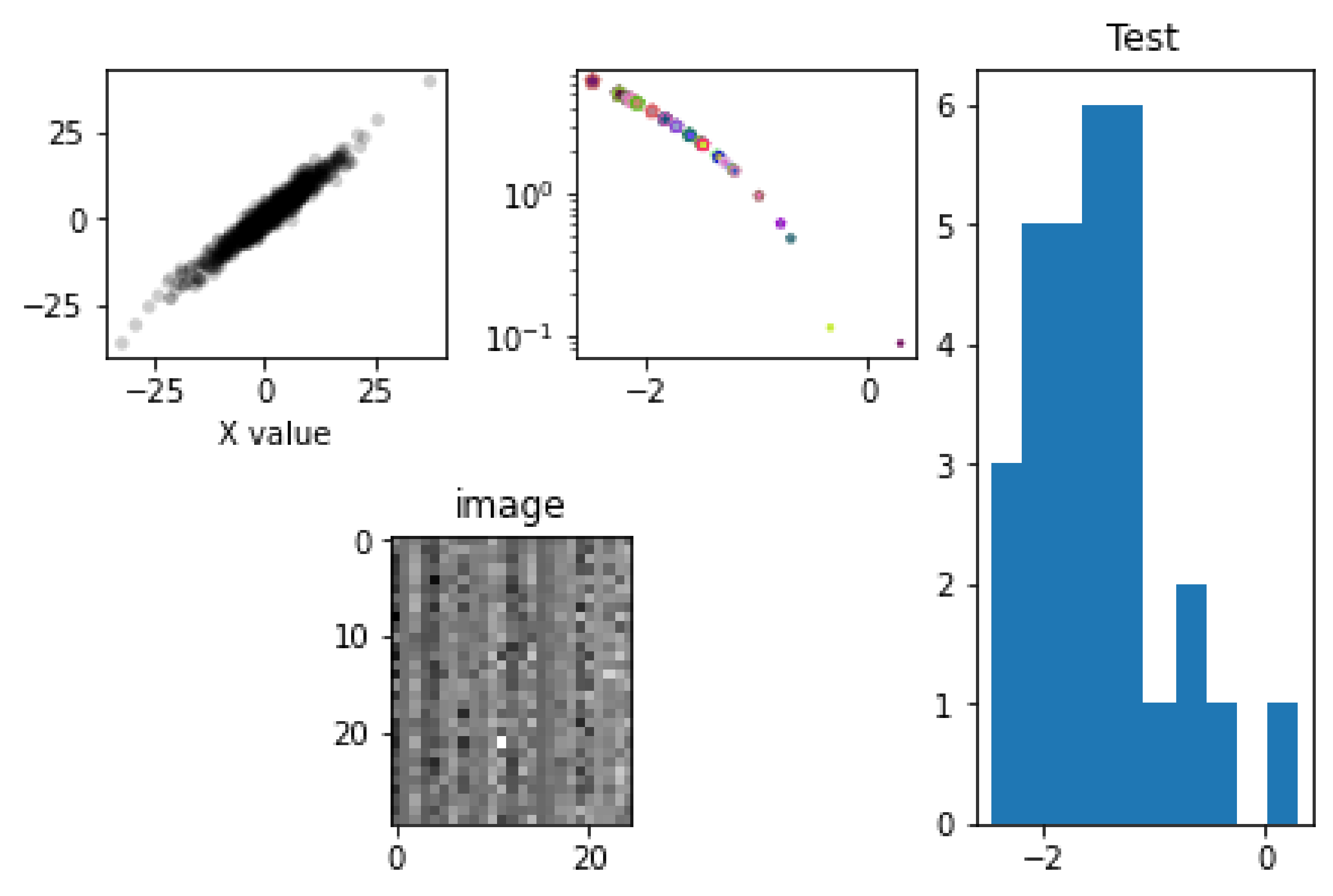


subplots

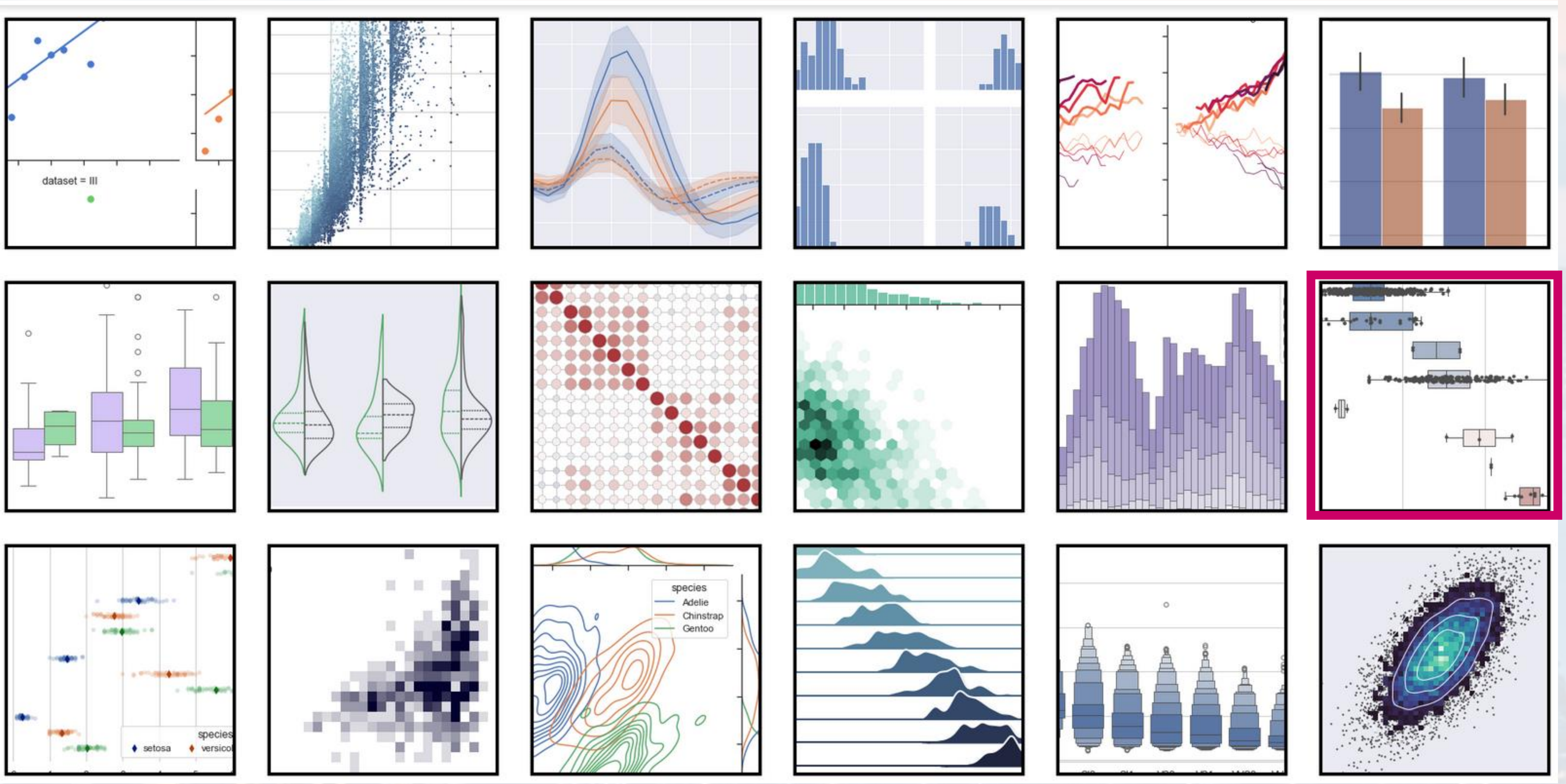
```
figMo, axes = plt.subplot_mosaic([['A', 'B', 'C'],\
                                 ['D', 'D', 'C']], layout = "constrained")

axes['A'].scatter(xr, yr, s = 20, c = 'k', alpha = 0.2, edgecolors = 'none')
axes['A'].set(xlabel = 'X value')

axes['B'].scatter(x, x**2, c = rgb_color, edgecolors = rgb_edge_color,\
                      marker = m, s = size)
axes['B'].set(yscale = 'log')

axes['C'].hist(x)
axes['C'].set(title = 'Test')

axes['D'].imshow(X, cmap = 'gray')
axes['D'].set(title = 'image')
plt.show()

figMo.savefig('test.pdf')
```

subplots

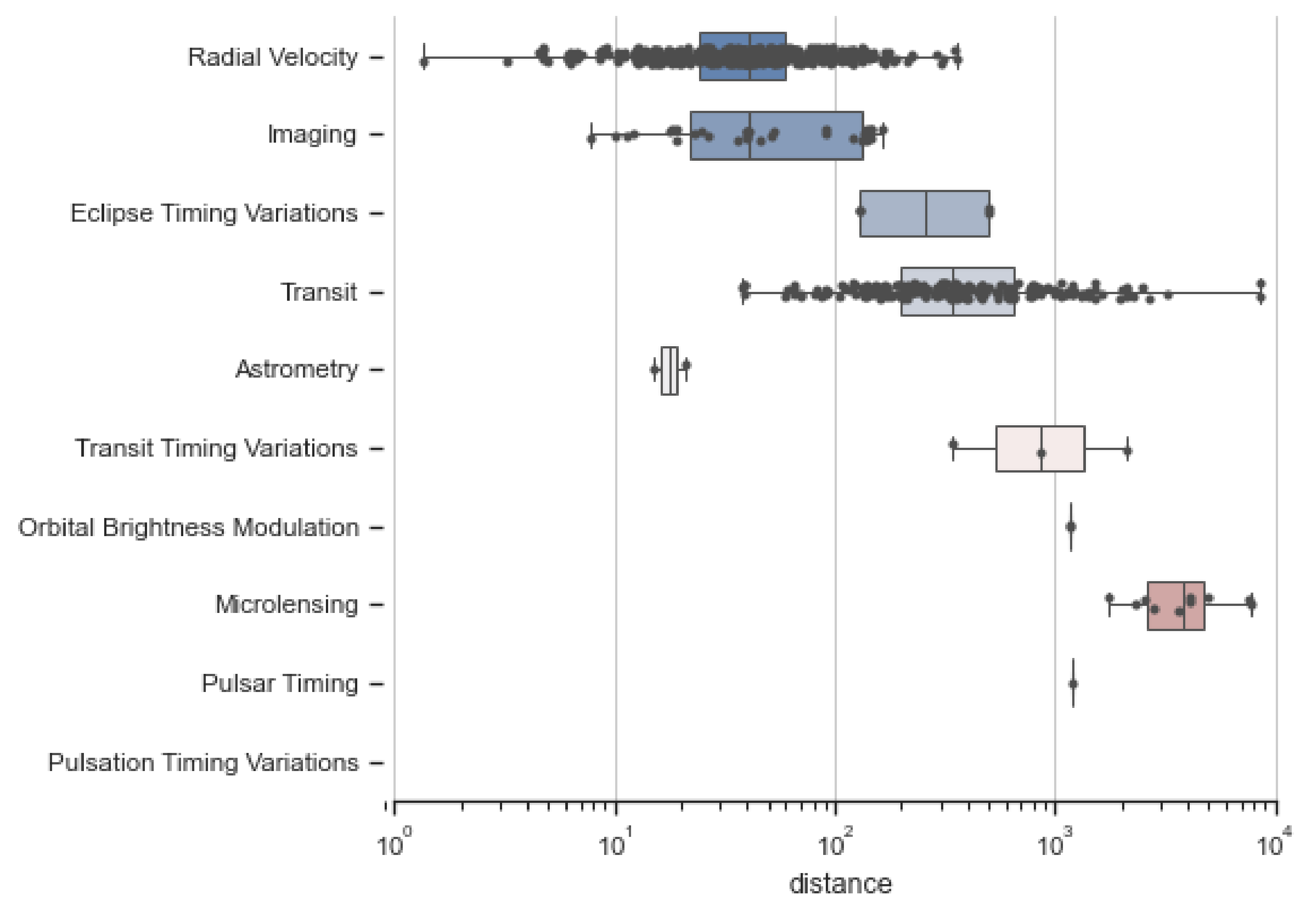
```
figMo, axes = plt.subplot_mosaic([['A', 'B', 'C'],\

axes['A'].scatter(xr, yr
axes['A'].set(xlabel = '

axes['B'].scatter(x, x**
                  ma
axes['B'].set(yscale = '

axes['C'].hist(x)
axes['C'].set(title = 'T

axes['D'].imshow(X, cmap
axes['D'].set(title = 'i
plt.show()

figMo.savefig('test.pdf'
```



- **Matplotlib** → simple standard plots
- **Seaborn** → sophisticated plots



- **Matplotlib** → simple standard plots
- **Seaborn** → sophisticated plots



exoplanets data set

- **Matplotlib** → simple standard plots
- **Seaborn** → sophisticated plots

```
import seaborn as sns

import pandas as pd
```

standard library for *data frames* **(more: next lecture)**

```
X_df         = pd.DataFrame(X)
X_df.index   = {'Data Set ' + str(i): i for i in range(N)}
X_df.columns = ['t' + str(i) for i in range(M)]
```

Index	t0	t1	t2
Data Set 0	-8.00758	6.33521	-0.105416
Data Set 1	-10.1712	15.6568	4.80345
Data Set 2	0.88356	8.20249	2.46666
Data Set 3	-6.58955	15.8528	4.95933

- **Matplotlib** → simple standard plots
- **Seaborn** → sophisticated plots

```
X_df         = pd.DataFrame(X)
X_df.index   = {'Data Set ' + str(i): i for i in range(N)}
X_df.columns = ['t' + str(i) for i in range(M)]
```

<u>important commands:</u>

`X_df.values`	extracts only values as `np.array`
`X_df.corr()`	calculates correlation coefficients
`X_df.index`	returns rows
`X_df.columns`	returns columns
`X_df[['t1', 't4']]`	returns **data frame** of **selected columns**
`X_df.loc[['Data Set 12', 'Data Set 2']]`	returns **data frame** of **selected rows**
`X_df.iloc[4:6, 5:9]`	slicing **data frame** using `iloc`

- **Matplotlib** → simple standard plots
- **Seaborn** → sophisticated plots

inputs are `np.arrays` and `data frames!`

`set_theme()`, `load_dataset()`, `boxplot()`, `stripplot()`, `despine()`

```
sns.set_theme(style = "ticks")

f, ax = plt.subplots(figsize = (7, 6))
ax.set_xscale("log")

planets = sns.load_dataset("planets")

sns.boxplot(planets, x = "distance", y = "method", hue = "method",\
    whis = [0, 100], width = .6, palette = "vlag")

sns.stripplot(planets, x = "distance", y = "method", size = 4, color = ".3")


ax.xaxis.grid(True)
ax.set(ylabel = "")
sns.despine(trim = True, left = True)
```

Introduction to Computational Techniques in Physics:

- **Matplotlib** → simple standard plots
- **Seaborn** → sophisticated plots

short ("messy") table

Index	t0	t1	t2
Data Set 0	-8.00758	6.33521	-0.105416
Data Set 1	-10.1712	15.6568	4.80345
Data Set 2	0.88356	8.20249	2.46666
Data Set 3	-6.58955	15.8528	4.95933

long ("tidy") table

planets - DataFrame

Index	method	number	orbital_period	mass	distance
0	Radial Velocity	1	269.3	7.1	77.4
1	Radial Velocity	1	874.774	2.21	56.95
2	Radial Velocity	1	763	2.6	19.84
3	Radial Velocity	1	326.03	19.4	110.62

- **Matplotlib** → simple standard plots
- **Seaborn** → sophisticated plots

```
X_df['Data Set'] = X_df.index
```

adding indices as new column, which we name *'Data Set'*

```
X_df_tidy        = X_df.melt('Data Set', var_name = 'time point',\
                                         value_name = 'time')
```

melt turns data frame from „messy" to „tidy".

X_df_tidy - DataFrame

Index	Data Set	time point	time
0	Data Set 0	t0	-8.00758
1	Data Set 1	t0	-10.1712
2	Data Set 2	t0	0.88356
3	Data Set 3	t0	-6.58955

planets - DataFrame

Index	method	number	orbital_period	mass	distance
0	Radial Velocity	1	269.3	7.1	77.4
1	Radial Velocity	1	874.774	2.21	56.95
2	Radial Velocity	1	763	2.6	19.84
3	Radial Velocity	1	326.03	19.4	110.62

- **Matplotlib** → simple standard plots
- **Seaborn** → sophisticated plots

```
sns.set_theme(style = "ticks")

f, ax = plt.subplots(figsize = (7, 6))
ax.set_xscale("log")

planets = sns.load_dataset("planets")

sns.boxplot(planets, x = "distance", y = "method", hue = "method",\
            whis = [0, 100], width = .6, palette = "vlag")

sns.stripplot(planets, x = "distance", y = "method", size = 4, color = ".3")


ax.xaxis.grid(True)
ax.set(ylabel = "")
sns.despine(trim = True, left = True)
```

- **Matplotlib** → simple standard plots
- **Seaborn** → sophisticated plots

```
sns.set_theme(style = "ticks")

f, ax = plt.subplots(figsize = (7, 6))



sns.boxplot(planets, x = "distance", y = "method", hue = "method",\
                 whis = [0, 100], width = .6, palette = "vlag")

sns.stripplot(planets, x = "distance", y = "method", size = 4, color = ".3")


ax.xaxis.grid(True)
ax.set(ylabel = "")
sns.despine(trim = True, left = True)
```

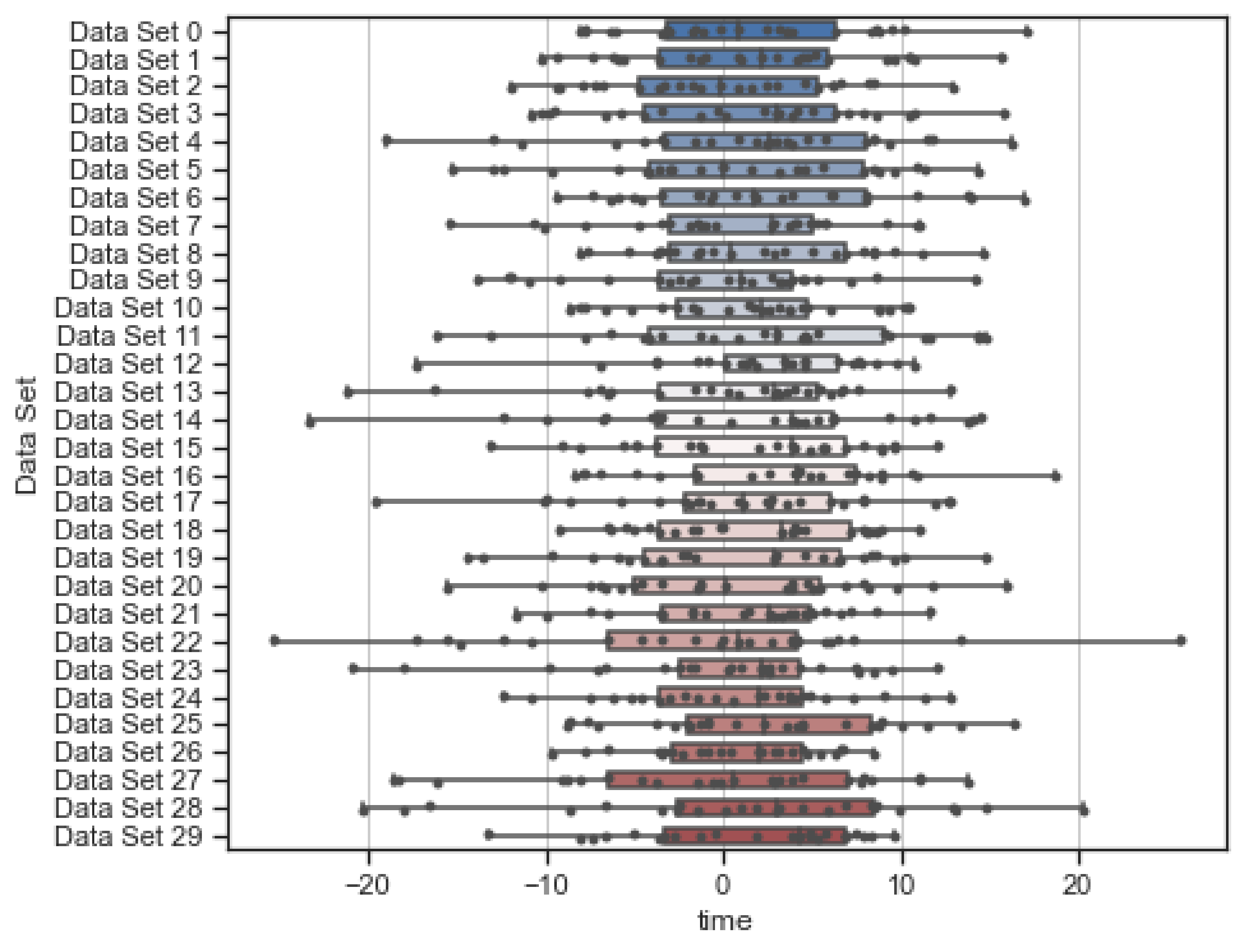
- **Matplotlib** → simple standard plots
- **Seaborn** → sophisticated plots

```
sns.set_theme(style = "ticks")

f, ax = plt.subplots(figsize = (7, 6))

sns.boxplot(planets, x = "distance", y = "method", hue = "method",\
                whis = [0, 100], width = .6, palette = "vlag")

sns.stripplot(planets, x = "distance", y = "method", size = 4, color = ".3")


ax.xaxis.grid(True)
ax.set(ylabel = "")
sns.despine(trim = True, left = True)
```

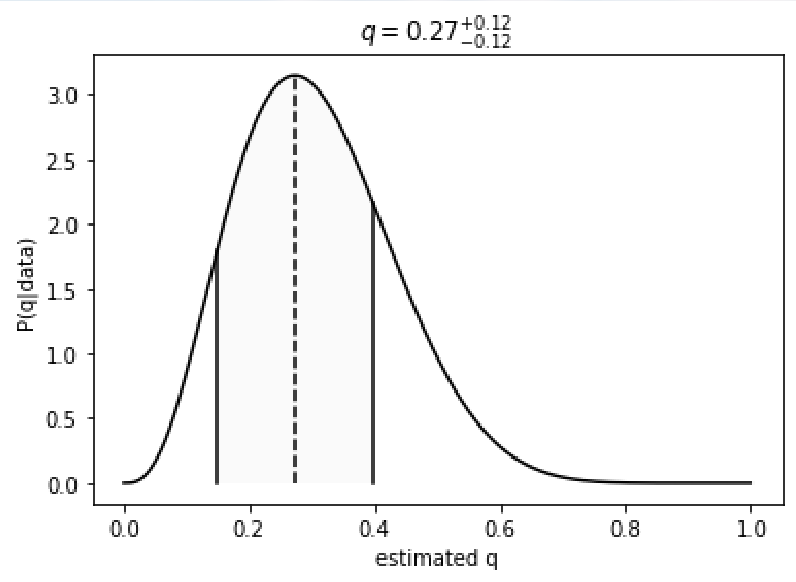
- **Matplotlib** → simple standard plots
- **Seaborn** → sophisticated plots

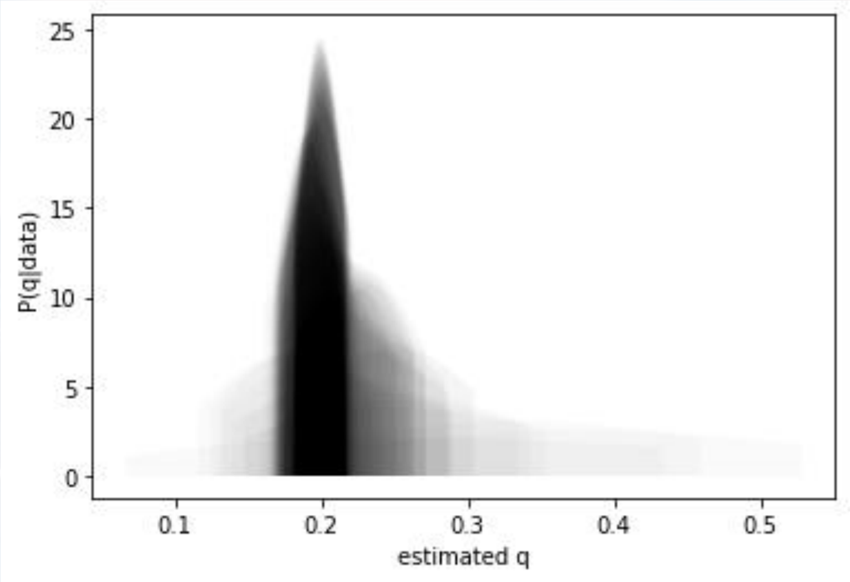
```
sns.set_theme(style = "ticks")

f, ax = plt.subplots(figsize = (7, 6))

sns.boxplot( X_df_tidy,    x = "time" , y ="Data Set",\
               whis = [0, 100], width = .6, palette = "vlag")

sns.stripplot(X_df_tidy, x = "time" , y ="Data Set", size = 4, color = ".3")


ax.xaxis.grid(True)
ax.set(ylabel = "")
sns.despine(trim = True, left = True)
```

- **Matplotlib** → simple standard plots
- **Seaborn** → sophisticated plots



other useful commands/tools



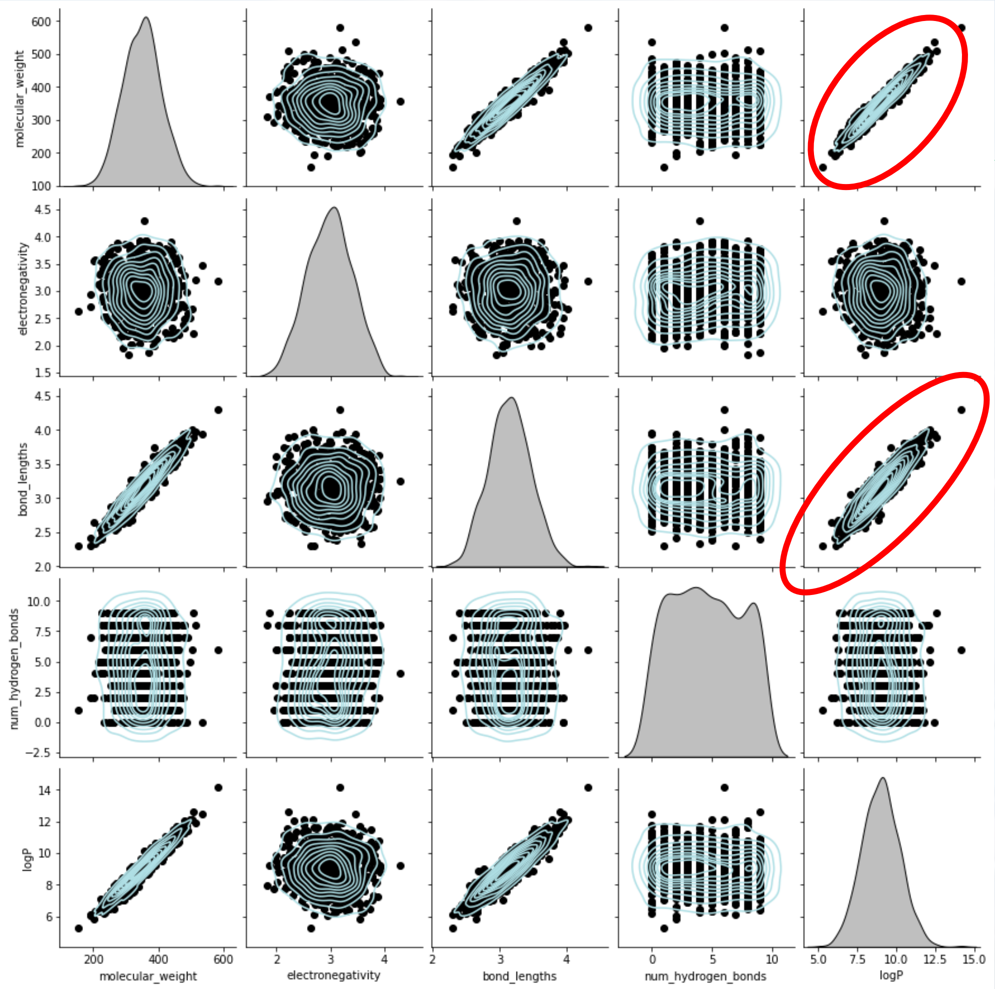



```
plt.fill(xtofill, ytofill, facecolor = 'black', alpha = 0.02)
```

Introduction to Computational Techniques in Physics:


```
sns.pairplot(Data, kind = "kde")
out.map_offdiag(plt.scatter, color = 'black')
plt.show()
```

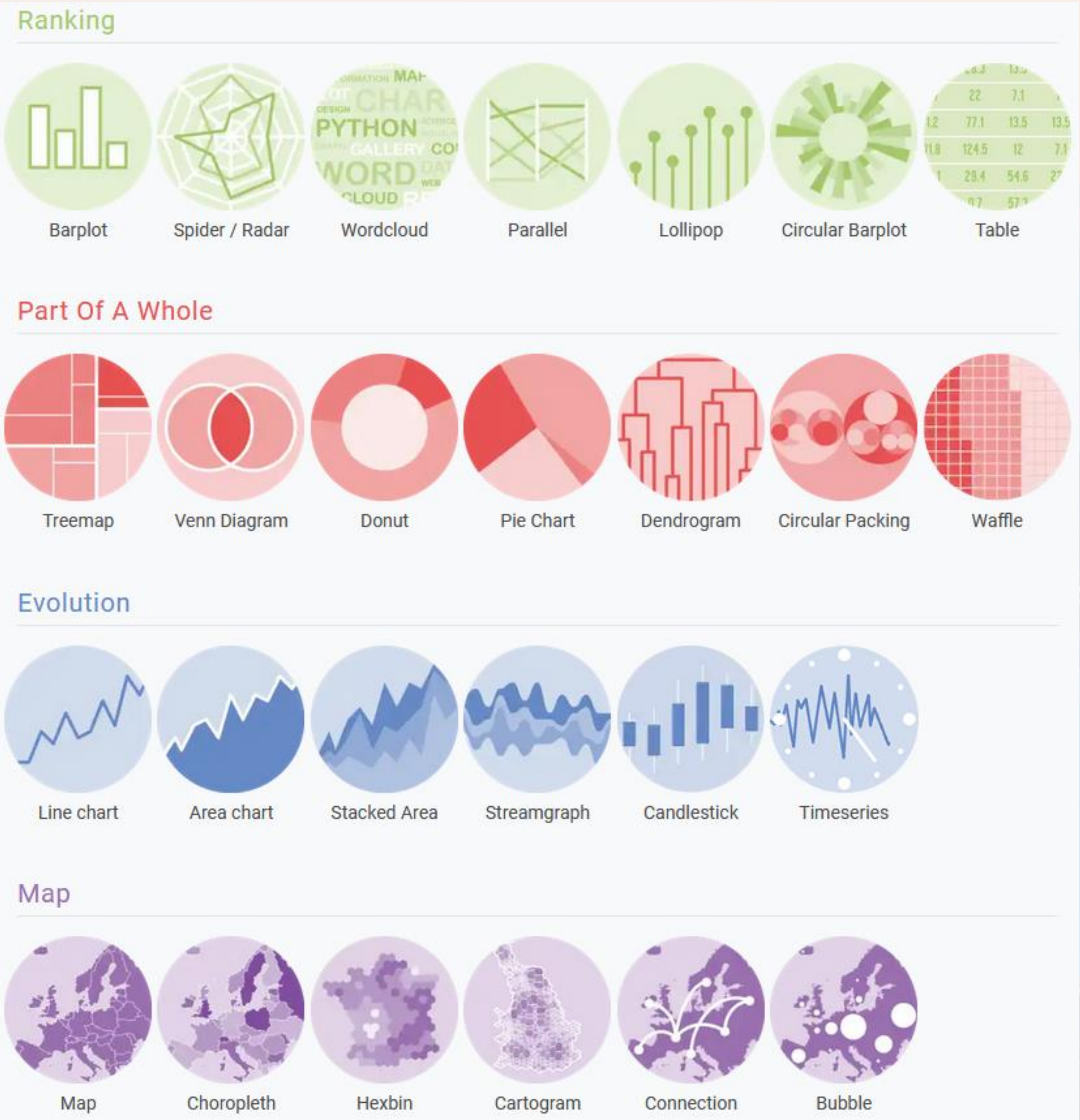
label	molecular_weight	electronegativity	bond_lengths	num_hydrogen_bonds	logP
Toxic	382.602	2.00269	3.61153	3	9.82666
Toxic	408.961	2.93626	3.47904	6	9.85889
Non-Toxic	239.548	2.71413	2.63922	8	6.75962
Non-Toxic	315.58	2.85598	2.86034	9	8.70674
Non-Toxic	282.521	2.83877	2.9664	1	7.8173



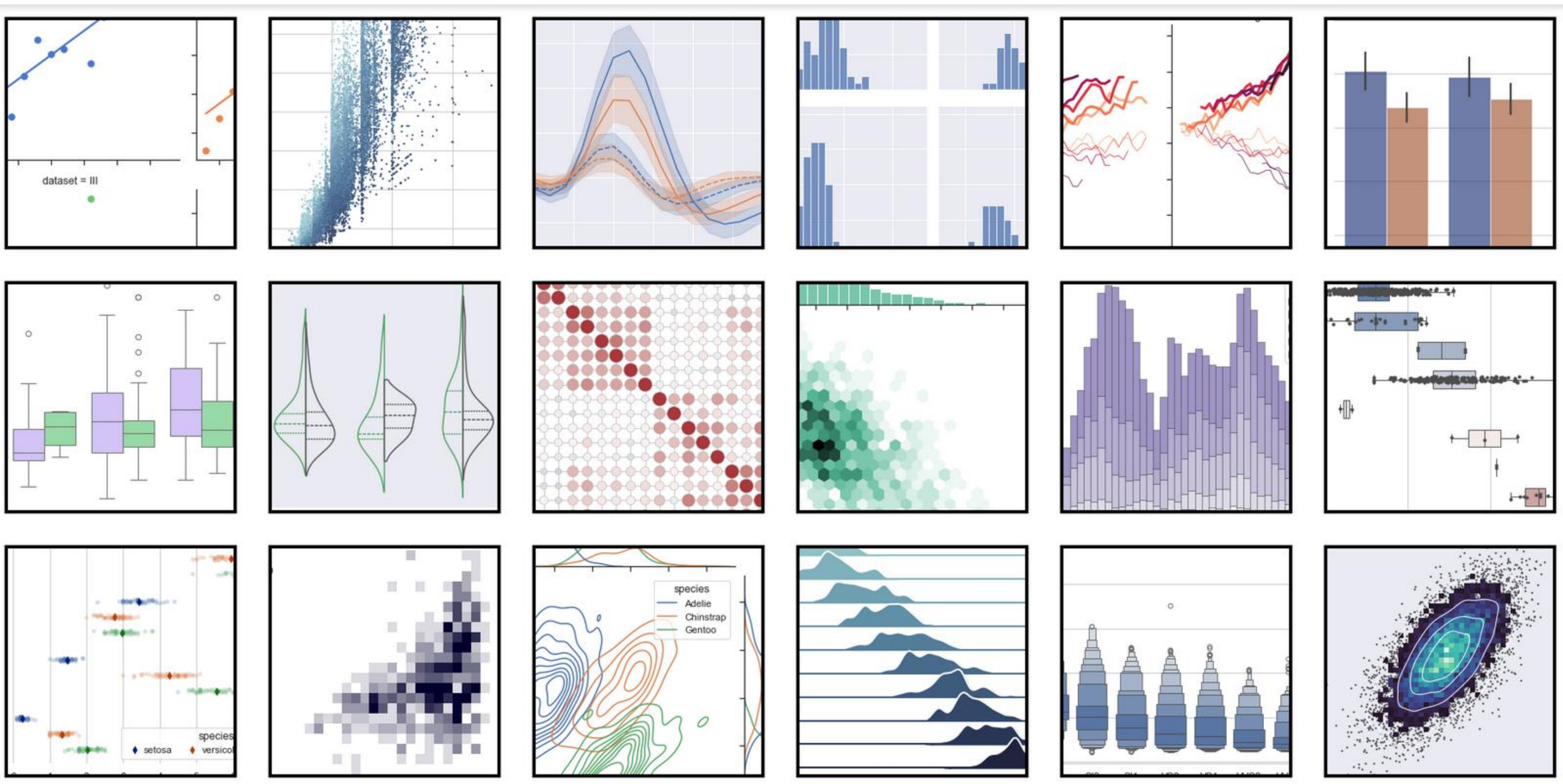

```
sns.heatmap(corr, annot = True)
```

$$corr(x,y) = \frac{cov(x,y)}{\sqrt{var(x)var(y)}}$$

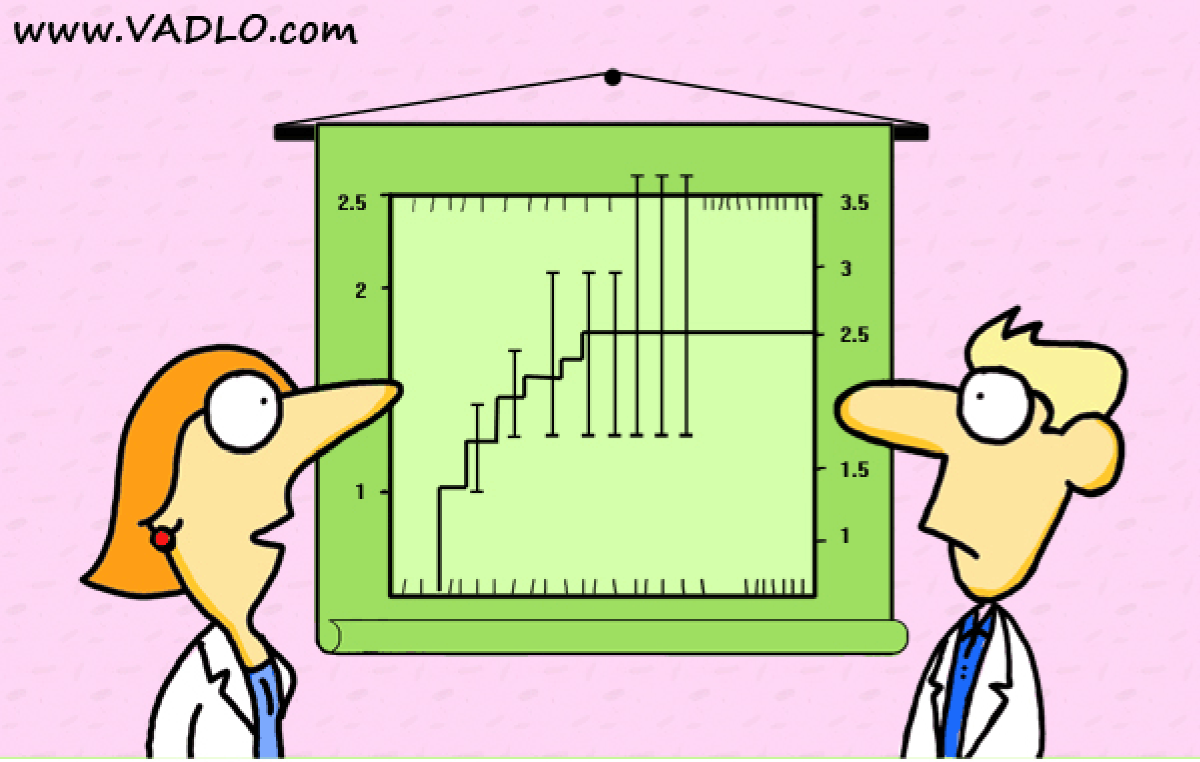
https://python-graph-gallery.com/



https://seaborn.pydata.org/examples/index.html



Thank you for your attention!



“Did you really have to show the error bars?”