# Data visualization with Matplotlib and Seaborn

Inteligencia Artificial en los Sistemas de Control Autónomo
Máster en Ciencia y Tecnología desde el Espacio

Departamento de Automática

Universidad de Alcalá

ISG

## Objectives

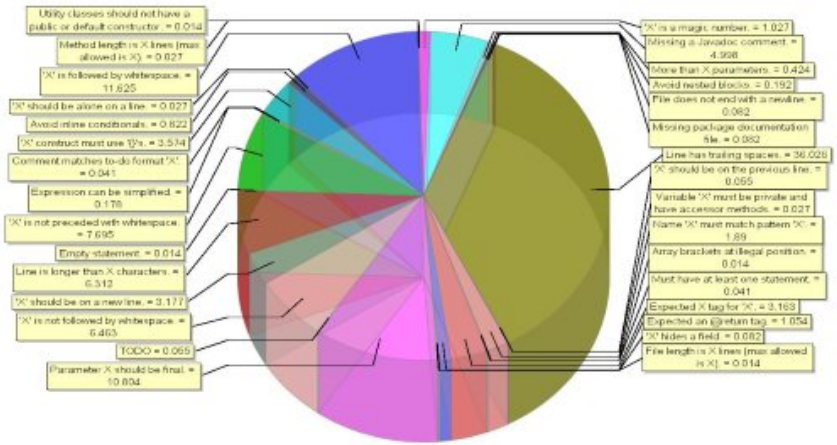
1. Motivate the importance of data visualization
2. Avoid some common mistakes in data visualization
3. Choose the proper visualization technique
4. Overview Matplotlib
5. Introduce Seaborn

## Bibliography

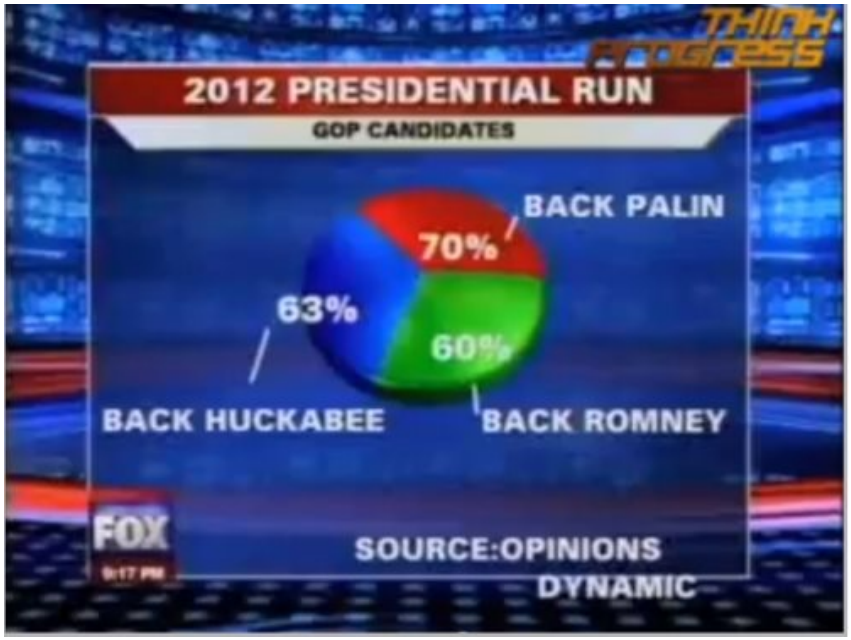
Jake VanderPlas. *Python Data Science Handbook*. Chapters 4. O'Reilly. (Link).

# Table of Contents

# Visualization examples (I)

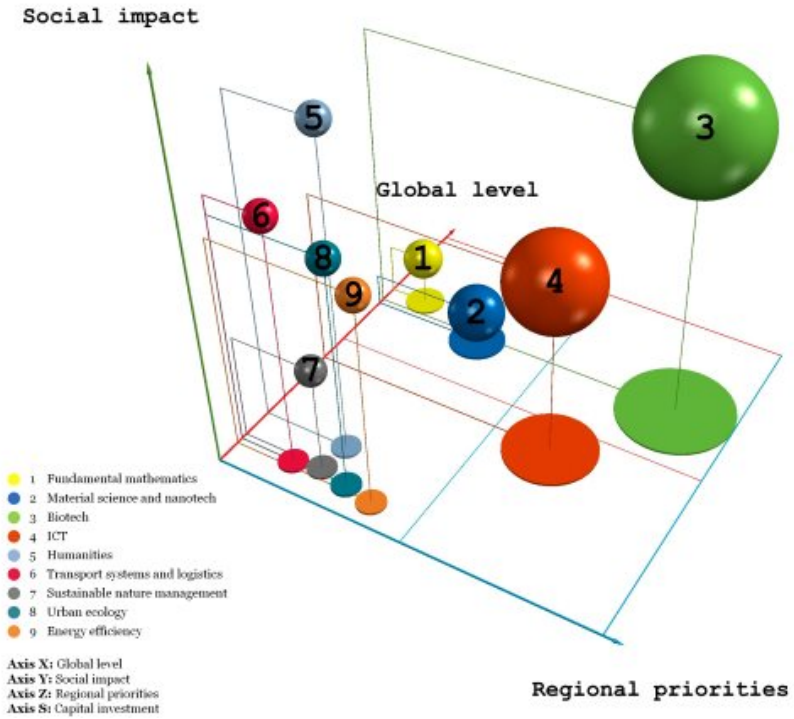



(Source)

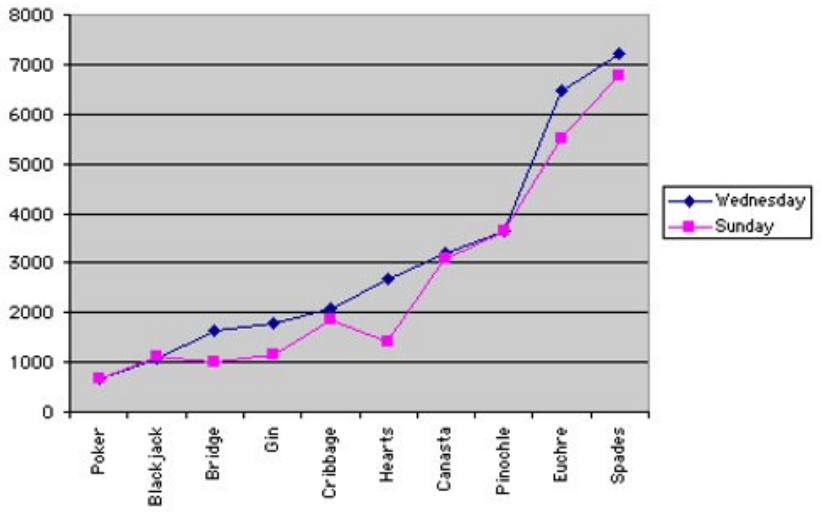



(Source)

# Visualization examples (II)

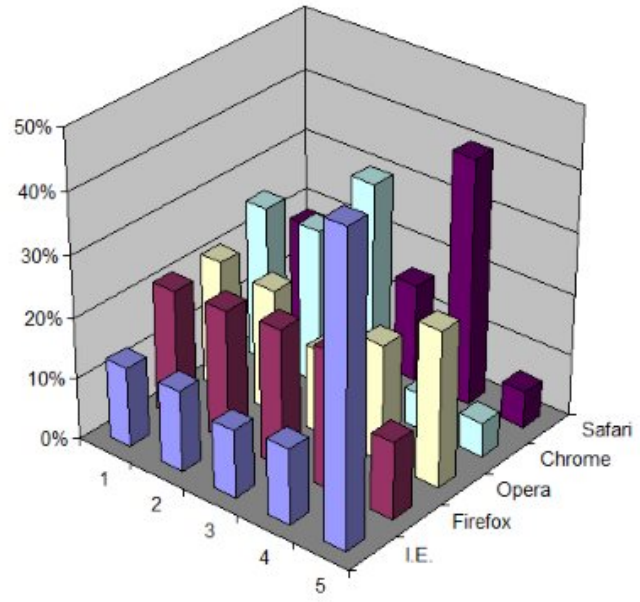



(Source)

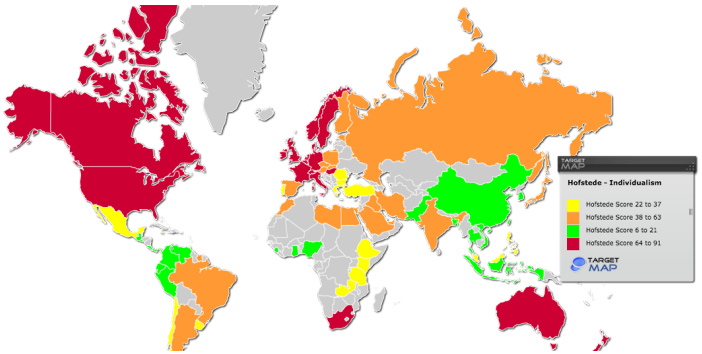



(Source)

# Visualization examples (III)

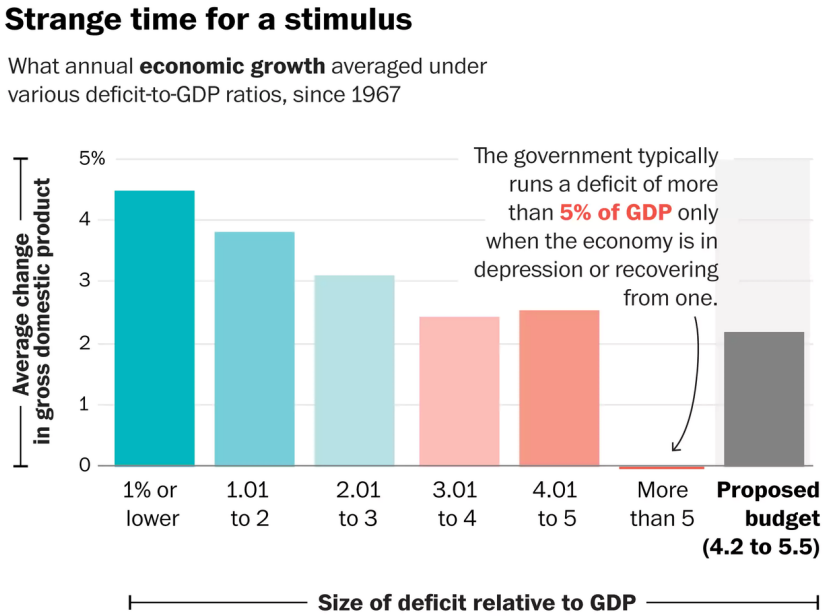



(Source)

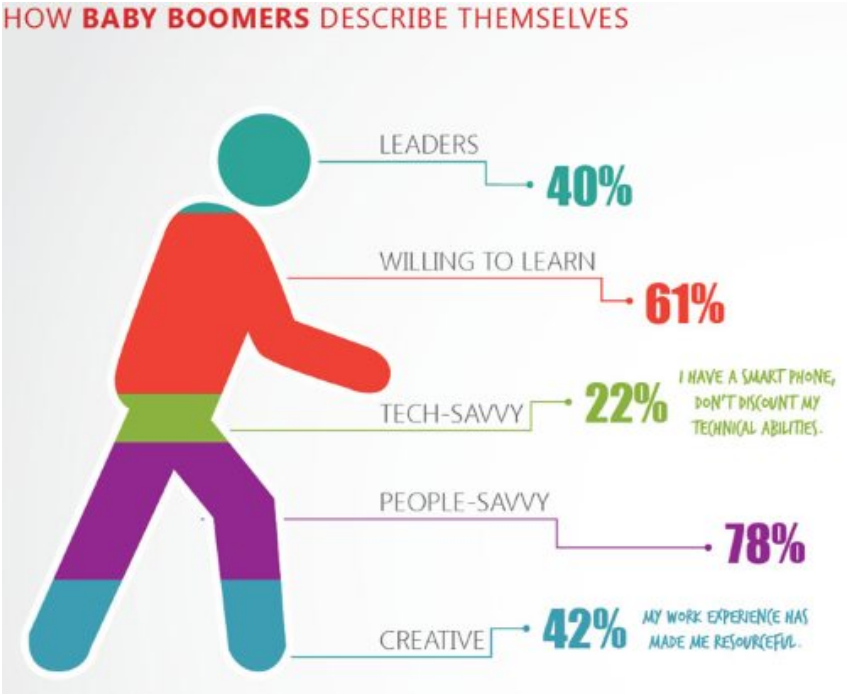



(Source)

# Visualization examples (IV)

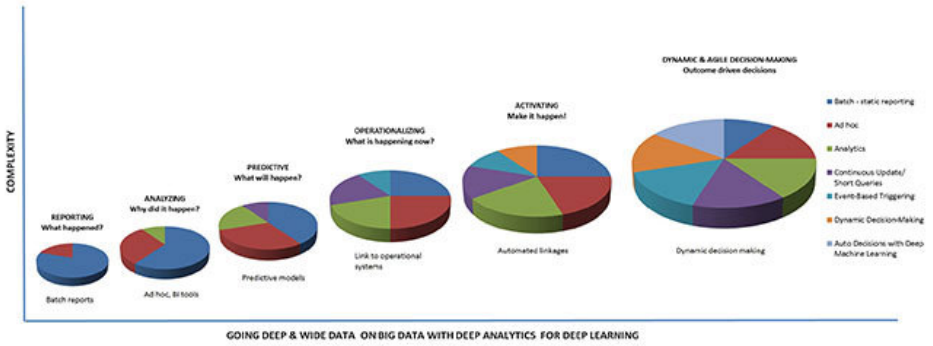



(Source)

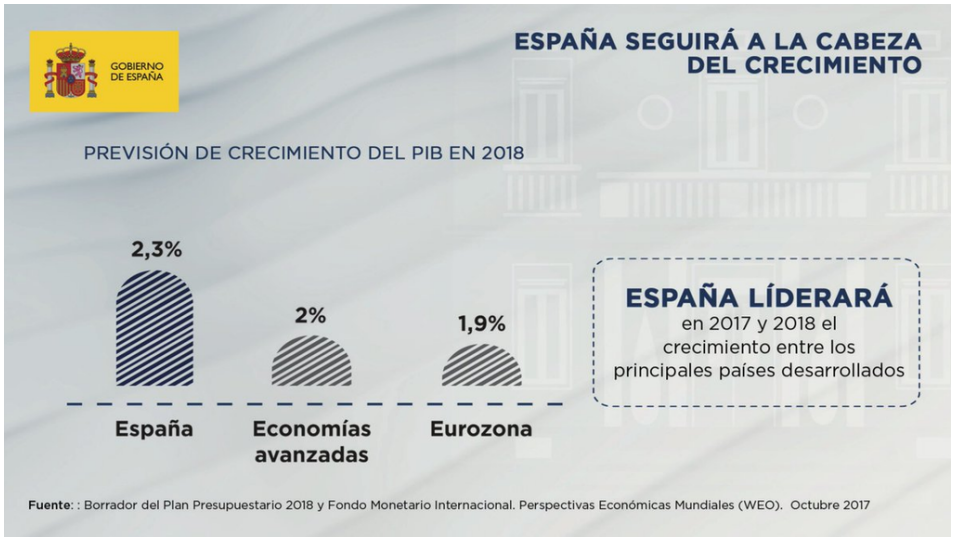



(Source)

# Visualization examples (V)




(Source)




(Source)

# Motivation (I)

Efficient data visualization tips

- **Define your story**
- The chart must tell the story
- Don't distract from your story (with irrelevant data or visual elements)
- One story, one chart
- Put the story comprension in first term
- Better several simple charts than one complex chart
- Choose colors wisely (color scale or high contrast)
- Elements order must support the story (leyend, bars, etc)
- There is life beyond pies and bars
- Keep it simple, stupid!

# Motivation (II)

Know your data

- Categorical or numerical
- Number of dimensions to represent (1D, 2D, 3D, more dimensions)

Can you use other representation?

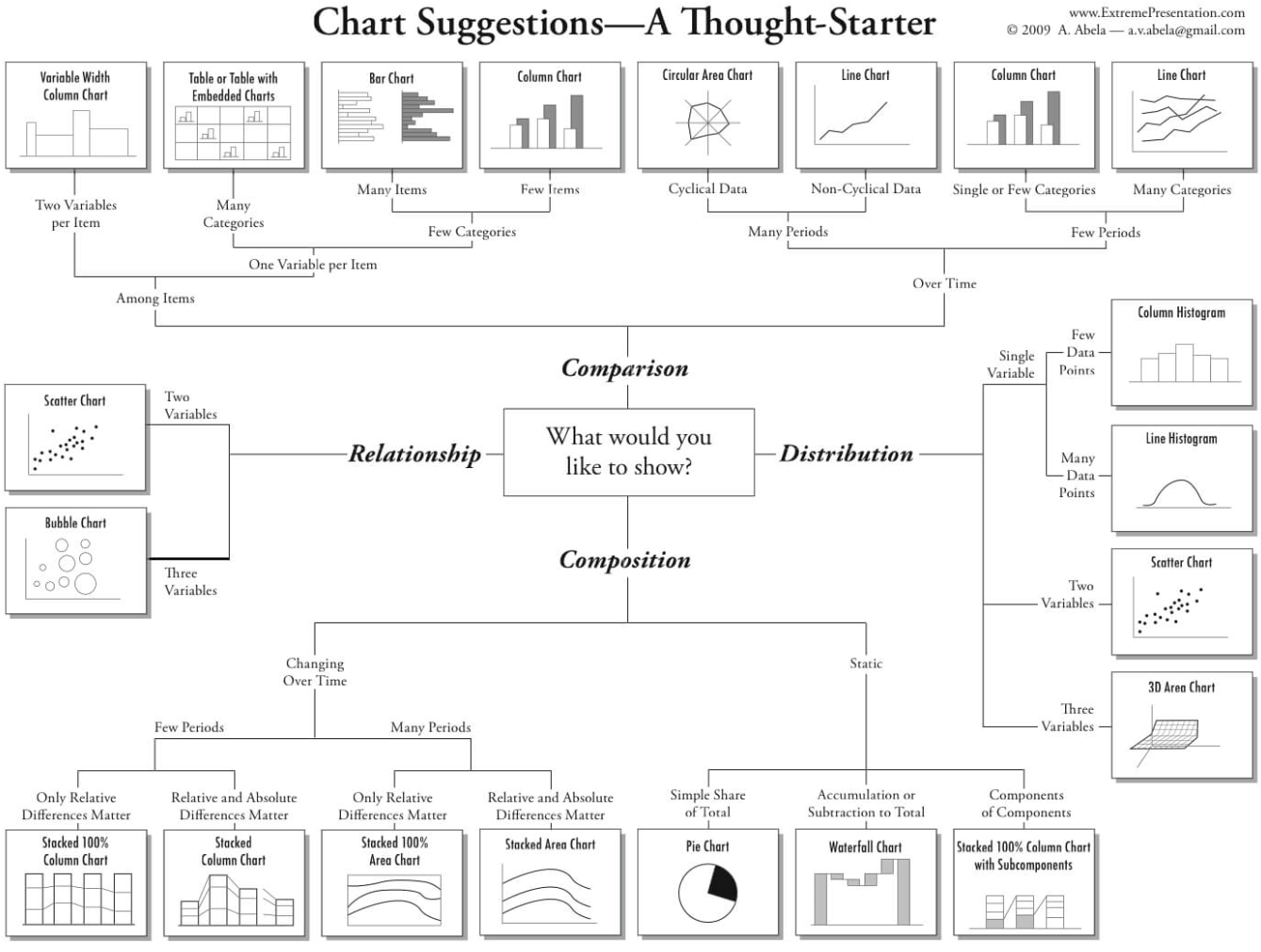
- Chart better than table? ...
- ... that depends

What do you want to represent?

- Distribution, relationship, comparison or composition

Look for templates: (`https://python-graph-gallery.com/`)

# Motivation (III)




(Source)

(Alternative resource)

# Matplotlib (I)

Matplotlib is a Python package

- Based on NumPy
- Imitates Matlab

Three operation modes

- Scripts.
  Must use `plt.show()` to enter event loop. Use it once!
- IPython shell.
  Must use `%matplotlib`
- IPython notebook. Two modes
  - `%matplotlib inline`
  - `%matplotlib notebook`

**Convention**

```
import matplotlib as mpl
import matplotlib.pyplot as plt
```

myplot.py

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

x = np.linspace(0, 10, 100)

plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))

plt.show()
```

# Matplotlib (II)

Matplotlib comes with two interfaces

- Matlab-like. Old-fashioned function-oriented API.
- Object-oriented. Object-oriented and more powerfull API.

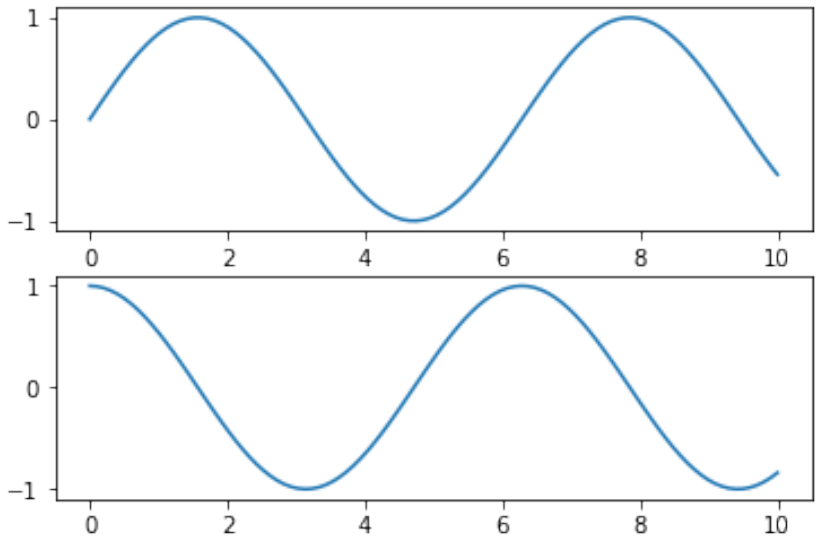
## Matlab API

```
plt.figure()  # create a plot

# create the first of two panels
    and set current axis
plt.subplot(2, 1, 1) # (rows,
    columns, panel number)
plt.plot(x, np.sin(x))

# create the second panel and
    set current axis
plt.subplot(2, 1, 2)
plt.plot(x, np.cos(x));
```

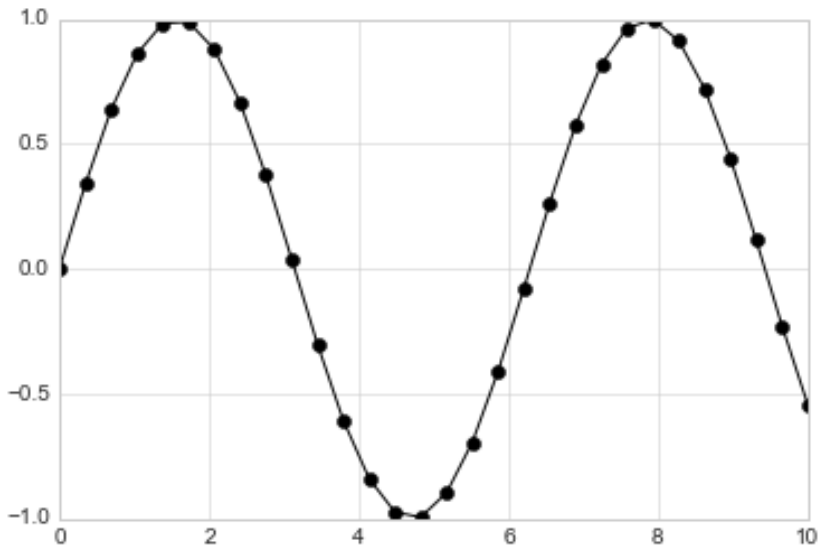
## OO API

```
# First create a grid of plots
# ax will be an array of two
    Axes objects
fig, ax = plt.subplots(2)

# Call plot() method on the
    appropriate object
ax[0].plot(x, np.sin(x))
ax[1].plot(x, np.cos(x));
```

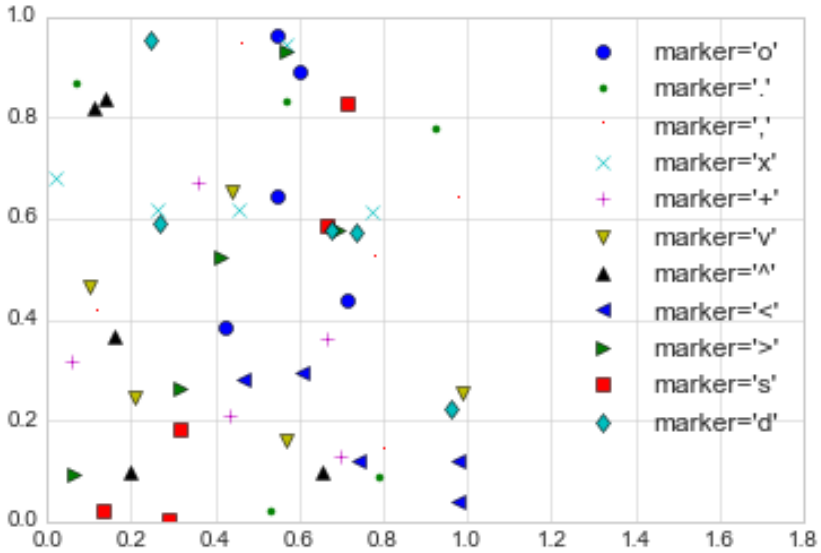
# Matplotlib (III)



# Matplotlib (IV)

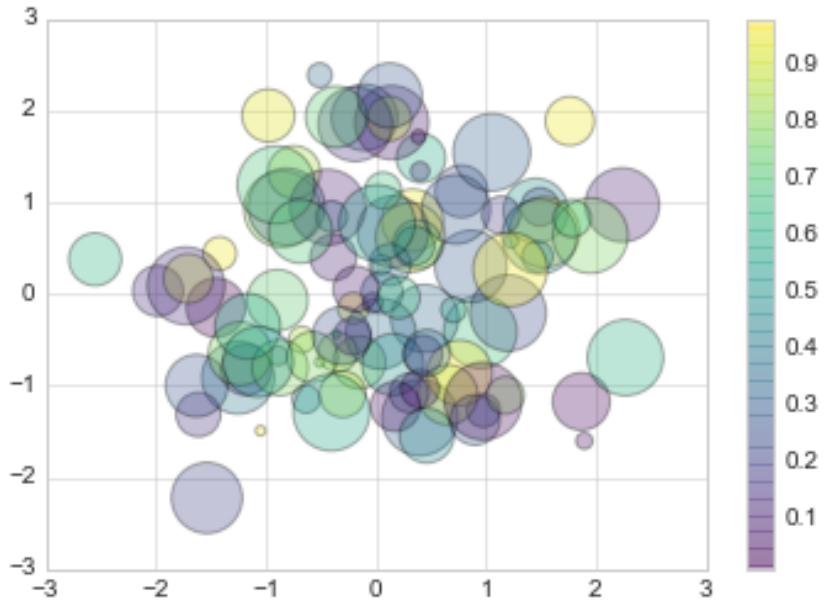



```
plt.plot(x, np.sin(x), '-ok',
         color='black')
```

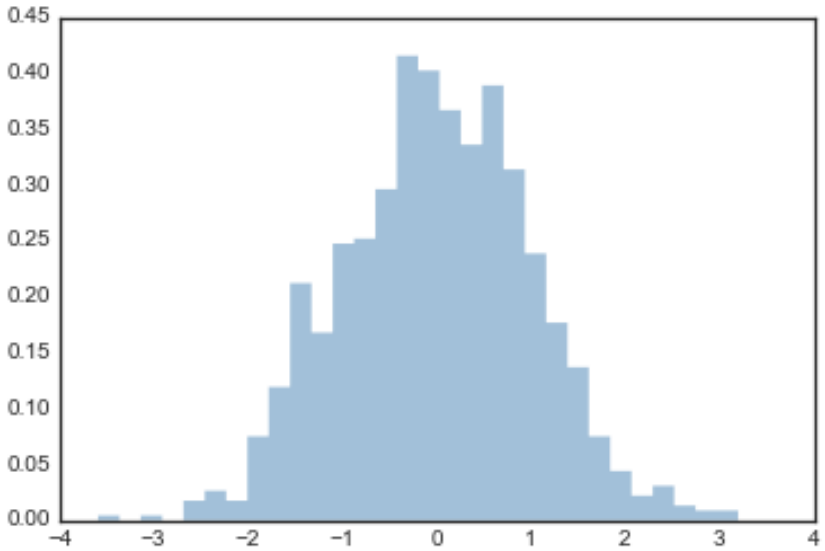



```
for marker in ['o', '.', ',', 'x', '+', 'v', '^', '
    <', '>', 's', 'd']:
    plt.plot(rng.rand(5), rng.rand(5), marker,
        label="marker='{0}'".format(marker))
plt.legend(numpoints=1)
plt.xlim(0, 1.8);
```

# Matplotlib (V)



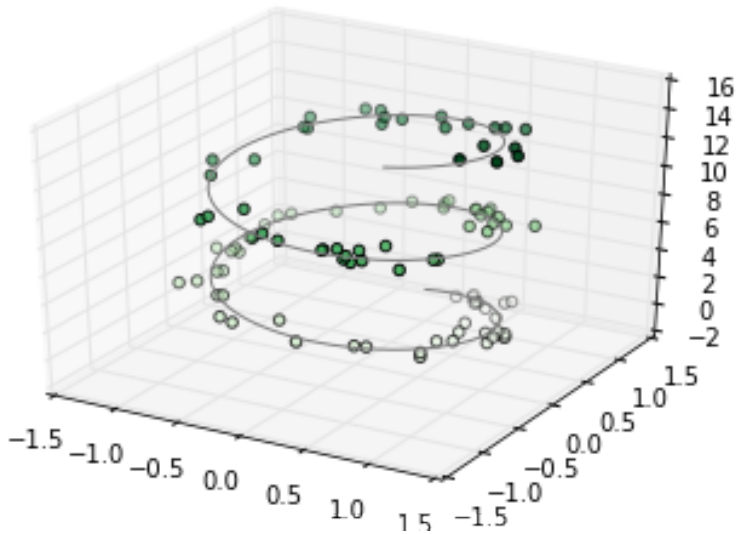

```
rng = np.random.RandomState(0)
x = rng.randn(100)
y = rng.randn(100)
colors = rng.rand(100)
sizes = 1000 * rng.rand(100)

plt.scatter(x, y, c=colors, s=sizes, alpha=0.3,
    cmap='viridis')
plt.colorbar();  # show color scale
```




```
data = np.random.randn(1000)

plt.hist(data, bins=30, normed=True, alpha=0.5,
     histtype='stepfilled', color='steelblue',
     edgecolor='none');
```

# Matplotlib (VI)

```
ax = plt.axes(projection='3d')

# Data for a three-dimensional line
zline = np.linspace(0, 15, 1000)
xline = np.sin(zline)
yline = np.cos(zline)
ax.plot3D(xline, yline, zline, 'gray')

# Data for three-dimensional scattered points
zdata = 15 * np.random.random(100)
xdata = np.sin(zdata) + 0.1 * np.random.randn(100)
ydata = np.cos(zdata) + 0.1 * np.random.randn(100)
ax.scatter3D(xdata, ydata, zdata, c=zdata, cmap='
      Greens');
```



# Matplotlib

## Matplotlib notebook

### Matplotlib notebook

(Link to notebook)

# Seaborn (I)

Seaborn is a modern data-visualization Python package

- Based on matplotlib
- ... it uses matplotlib indeed
- Pandas-aware
- High level
- Advanced visualizations
- Easy to use

Still under development!

**Convention**

```
import seaborn as sns
```

This documentation is for Seaborn 0.9 or newer

# Seaborn (II)

Display initialization

- `plt.show()`
- `%matplotlib`

Style initialization

- Default Seaborn style `sns.set()`
- By default, same style than matplotlib

Several functions ...

- ... similar parameters

## Parameters

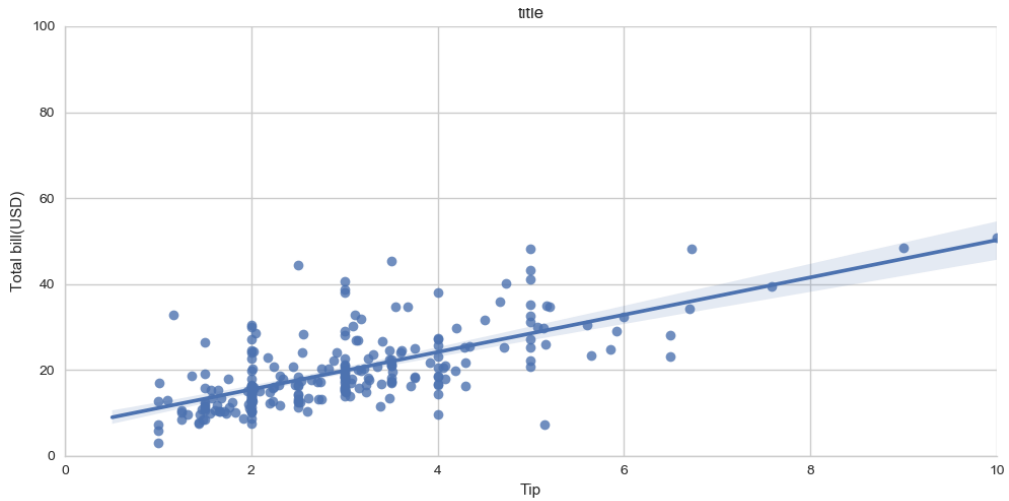
- x: Data axis x
- y: Data axis Y
- data: Dataframe name
- hue: Color
- style: Style
- sizes: Size
- kind: Alternate representation

# Seaborn (III)

## Typical Seaborn usage

1. Prepare data
2. Set up aesthetics
3. Plot
4. Customize the plot

```
import matplotlib.pyplot as plt
import seaborn as sns
# Prepare data
tips = sns.load_dataset("tips")
# Set up aesthetics
sns.set_style("whitegrid")
# Plot
g = sns.lmplot(x="tip",y="total_bill", data=tips,aspect=2)
# Plot customization
g = (g.set_axis_labels("Tip","Total bill(USD)").set(xlim
    =(0,10),ylim=(0,100)))
plt.title("title")
plt.show(g)
```



# Seaborn

## Datasets (I)

Seaborn comes with several dummy datasets

- `sns.load_dataset('name')`

We will use two datasets

- 'iris': The classical iris dataset, numerical
- 'tips': Numeric and categorical variables

**Tips dataset**

```
>>> tips = sns.load_dataset('tips')
>>> print(tips.head())
```

| | total_bill | tip | sex | smoker | day | time | size |
|---|---|---|---|---|---|---|---|
| 0 | 16.99 | 1.01 | Female | No | Sun | Dinner | 2 |
| 1 | 10.34 | 1.66 | Male | No | Sun | Dinner | 3 |
| 2 | 21.01 | 3.50 | Male | No | Sun | Dinner | 3 |
| 3 | 23.68 | 3.31 | Male | No | Sun | Dinner | 2 |
| 4 | 24.59 | 3.61 | Female | No | Sun | Dinner | 4 |

# Seaborn

## Datasets (II)

**Iris dataset**

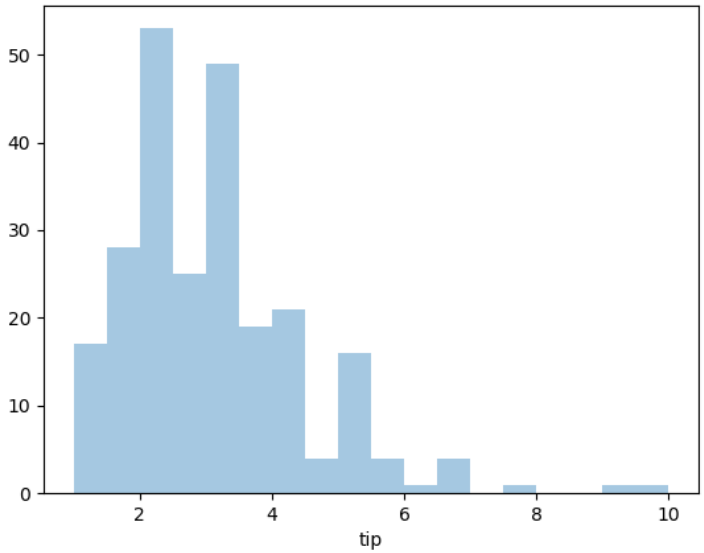
```
»> iris = sns.load_dataset('iris')
»> print(iris.head())
```

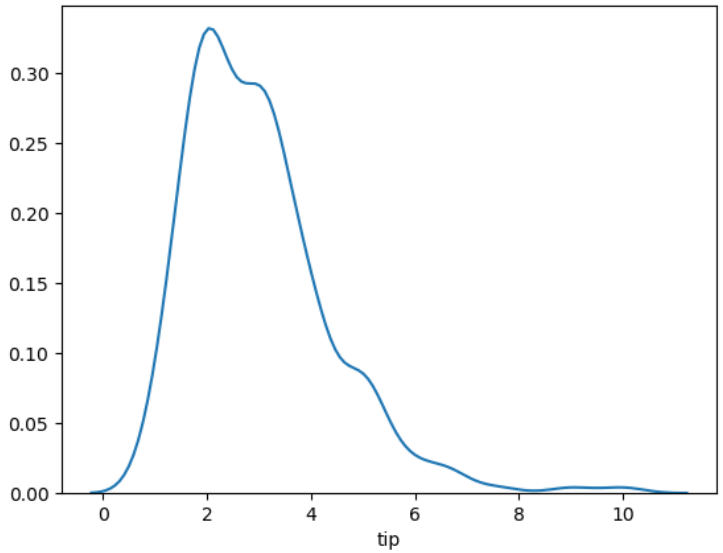
| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | setosa |


(Source)

# Seaborn

## Distributions (I)




### Histogram

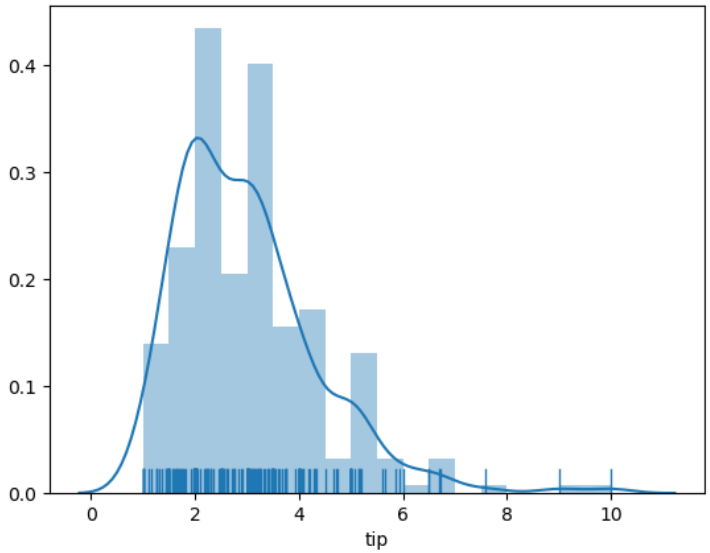
```
sns.distplot(tips['tip'],
    kde=False)
```

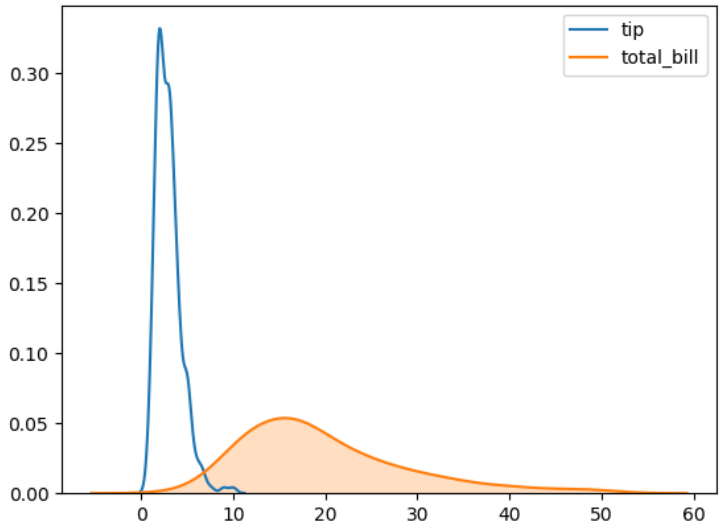
### Density plot

```
sns.distplot(tips['tip'],
    hist=False)
```

# Seaborn

## Distributions (II)




### Histogram + density plot

```
sns.distplot(tips['tip'],
    rug=True)
```

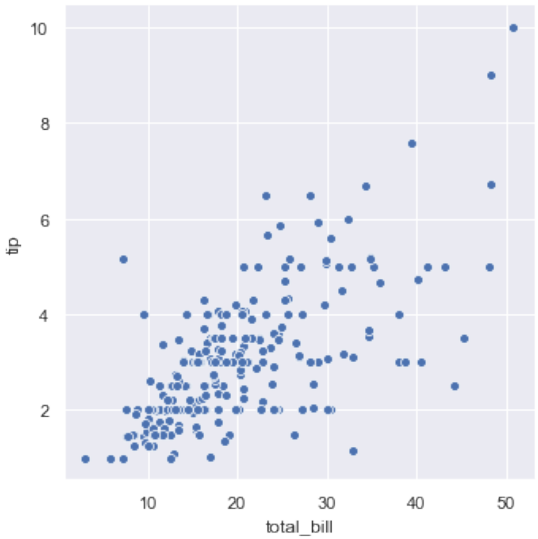
### Density plot

```
sns.kdeplot(tips['tip'])
sns.kdeplot(tips['total_bill
    '], shade=True)
```

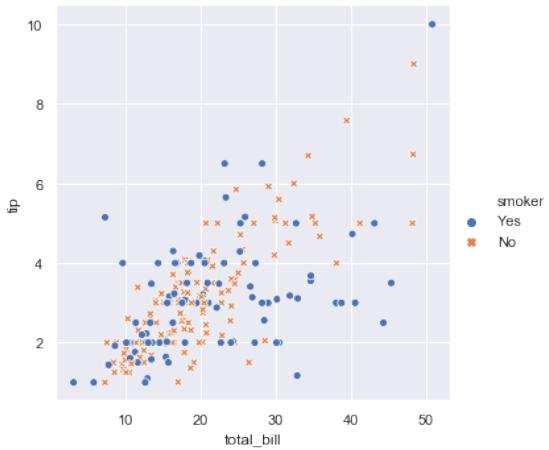
# Seaborn

## Relationships (I)

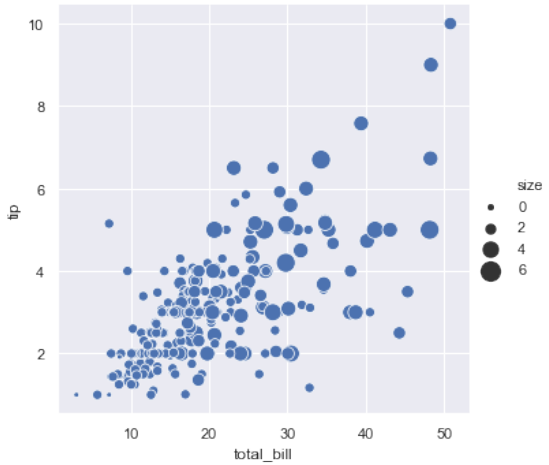
### Scatterplots




```
sns.relplot(x="total_bill", y="
    tip", data=tips)
```



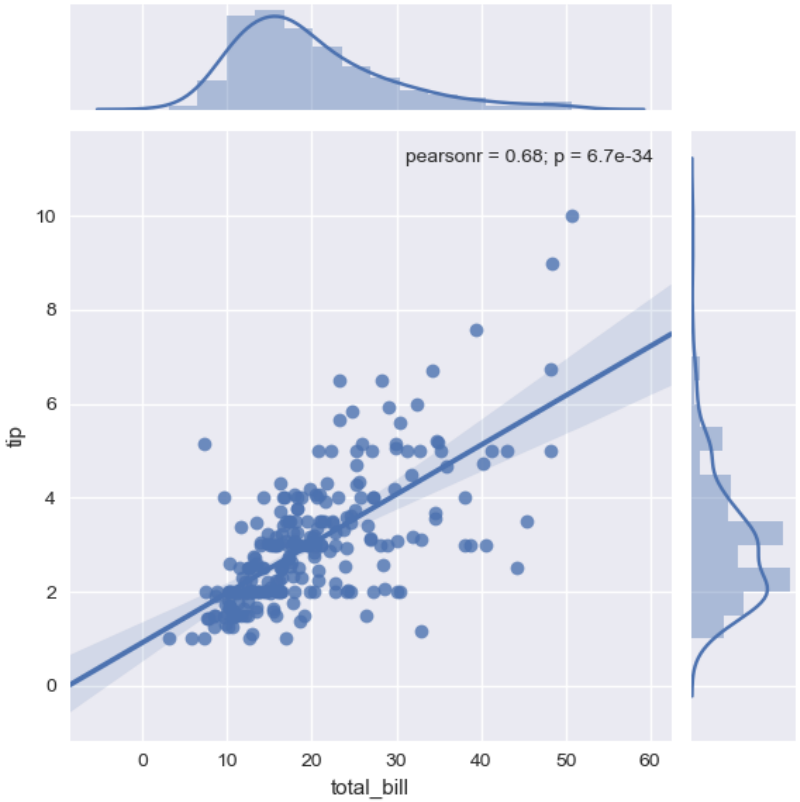

```
sns.relplot(x="total_bill", y="
    tip", hue="smoker", style="
    smoker", data=tips)
```



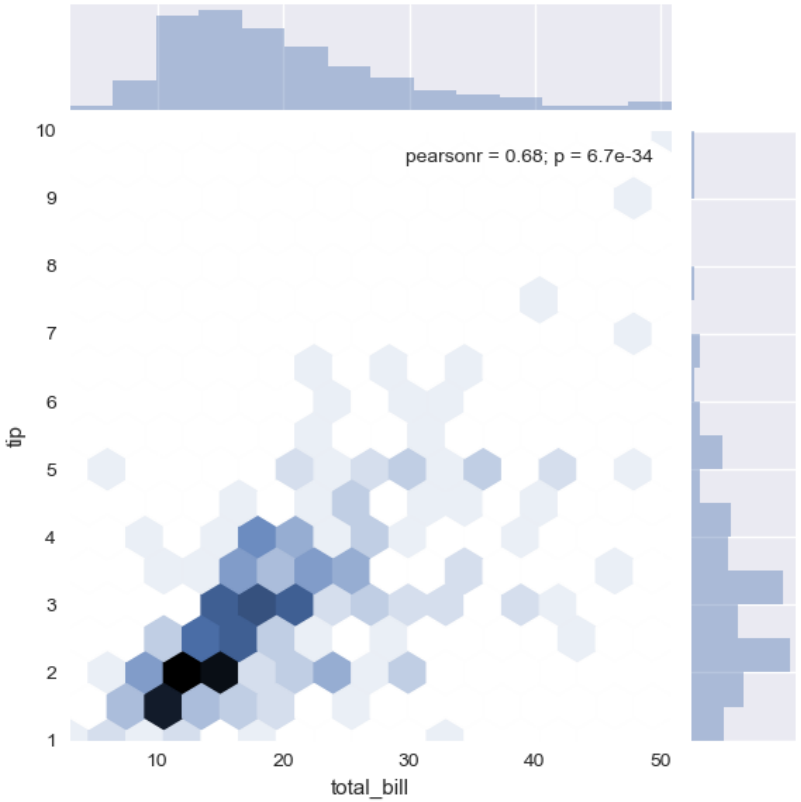

```
sns.relplot(x="total_bill", y="
    tip", size="size", sizes
    =(15, 200), data=tips);
```

# Seaborn

## Relationships (II)



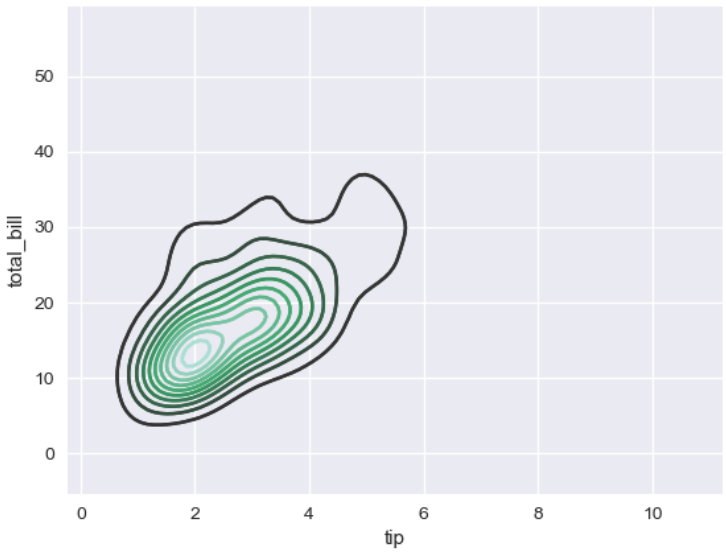

```
sns.jointplot("total_bill", "tip", tips, kind="reg")
```



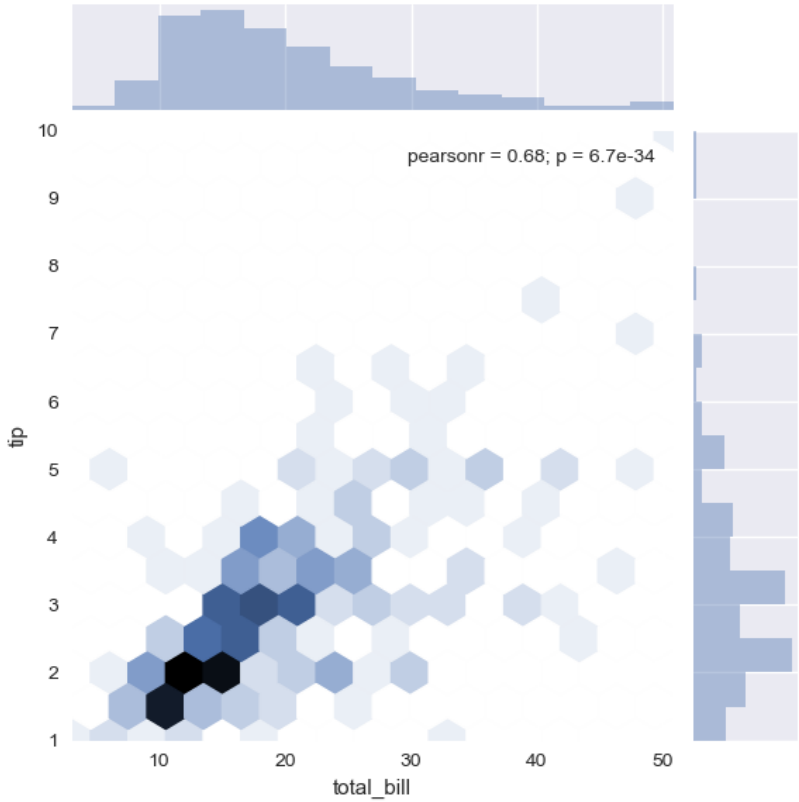

```
sns.jointplot("total_bill", "tip", tips, kind="hex")
```

# Seaborn

## Relationships (III)



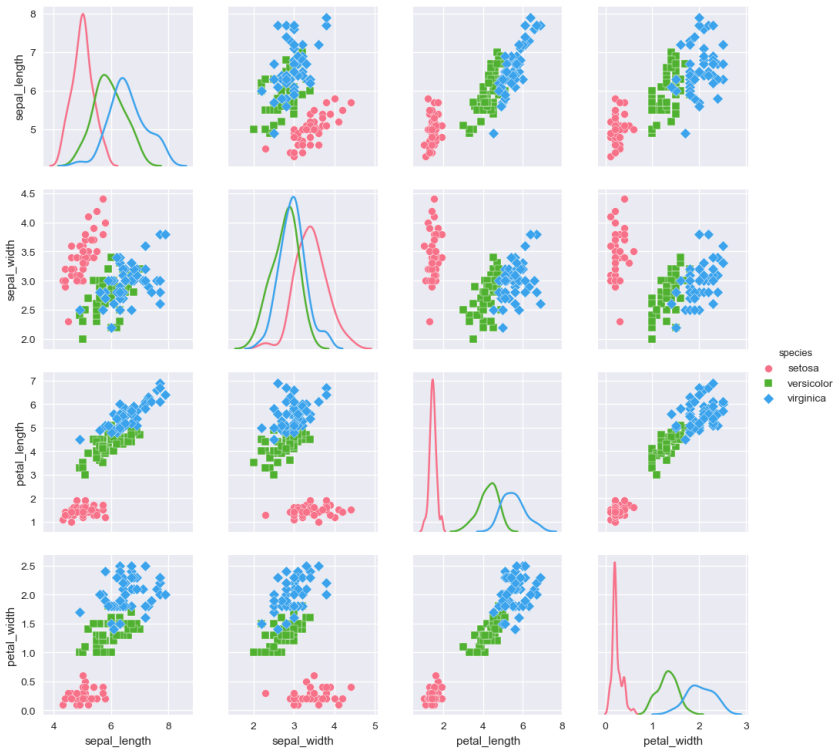

```
sns.kdeplot(tips['tip'], tips['total_bill'])
```




```
sns.jointplot("total_bill", "tip", tips , kind="hex")
```

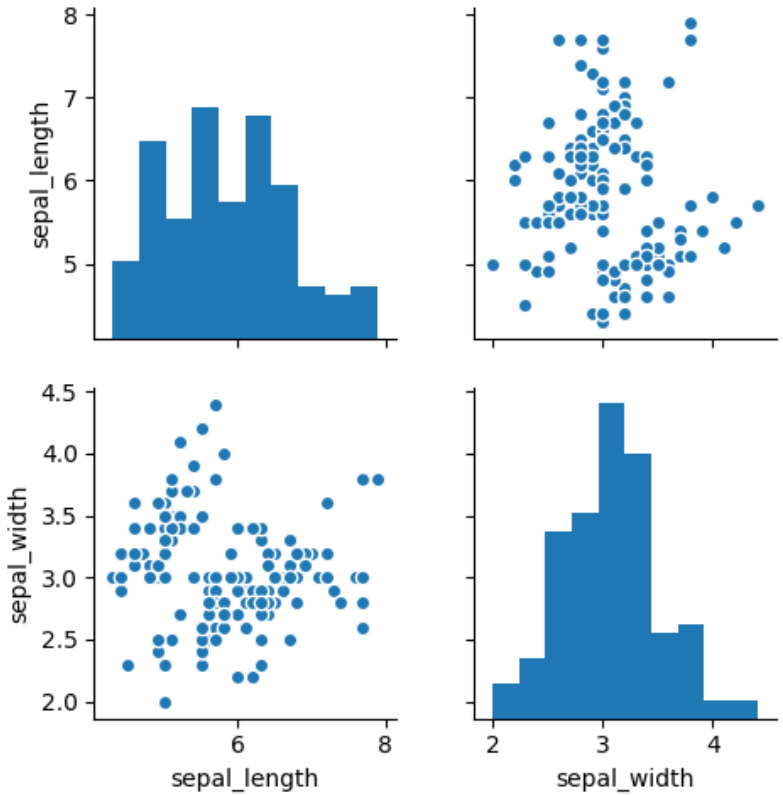
# Seaborn

## Relationships (IV)




### Scatterplot matrix

```
sns.pairplot(iris, hue="species", palette="husl",
             markers=["o", "s", "D"], diag_kind='kde')
```

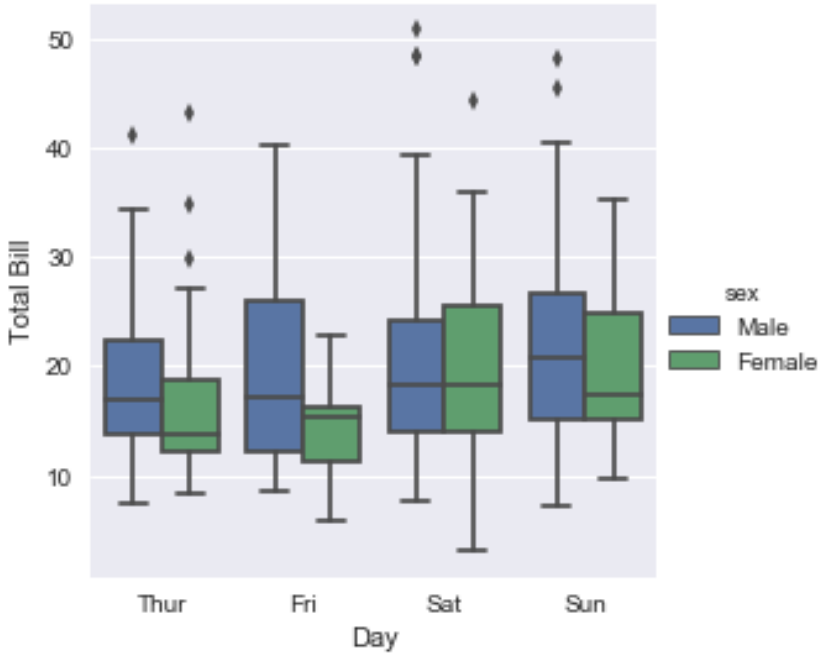



### Scatterplot matrix

```
sns.pairplot(iris, vars=["sepal_length", "
             sepal_width"])
```

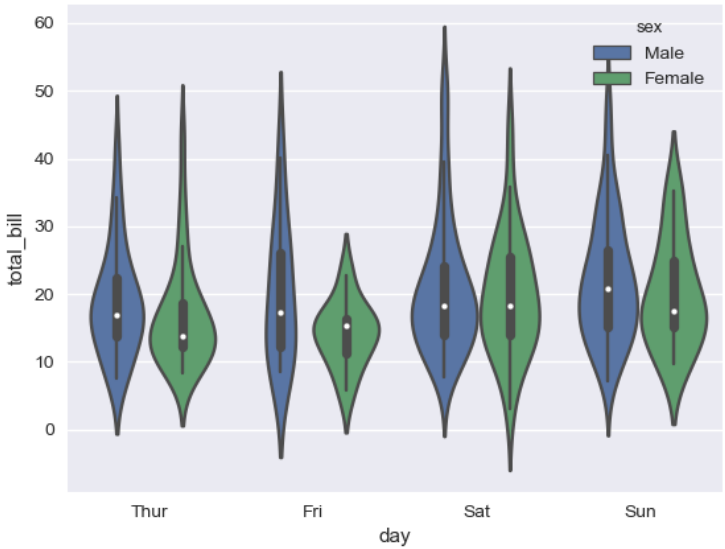
# Seaborn

## Comparisons (I)




### Boxplot

```
with sns.axes_style(style='ticks'):
    g = sns.factorplot("day", "total_bill", "sex",
            data=tips, kind="box")
g.set_axis_labels("Day", "Total Bill")
```

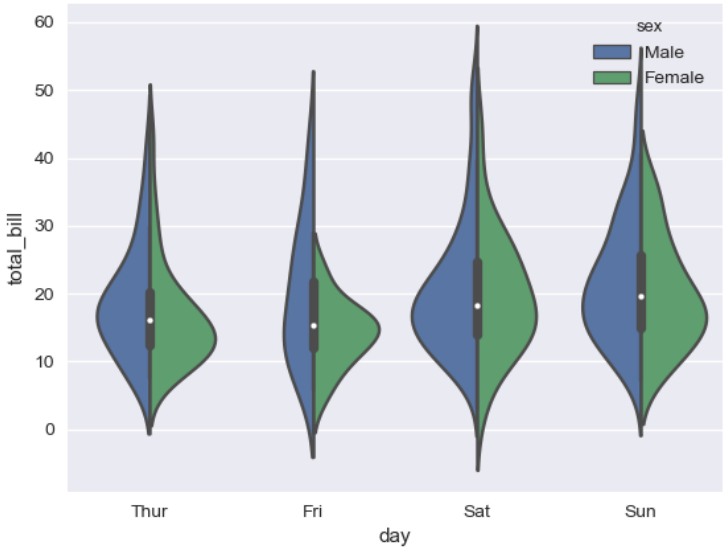


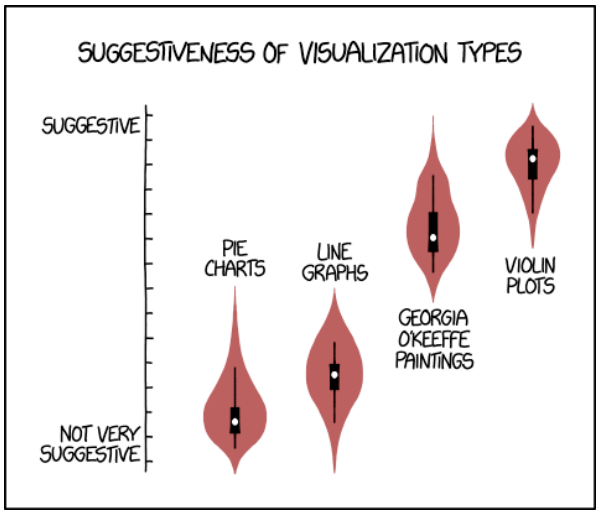

### Violin plot

```
sns.violinplot("day", "total_bill", "sex", data=
    tips)
```

# Seaborn

## Comparisons (II)



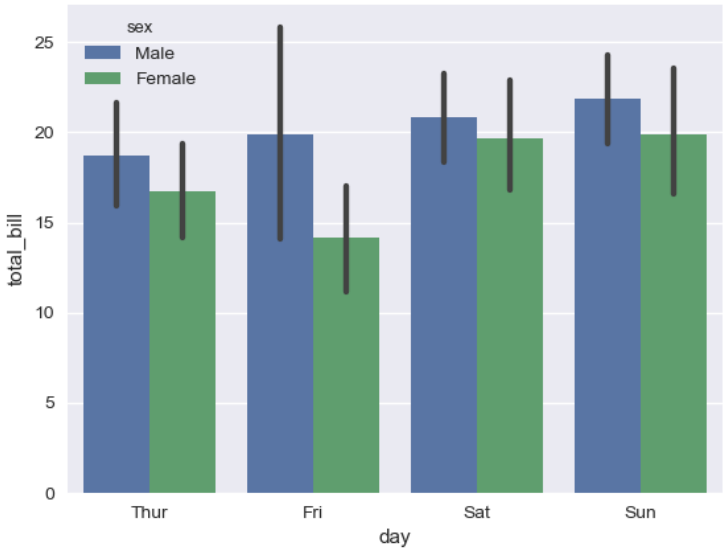

(Source)

### Violin plot

```
sns.violinplot(x="day", y="total_bill", hue="sex",
               data=tips, split=True)
```

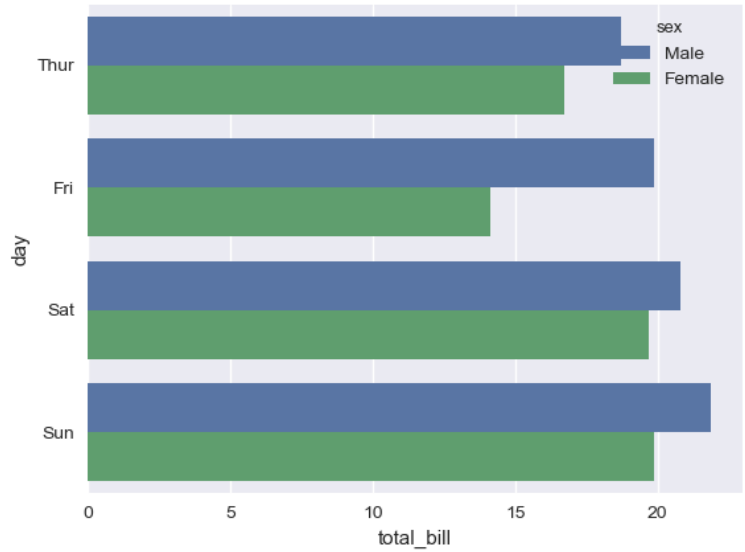
# Seaborn

## Barplots




### Barplot

```
sns.barplot(x="day", y="total_bill", hue="sex",
        data=tips)
```

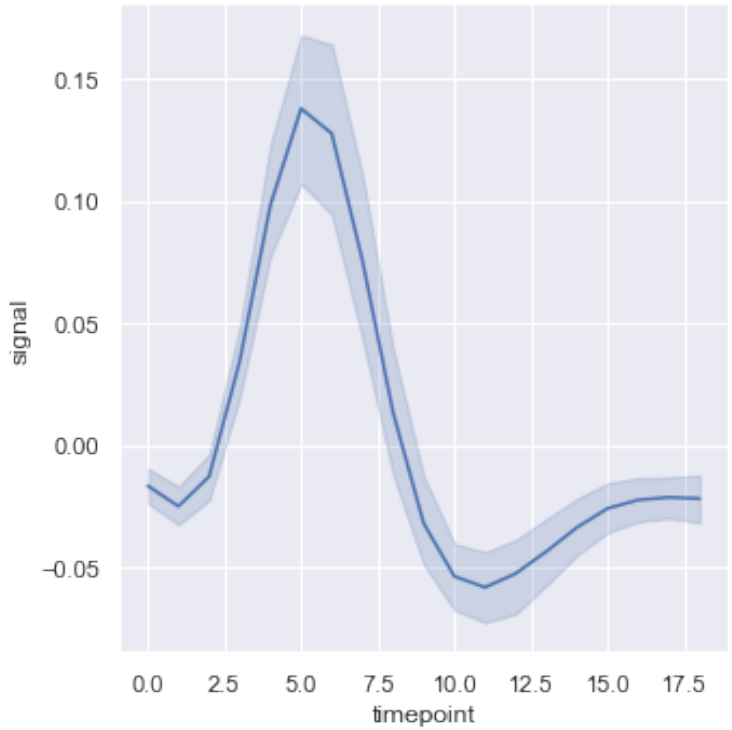


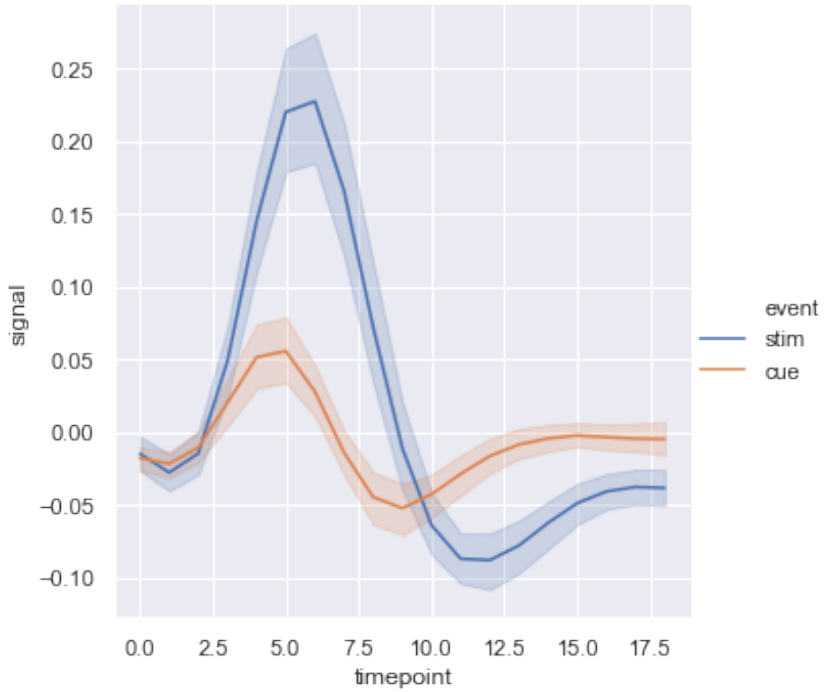

### Barplot

```
sns.barplot(x="total_bill", y="day", hue="sex",
        data=tips, ci=None)
```

# Seaborn

## Continuity



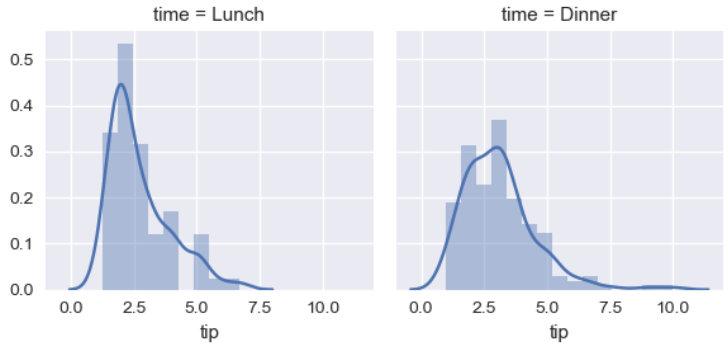




```
fmri = sns.load_dataset("fmri")
sns.relplot(x="timepoint", y="signal", kind="line",
            data=fmri)
```

```
sns.relplot(x="timepoint", y="signal", hue="event",
            kind="line", data=fmri)
```

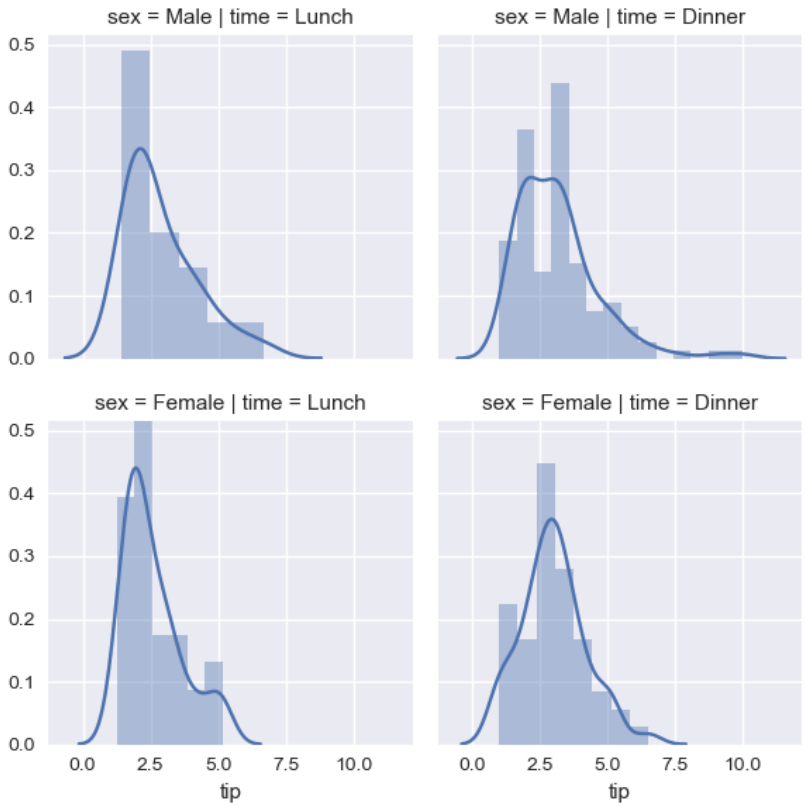
# Seaborn

## FacetGrid

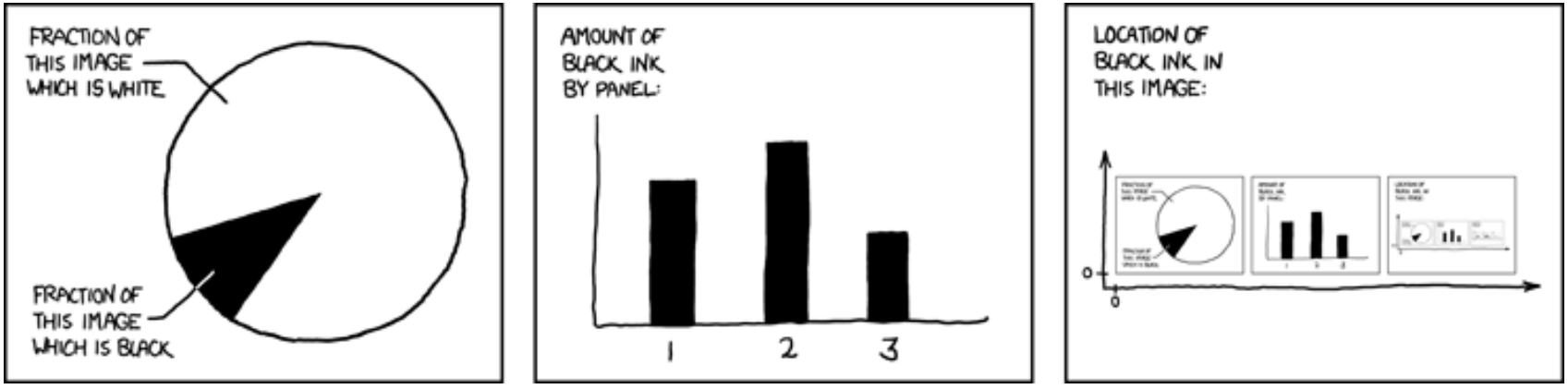



```
fmri = sns.load_dataset("fmri")
sns.relplot(x="timepoint", y="signal", kind="line",
            data=fmri)
```

Seaborn >= 0.9




```
g = sns.FacetGrid(tips, col="time", row="sex")
g.map(sns.distplot, "tip")
```


FRACTION OF THIS IMAGE WHICH IS WHITE
FRACTION OF THIS IMAGE WHICH IS BLACK
AMOUNT OF BLACK INK BY PANEL:
1
2
3
LOCATION OF BLACK INK IN THIS IMAGE:
0
0

(Source)

# Seaborn

## Seaborn notebook

### Seaborn notebook

(Link to notebook)