

EE4213 Human-Computer Interaction

Week 3 Data Visualization

Today's Menu

- Introduction to Data Visualization
- Introduction to Matplotlib
 - Line Plots
 - Histograms
 - Bar Charts
 - Box Plots
 - Scatter Plots
 - Bubble Plots
- Seaborn and Regression Plots
- Data Frame with Pandas
- Visualization of Geospatial Data
 - Introduction to Folium and Map Styles
 - Map with Marker
 - Choropleth Map
- Fun with Word Cloud

Reference:

- <https://matplotlib.org/>
- <https://aosabook.org/en/matplotlib.html>
- <https://www.coursera.org/learn/python-for-data-visualization>
- <https://seaborn.pydata.org/tutorial.html>
- <https://pandas.pydata.org/>
- Numerical Methods in Engineering with Python 3, Chapter 1.6
- “The Visual Display of Quantitative Information” by Edward R. Tufte

Graphical Excellence

- Well-designed presentation of interesting data
 - Substance
 - Statistics
 - Design
- Communicate complex ideas with
 - Clarity
 - Precision
 - Efficiency
- Gives to the viewer the greatest number of ideas in the shortest time with the least ink in the smallest space
- Nearly always multivariate
- Telling the truth about data

“The Visual Display of Quantitative Information” by Edward R. Tufte

Graphical displays should

- Show the data
- Induce the viewer to think about the substance rather than about methodology, graphic design, technology of graphic production, or something else
- Avoid distorting what the data have to say
- Present many numbers in a small space
- Make large datasets coherent
- Encourage the eye to compare different pieces of data
- Reveal the data at several levels of details, from a broad overview to fine structure
- Serve a reasonably clear purpose : description, exploration, tabulation or decoration
- Be closely integrated with statistical and verbal descriptions of a dataset

“The Visual Display of Quantitative Information” by Edward R. Tufte

Example: Anscombe's quartet

Data set	1-3	1	2	3	4	4
Variable	x	y	y	y	x	y
Obs. no. 1 :	10.0	8.04	9.14	7.46	8.0	6.58
2 :	8.0	6.95	8.14	6.77	8.0	5.76
3 :	13.0	7.58	8.74	12.74	8.0	7.71
4 :	9.0	8.81	8.77	7.11	8.0	8.84
5 :	11.0	8.33	9.26	7.81	8.0	8.47
6 :	14.0	9.96	8.10	8.84	8.0	7.04
7 :	6.0	7.24	6.13	6.08	8.0	5.25
8 :	4.0	4.26	3.10	5.39	19.0	12.50
9 :	12.0	10.84	9.13	8.15	8.0	5.56
10 :	7.0	4.82	7.26	6.42	8.0	7.91
11 :	5.0	5.68	4.74	5.73	8.0	6.89

TABLE. Four data sets, each comprising 11 (x, y) pairs.

Graphs in Statistical Analysis

F. J. Anscombe

The American Statistician

Vol. 27, No. 1 (Feb., 1973), pp. 17-21

4 datasets have same basic statistics outputs

Number of observations (n) = 11

Mean of the x 's ($\bar{x}$) = 9.0

Mean of the y 's ($\bar{y}$) = 7.5

Regression coefficient (b_1) of y on x = 0.5

Equation of regression line: $y = 3 + 0.5x$

Sum of squares of $x - \bar{x} = 110.0$

Regression sum of squares = 27.50 (1 d.f.)

Residual sum of squares of $y = 13.75$ (9 d.f.)

Estimated standard error of $b_1 = 0.118$

Multiple $R^2 = 0.667$

Graphs in Statistical Analysis

F. J. Anscombe

The American Statistician

Vol. 27, No. 1 (Feb., 1973), pp. 17-21

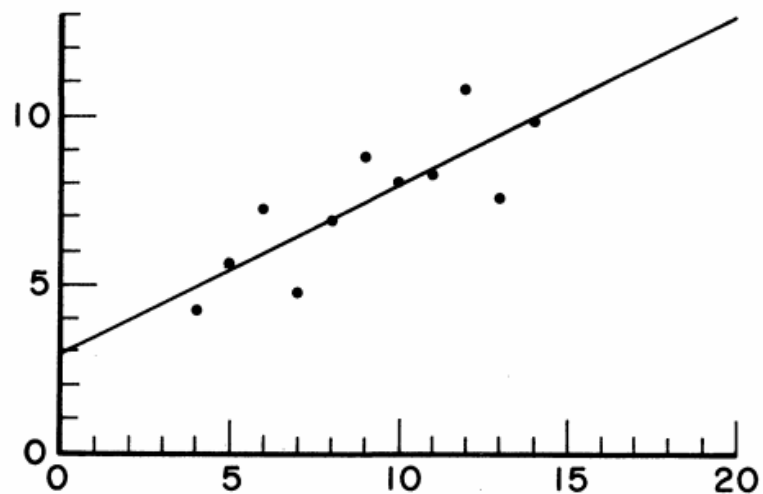


Figure 1

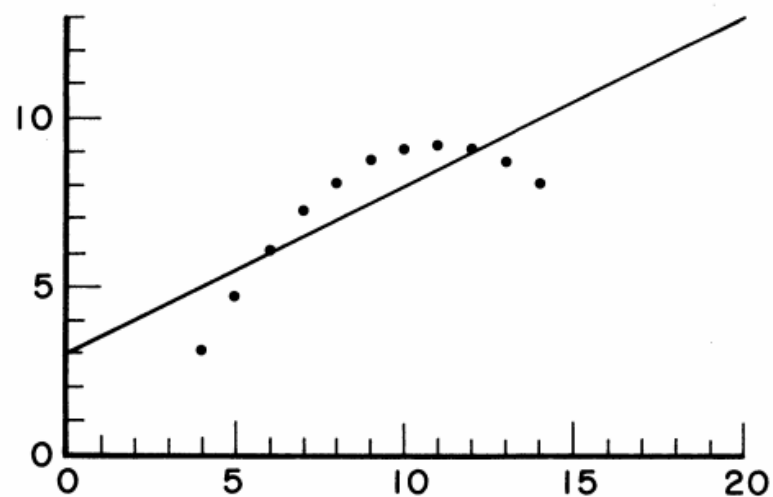


Figure 2

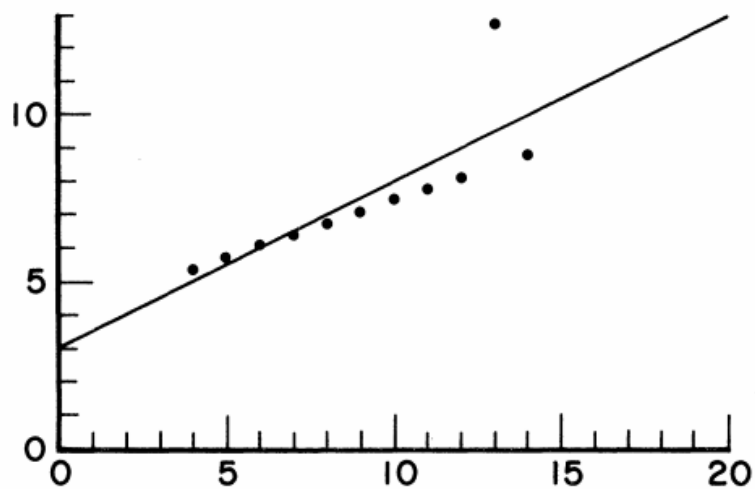


Figure 3

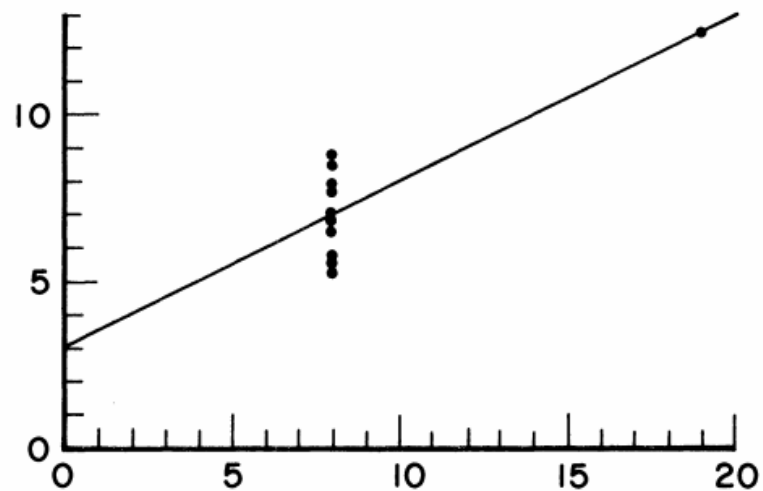


Figure 4

ACAP 1.

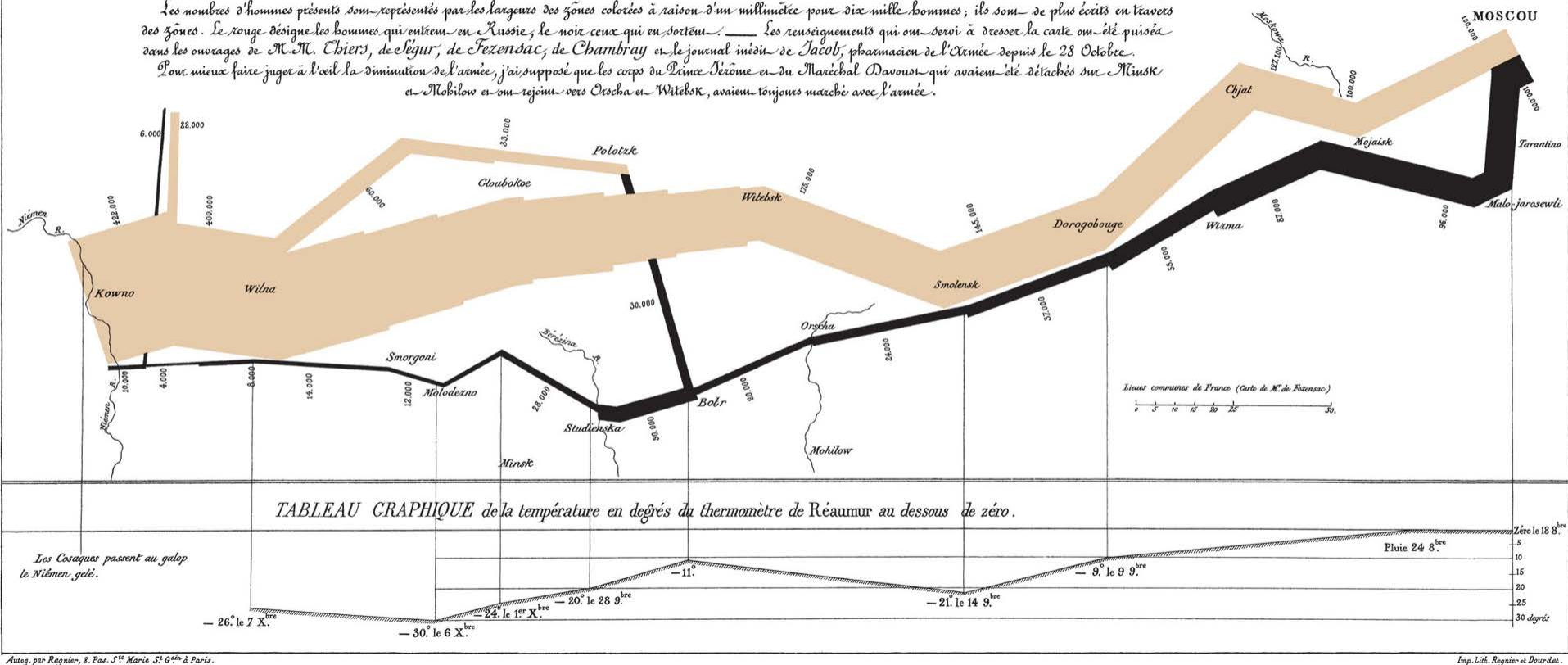


SCALE 30 INCHES TO A MILE.

Narrative Graphics of Space and Time

Carte Figurative des pertes successives en hommes de l'Armée Française dans la campagne de Russie 1812-1813.
Dressée par M. Minard, Inspecteur Général des Ponts et Chaussées en retraite. Paris, le 20 Novembre 1869.

Les nombres d'hommes présents sont représentés par les largeurs des zones colorées à raison d'un millimètre pour dix mille hommes; ils sont de plus écrits en travers des zones. Le rouge désigne les hommes qui entrent en Russie, le noir ceux qui en sortent. Les renseignements qui ont servi à dresser la carte ont été puisés dans les ouvrages de M. M. Chiers, de Légar, de Fezensac, de Chambray et le journal inédit de Jacob, pharmacien de l'Armée depuis le 28 Octobre. Pour mieux faire juger à l'œil la diminution de l'armée, j'ai supposé que les corps du Prince Jérôme et du Maréchal Davout, qui avaient été détachés sur Minsk et Mohilow et ont rejoint vers Orscha et Witebsk, avaient toujours marché avec l'armée.



Charles Joseph Minard's Carte Figurative illustrates facts related to French Invasion of Russia (1812)

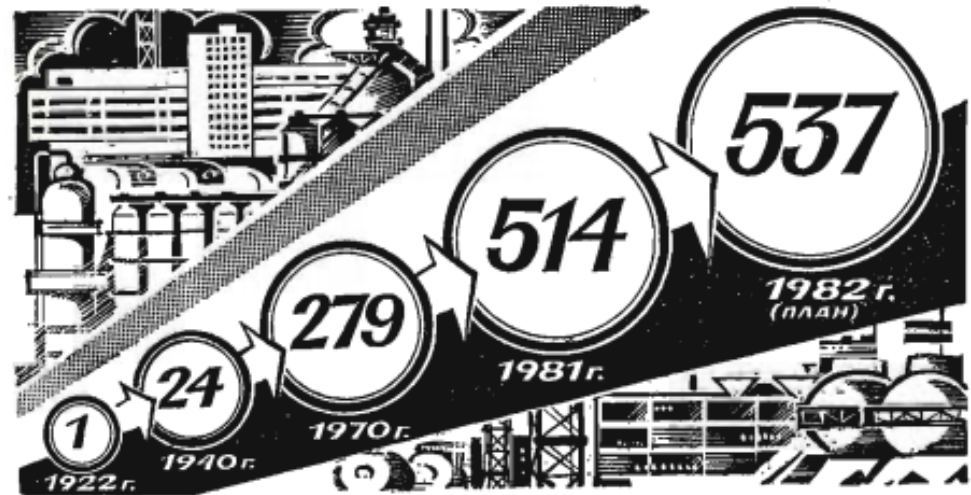
Graphical Integrity

- The representation of numbers, as physically measured on the surface of the graphics itself, should be directly proportional to the numerical quantities represented.
- Clearly, detailed and thorough labeling should be used to defeat graphical distortion and ambiguity.
 - Write out explanations of the data on the graphic itself.
 - Label important events in the data.
- Lie Factor = $\frac{\text{size of effect shown in graphic}}{\text{size of effect in data}}$
 - Lie Factor > 1.05 → overstating
 - Lie Factor < 0.95 → understating

What's the common problem in the following figures?

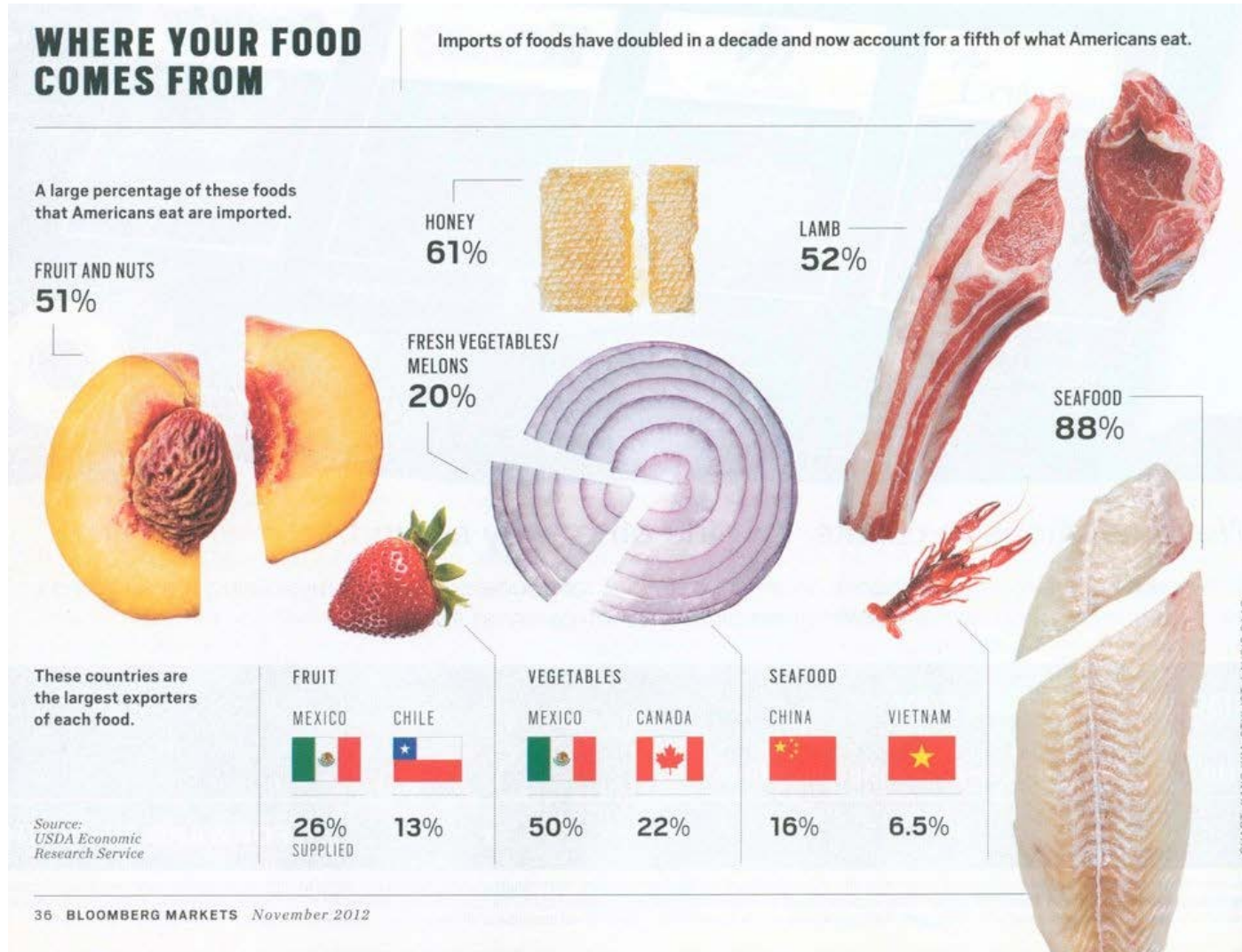


Washington Post, Oct 25, 1978, 1.



Pravda, May 24, 1982, 2.

More confusing...



<https://www.businessinsider.com/the-27-worst-charts-of-all-time-2013-6#did-anyone-learn-anything-by-looking-at-this-pseudo-pie-chart-what-do-these-colors-even-mean-why-is-it-divided-into-quadrants-we-ll-never-know-1>

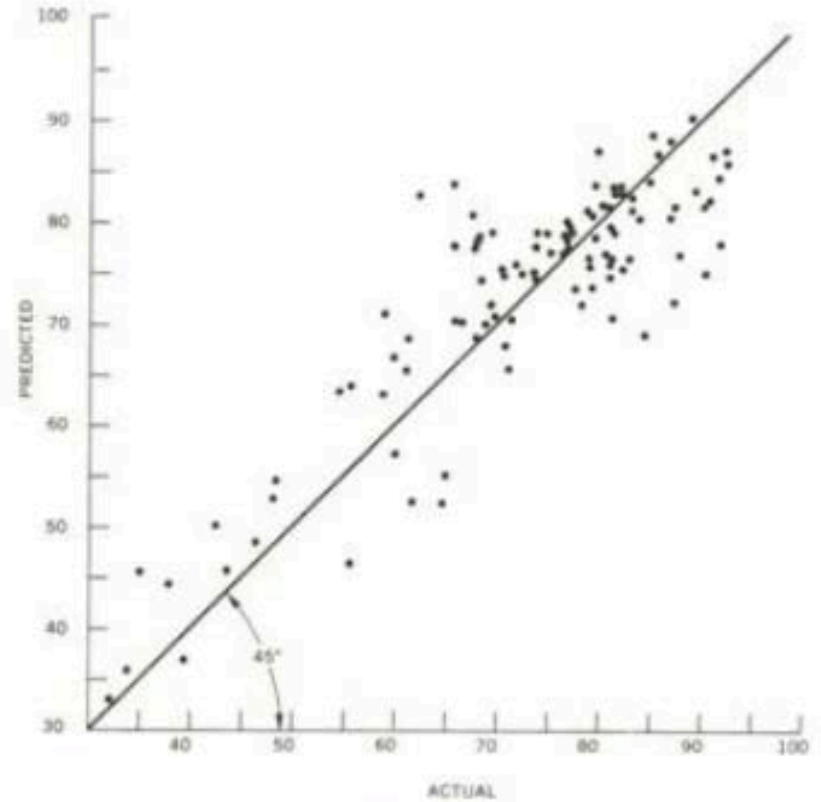
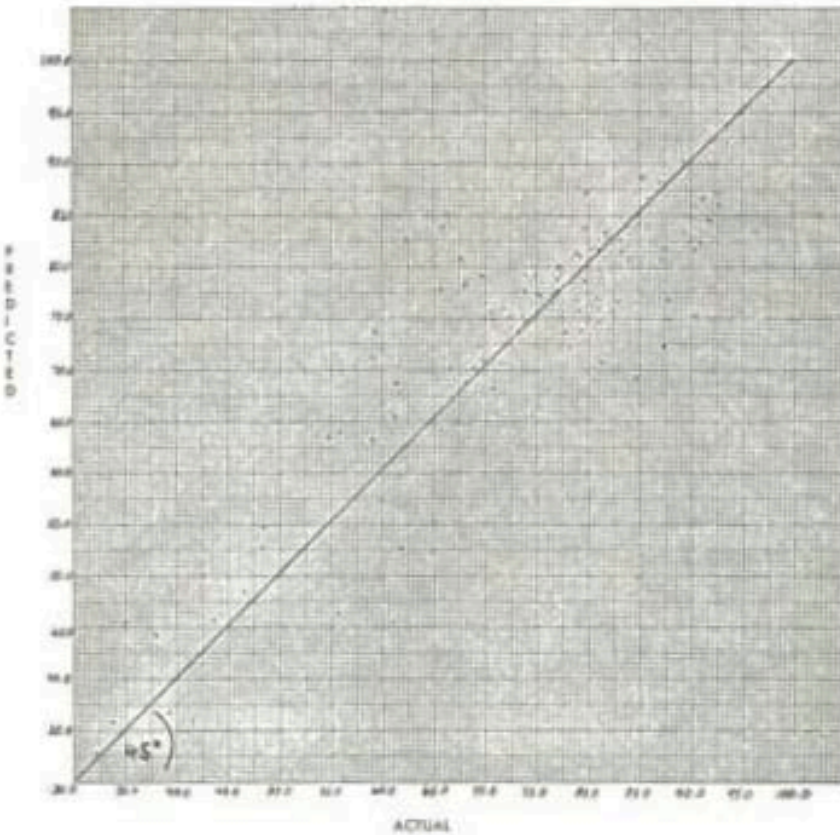
“**Data-Ink**”: A large share of ink (or **pixels**) on a graphic should present data information.

$$\begin{aligned} & \text{“Data – ink” ratio} \\ & \text{“data – ink”} \\ = & \frac{\text{data – ink}}{\text{total ink used to print the graphics}} \end{aligned}$$

= proportion of a graphic’s ink (or pixels) devoted to the non-redundant display of data-information

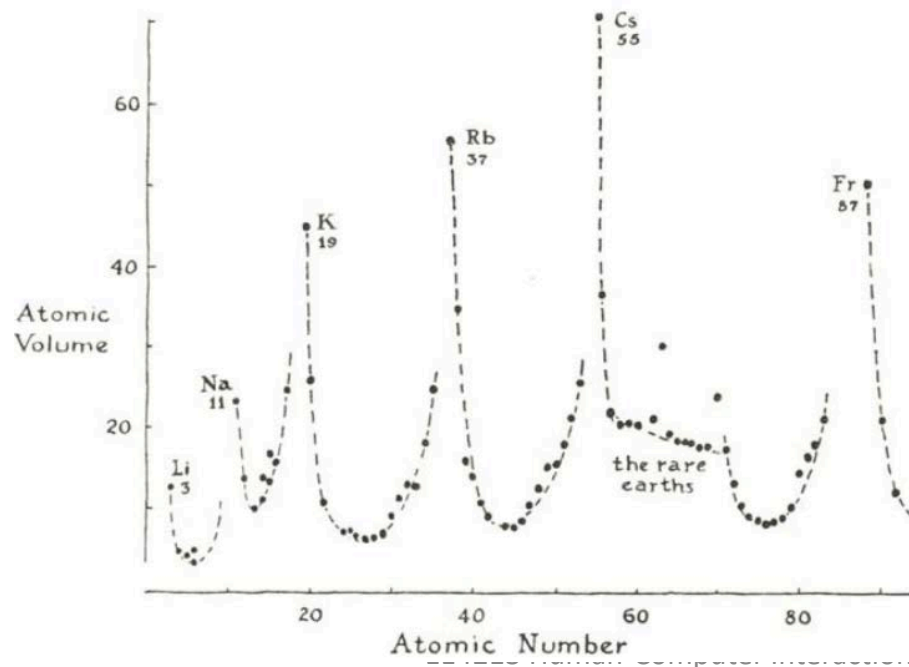
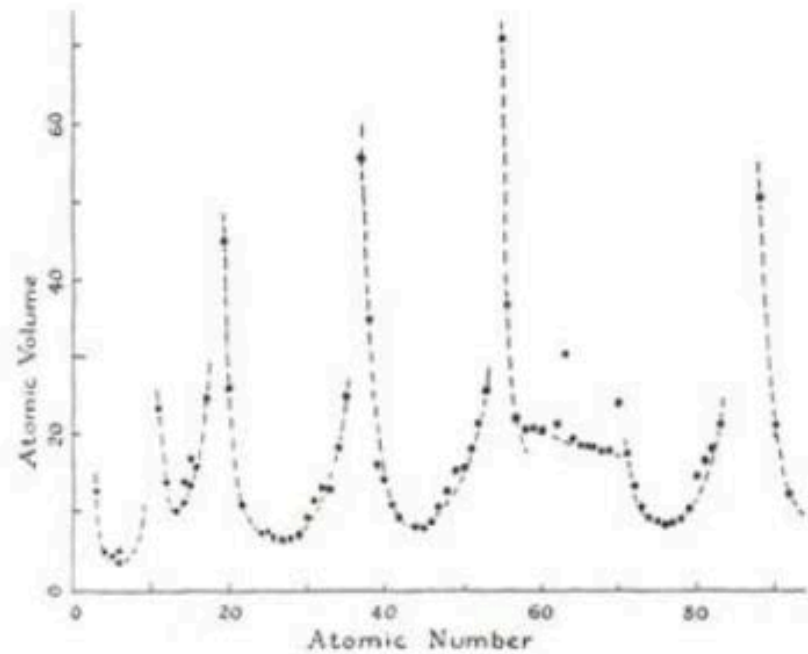
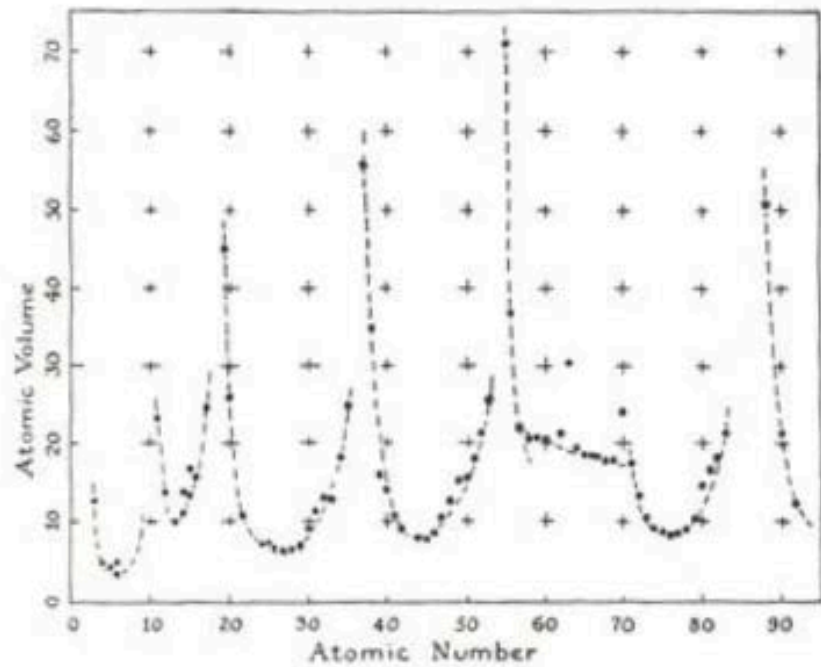
= 1.0 – proportion of a graphic that can be erased without loss of data-information

Relationship of Actual Rates of Registration to Predicted Rates
(104 cities 1960).



Relationship of Actual Rates of Registration to Predicted Rates (104 cities 1960).

“The Visual Display of Quantitative Information” by Edward R. Tufte

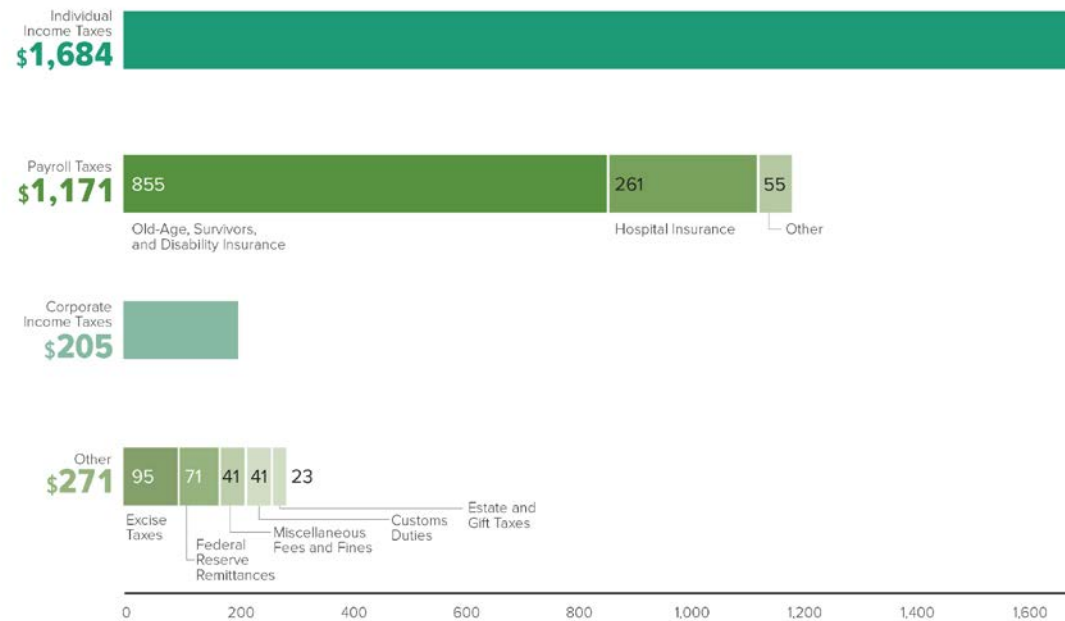


Linus Pauling, General Chemistry
(San Francisco, 1947), 64

Bar chart is more efficient than pie chart

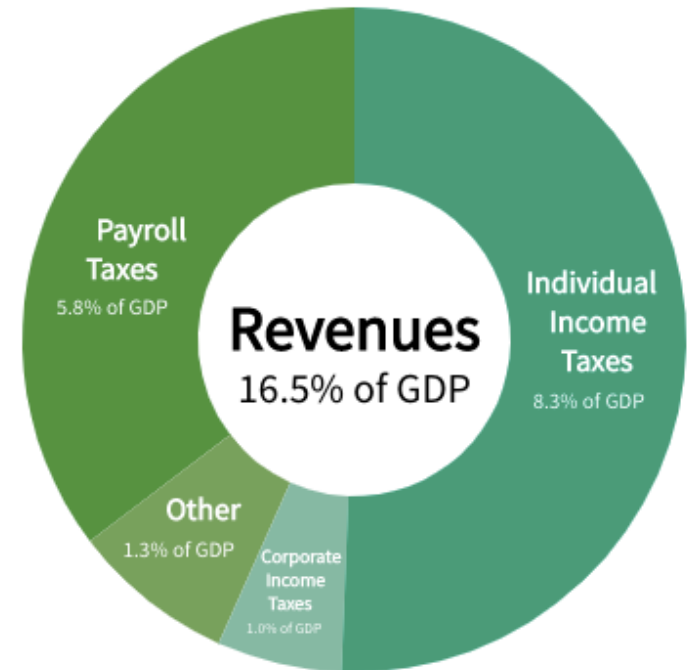
REVENUES IN 2018

Billions of Dollars

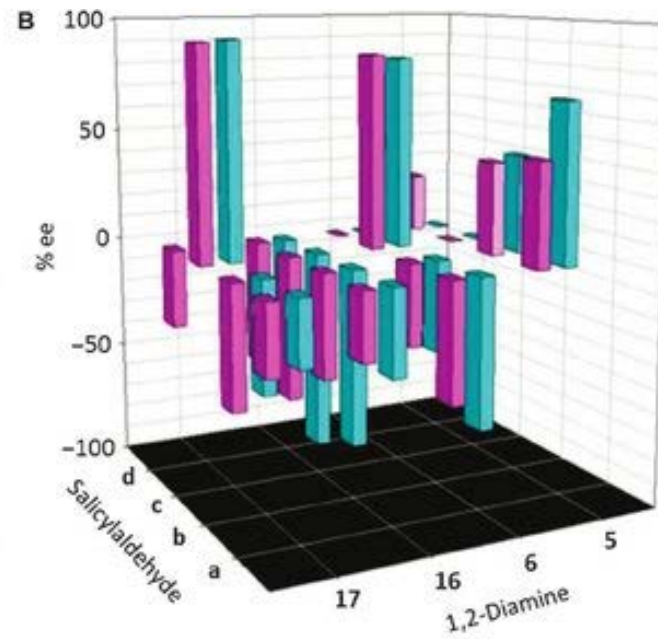
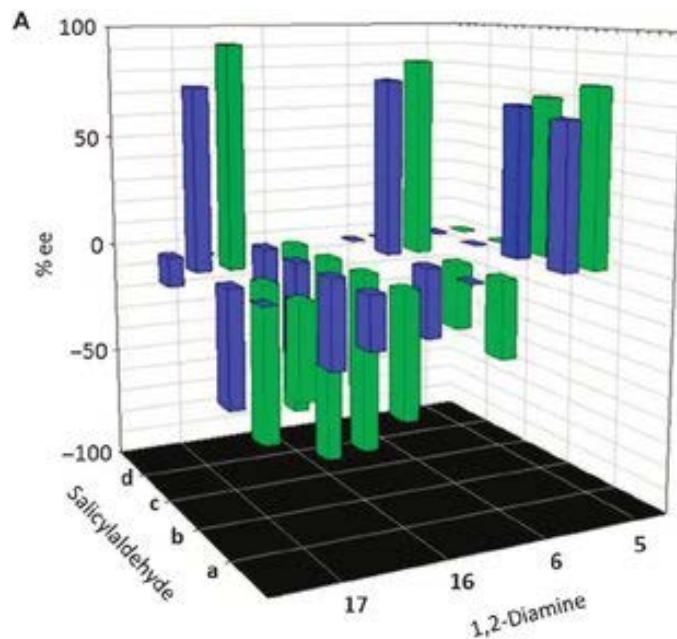


Prepared by Leigh Angres and Jorge Salazar
Source: Congressional Budget Office, June 2019

All data are for federal fiscal years,
which run from October 1 to September 30.
Numbers may not add up to totals because of rounding.



Avoid 3D charts



Mini-ISES Identifies Promising (Carba)fructopyranose-Based Salens for Asymmetric Catalysis: Tuning Ligand Shape via the Anomeric Effect,

Science Advances 1(6), July 2015

Plotting Figures via Python

Installation (Workable Offline)

The easiest way to install the *Jupyter Notebook App* is installing a scientific python distribution which also includes scientific python packages. The most common distribution is called **Anaconda**:

- Download [Anaconda Distribution](#) , Python 3, 64 bits.
- Install it using the default settings for a *single user*.

<https://www.anaconda.com/distribution/>

Anaconda Distribution

The World's Most Popular Python/R Data Science Platform

Download

The open-source **Anaconda Distribution** is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 15 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with **Conda**
- Develop and train machine learning and deep learning models with **scikit-learn**, **TensorFlow**, and **Theano**
- Analyze data with scalability and performance with **Dask**, **NumPy**, **pandas**, and **Numba**
- Visualize results with **Matplotlib**, **Bokeh**, **Datashader**, and **Holoviews**



 Windows |  macOS |  Linux

Home

Environments

Learning

Community

Documentation

Developer Blog



Applications on base (root)

Channels

Refresh



JupyterLab
1.1.4

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Launch



Notebook
6.0.1

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Launch



Spyder
3.3.6

Scientific PYTHON Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Launch



Glueviz
0.15.2

Multidimensional data visualization across files. Explore relationships within and among related datasets.

Install



Orange 3
3.23.0

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

Install



RStudio
1.1.456

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Install



VS Code
1.52.1

Streamlined code editor with support for development operations like debugging, task running and version control.

Install

Web-Based JupyterLab

<https://mybinder.org/v2/gh/jupyterlab/jupyterlab-demo/master?urlpath=lab/tree/demo>

← → ↻ <https://hub.gke.mybinder.org/user/jupyterlab-jupyterlab-demo-dftv720i/lab/workspaces/auto-K/tree/demo>

File Edit View Run Kernel Tabs Settings Help

+

+

+

+

📁 /

Name	Last Modified
📁 binder	a month ago
📁 data	a month ago
📁 demo	a month ago
Y: appveyor.yml	a month ago
📄 LICENSE	a month ago
M README.md	a month ago
Y: talks.yml	a month ago
🔗 tasks.py	a month ago

🔗 Launcher

Notebook

Python 3

C++11

C++14

C++17

R

Console

Python 3

C++11

C++14

C++17

R

Other

Terminal

Text File

Markdown File

Contextual Help

EE4213 Human-Computer Interaction

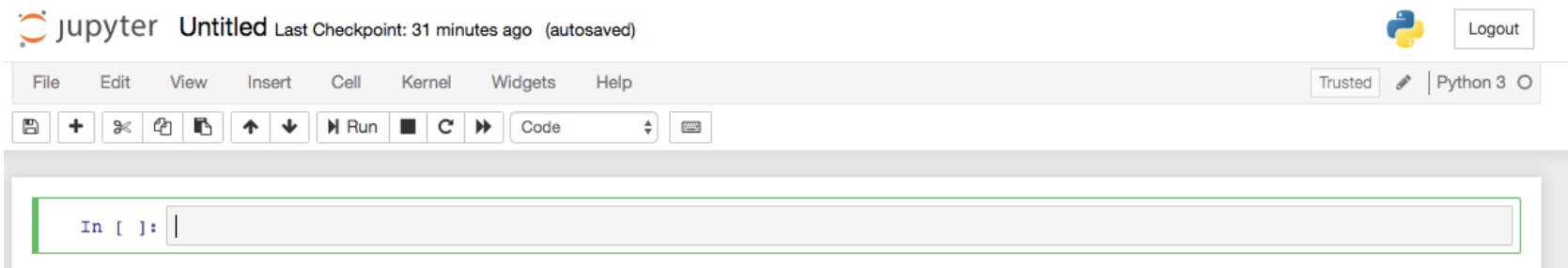
22

Jupyter Notebook

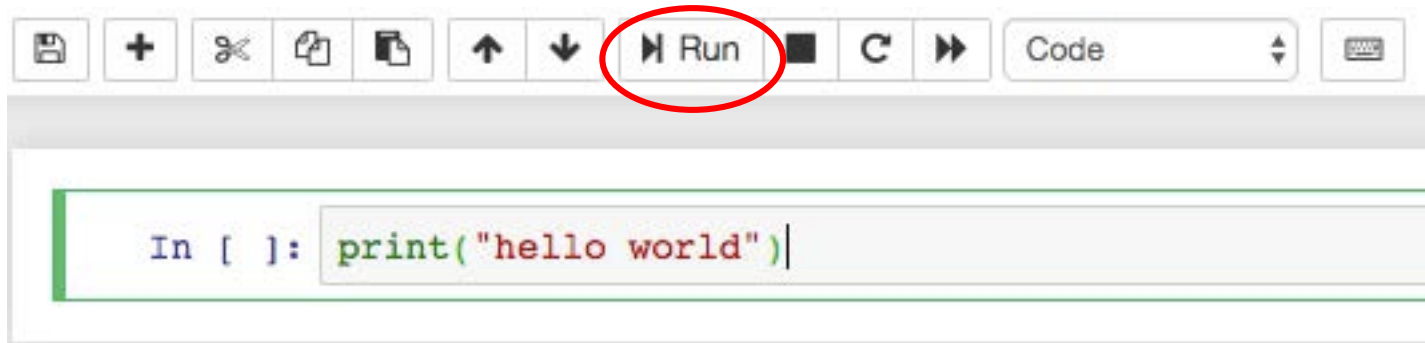
- Start a new one



- Or open an existing one



Hello World



```
prinq("hello world")
```

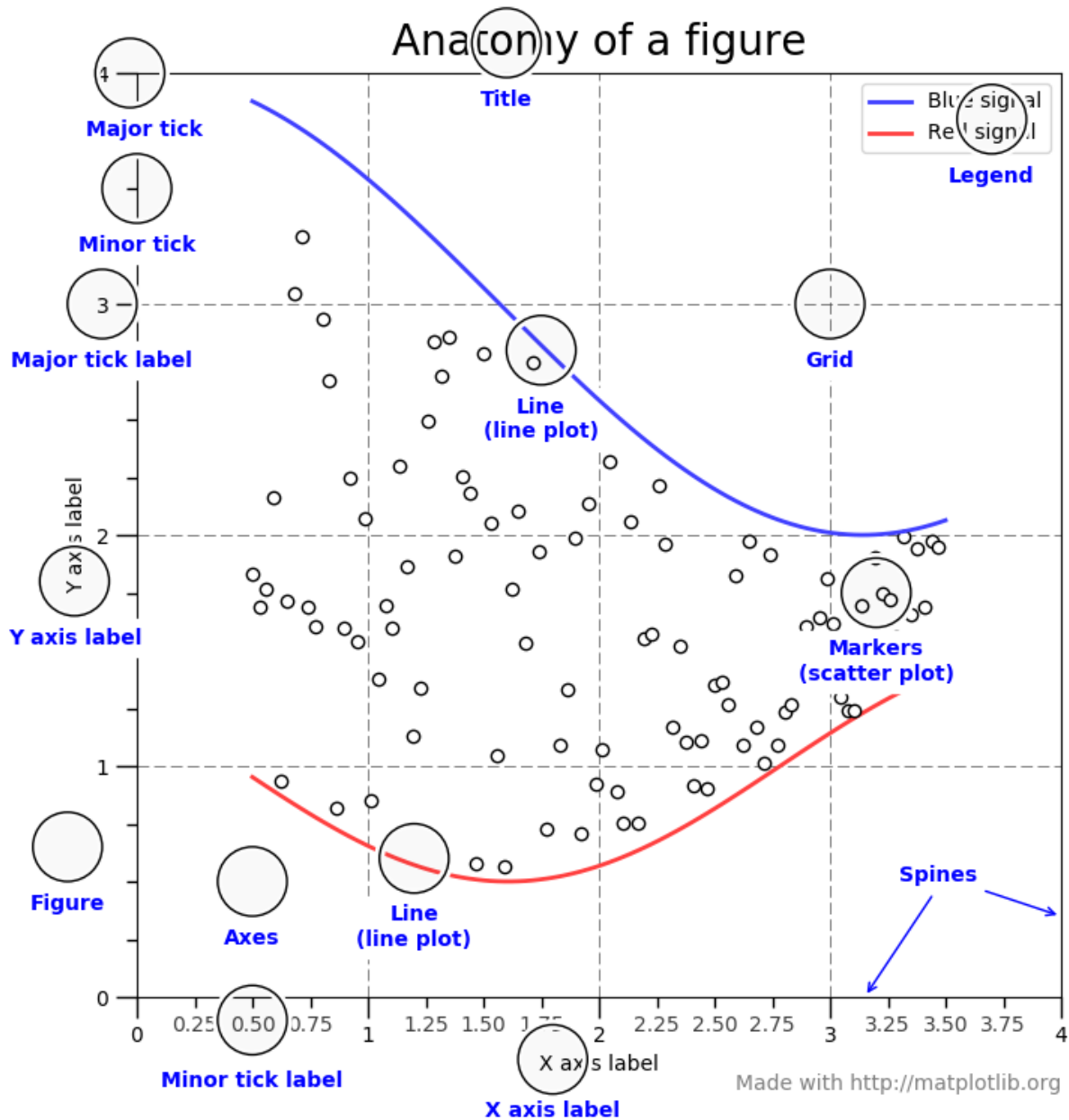
```
-----  
NameError                                Traceback (most recent call last)  
<ipython-input-2-9cf9f83ce359> in <module>  
----> 1 prinq("hello world")  
  
NameError: name 'prinq' is not defined
```



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

```
# usage similar to Matlab
```

<https://jupyter.org/>



Line Plot : Example

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

```
from numpy import arange,sin,cos
```

```
x = arange(0.0,8,0.2)
```

```
plt.plot(x,sin(x),'go-',x,cos(x),'b^--')
```

```
# Plot with specified line and marker style
```

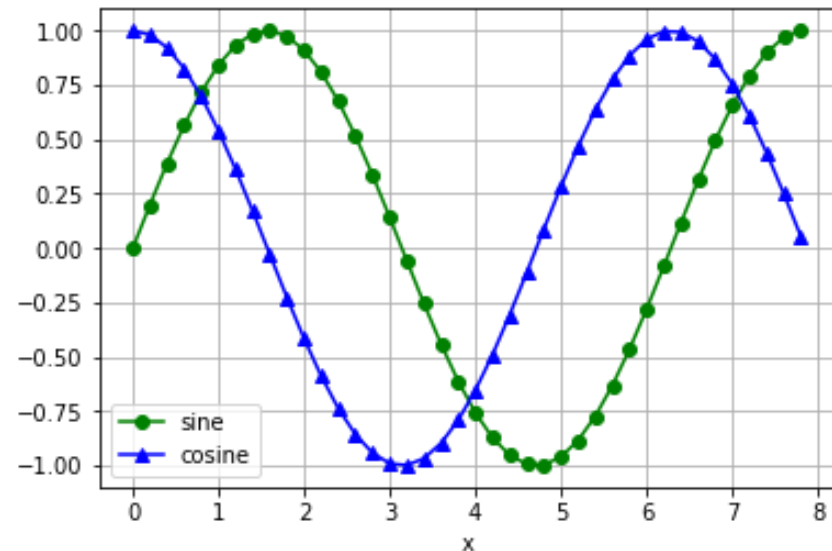
```
plt.xlabel('x') # Add label to x-axis
```

```
plt.legend(('sine','cosine'),loc = 0) # Add legend in loc. 0
```

```
plt.grid(True) # Add coordinate grid
```

```
plt.savefig('testplot.png',format='png') # Save plot in png
```

```
plt.show() # Show plot on screen
```



Line Plot Parameters

Markers

character	description
'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'1'	tri_down marker
'2'	tri_up marker
'3'	tri_left marker
'4'	tri_right marker
's'	square marker
'p'	pentagon marker
'*'	star marker
'h'	hexagon1 marker
'H'	hexagon2 marker
'+'	plus marker
'x'	x marker
'D'	diamond marker
'd'	thin_diamond marker
' '	vline marker
'_'	hline marker

Line Styles

character	description
'_'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
'...'	dotted line style

Colors

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

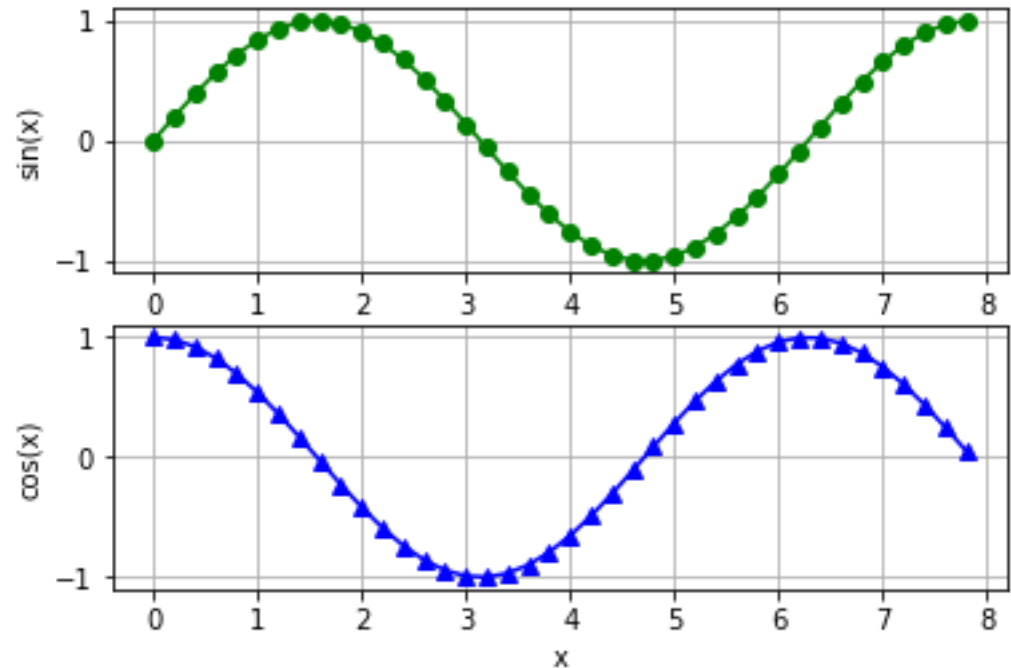
Line Plot Legend Parameters

Location String (loc)	Location Code
'best'	0
'upper right'	1
'upper left'	2
'lower left'	3
'lower right'	4
'right'	5
'center left'	6
'center right'	7
'lower center'	8
'upper center'	9
'center'	10

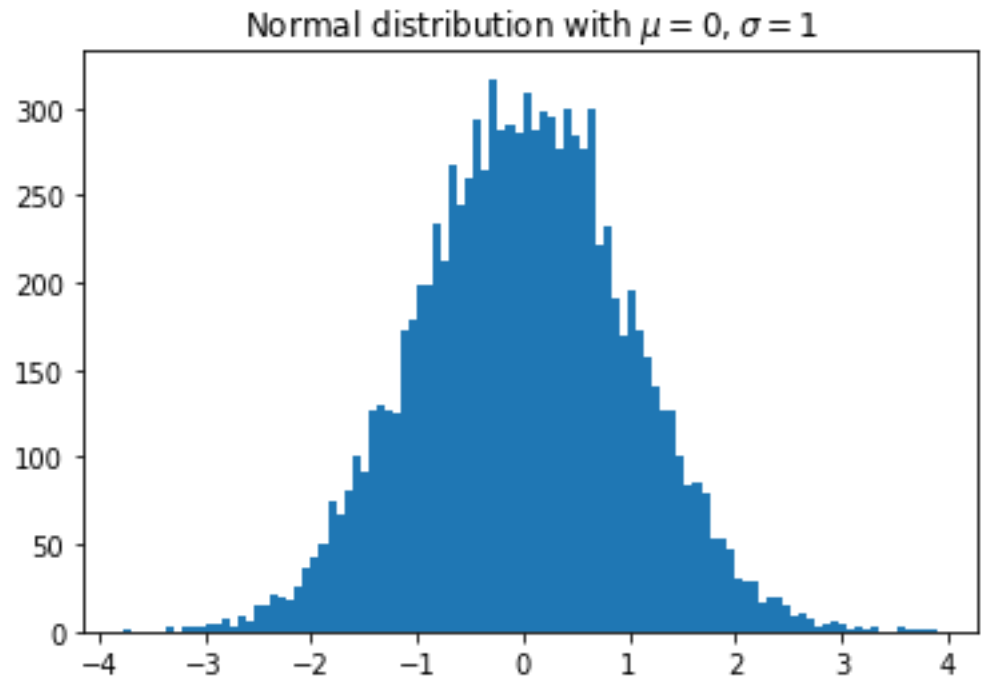
More than one plot in a figure:

`subplot(rows, cols, plot number)`

```
import matplotlib.pyplot as plt
from numpy import arange, sin, cos
x = arange(0.0, 8, 0.2)
plt.subplot(2,1,1)
plt.plot(x, sin(x), 'go-')
plt.xlabel('x');
plt.ylabel('sin(x)')
plt.grid(True)
plt.subplot(2,1,2)
plt.plot(x, cos(x), 'b^--')
plt.xlabel('x');
plt.ylabel('cos(x)')
plt.grid(True)
plt.show()
```



Histogram: Frequency distribution of a variable



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

```
x = np.random.randn(10000)  
plt.hist(x,100) # 100 bins  
plt.title(r'Normal distribution with  $\mu=0, \sigma=1$ ')  
plt.savefig('matplotlib_histogram.png')  
plt.show()
```

Text in Matplotlib Plots: Example

```
import matplotlib
import matplotlib.pyplot as plt

fig = plt.figure()
fig.suptitle('bold figure suptitle', fontsize=14, fontweight='bold')

ax = fig.add_subplot(111)
fig.subplots_adjust(top=0.85)
ax.set_title('axes title')

ax.set_xlabel('xlabel')
ax.set_ylabel('ylabel')

ax.text(3, 8, 'boxed italics text in data coords', style='italic',
        bbox={'facecolor': 'red', 'alpha': 0.5, 'pad': 10})

ax.text(2, 6, r'an equation:  $E=mc^2$ ', fontsize=15)

ax.text(3, 2, 'unicode: Institut für Festkörperphysik')

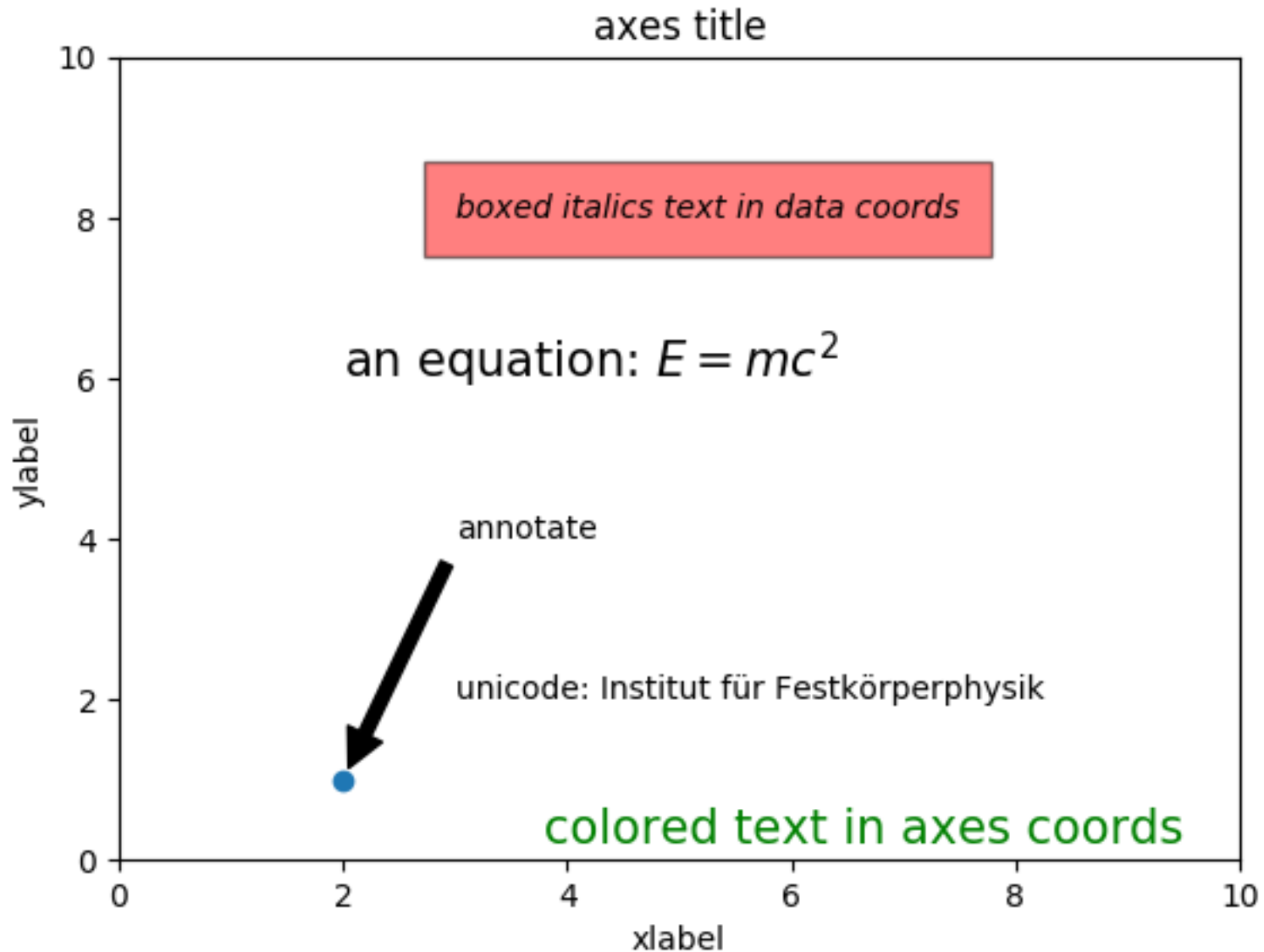
ax.text(0.95, 0.01, 'colored text in axes coords',
        verticalalignment='bottom', horizontalalignment='right',
        transform=ax.transAxes,
        color='green', fontsize=15)

ax.plot([2], [1], 'o')
ax.annotate('annotate', xy=(2, 1), xytext=(3, 4),
            arrowprops=dict(facecolor='black', shrink=0.05))

ax.axis([0, 10, 0, 10])

plt.show()
```

bold figure supitle



Bar Chart:

Compare variable values at a certain time

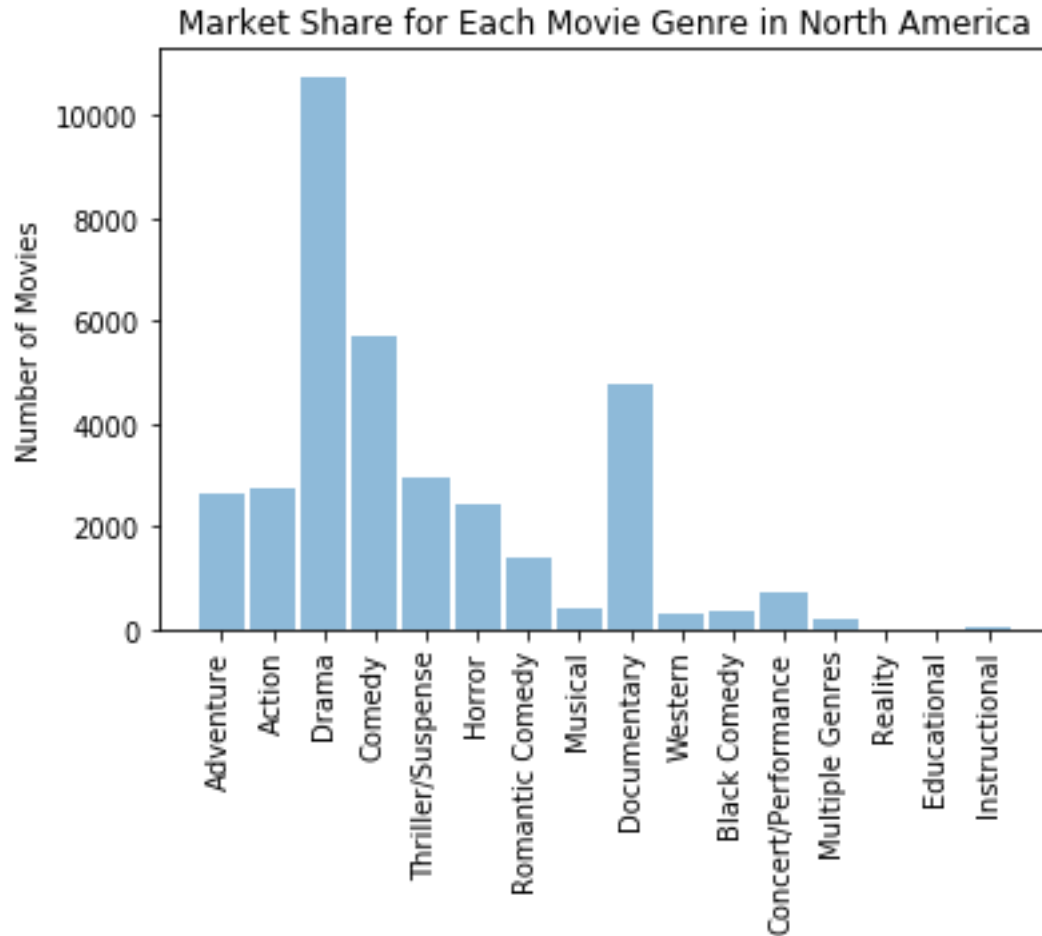
```
import matplotlib.pyplot as plt
import numpy as np

# data from https://www.the-numbers.com/movies/#tab=genre
genre = ('Adventure', 'Action', 'Drama', 'Comedy', 'Thriller/Suspense', 'Horror', 'Romantic Comedy',
         'Musical', 'Documentary', 'Western', 'Black Comedy', 'Concert/Performance', 'Multiple Genres',
         'Reality', 'Educational', 'Instructional')
total_movies = [2644, 2765, 10756, 5697, 2928, 2424, 1372, 403, 4772, 293, 330, 703, 182, 7, 7, 19]

x_pos = np.arange(len(genre))
plt.bar(x_pos, total_movies, align='center', width=0.9, alpha=0.5)
plt.xticks(x_pos, genre, rotation=90)
plt.ylabel('Number of Movies')
plt.title('Market Share for Each Movie Genre in North America')

plt.show()
```

Bar Chart Example I



Check the following link for more examples:

https://matplotlib.org/3.1.1/api/as_gen/matplotlib.pyplot.bar.html

Horizontal bar chart

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

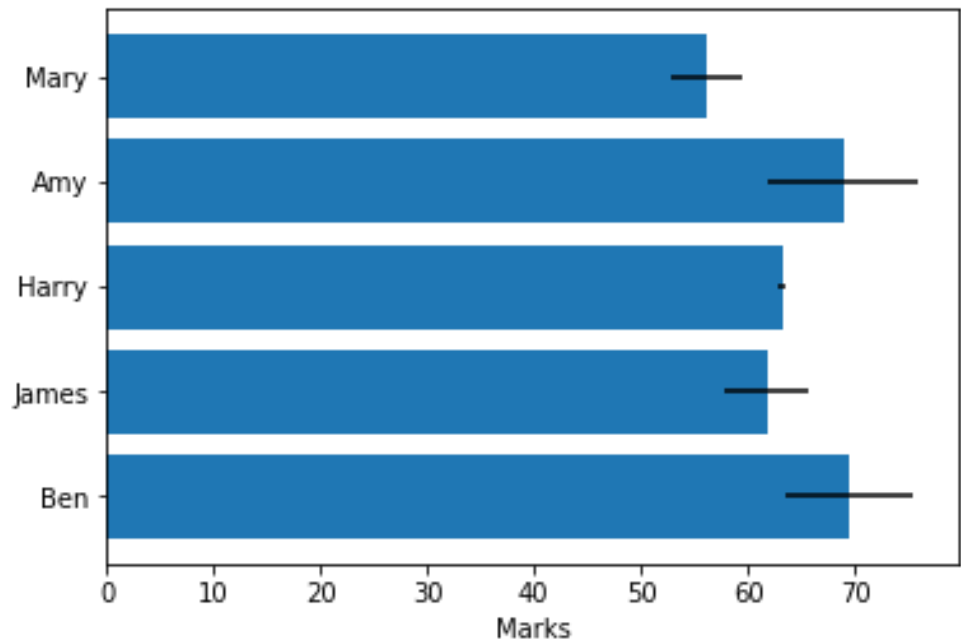
# Fixing random state for reproducibility
np.random.seed(312489359)

fig, ax = plt.subplots()

# Generate some random data for demo
people = ('Mary', 'Amy', 'Harry', 'James', 'Ben')
y_pos = np.arange(len(people))
marks = 50 + 20 * np.random.rand(len(people))
error = 10 * np.random.rand(len(people))

ax.barh(y_pos, marks, xerr=error, align='center')
ax.set_yticks(y_pos)
ax.set_yticklabels(people)
ax.invert_yaxis() # labels read top-to-bottom
ax.set_xlabel('Marks')

plt.show()
```



https://matplotlib.org/3.1.1/gallery/lines_bars_and_markers/barh.html

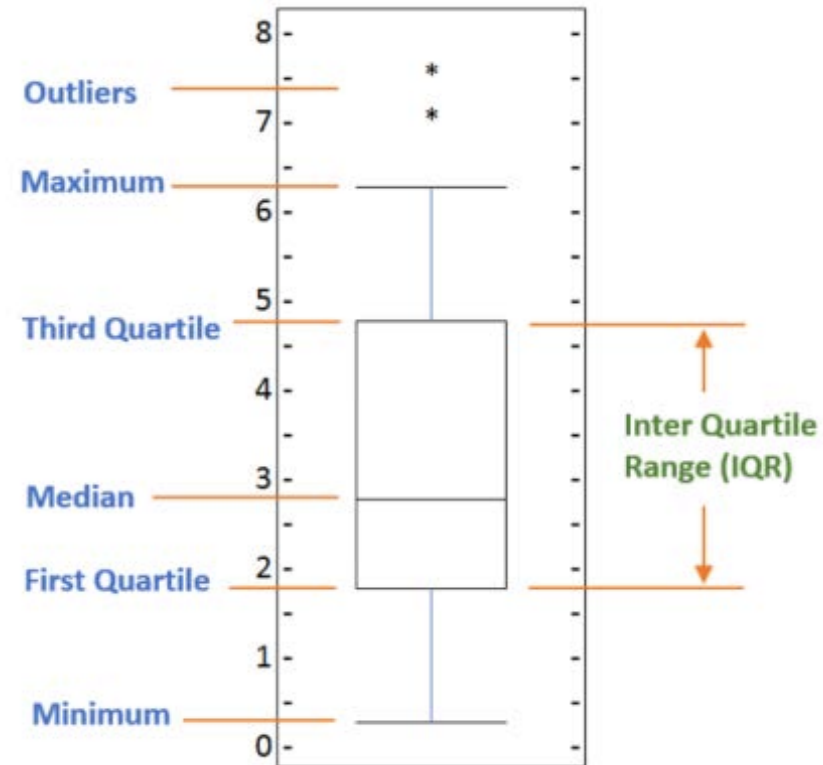
Exercise

Change bar chart example I to a horizontal bar chart.

Box Plot

Statistically representing the *distribution* of the data through 5 main dimensions:

- **Minimum:** Smallest number in the dataset.
- **1st quartile:** Middle number between the minimum and the median.
- **2nd quartile (Median):** Middle number of the (sorted) dataset.
- **3rd quartile:** Middle number between median and maximum.
- **Maximum:** Highest number in the dataset.



<https://labs.cognitiveclass.ai/tools/jupyterlab/lab/tree/labs/DV0101EN/DV0101EN-2-3-1-Pie-Charts-Box-Plots-Scatter-Plots-and-Bubble-Plots-py-v2.0.ipynb>

```

import numpy as np
import matplotlib.pyplot as plt

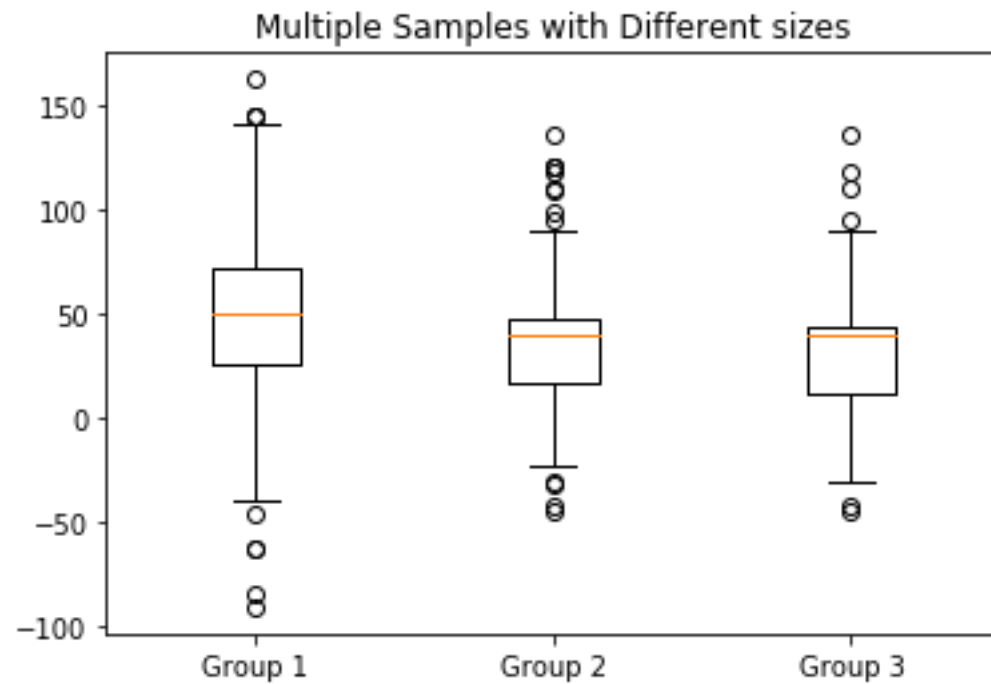
# fake up some data
spread = np.random.rand(50) * 100
center = np.ones(25) * 50
flier_high = np.random.rand(10) * 100 + 100
flier_low = np.random.rand(10) * -100
data = np.concatenate((spread, center, flier_high, flier_low))
d2 = np.concatenate((spread/1.5, center-10, flier_high/1.2, flier_low/2))

data.shape = (-1, 1)
d2.shape = (-1, 1)

data = [data, d2, d2[::2,0]] # "::2" means every 2nd element of the sequence
fig7, ax7 = plt.subplots()
ax7.set_title('Multiple Samples with Different sizes')
ax7.boxplot(data)
ax7.set_xticklabels(('Group 1', 'Group 2', 'Group 3'))

plt.show()

```



Scatter Plot in Groups

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

```
# Create data
```

```
N = 80
```

```
g1 = (0.7 + 0.6 * np.random.rand(N), np.random.rand(N))
```

```
g2 = (0.4 + 0.3 * np.random.rand(N), 0.5*np.random.rand(N))
```

```
g3 = (0.2*np.random.rand(N),0.3*np.random.rand(N))
```

```
data = (g1, g2, g3)
```

```
colors = ("red", "green", "blue")
```

```
groups = ("Group #1", "Group #2", "Group #3")
```

```
for group in arange(len(groups)):
```

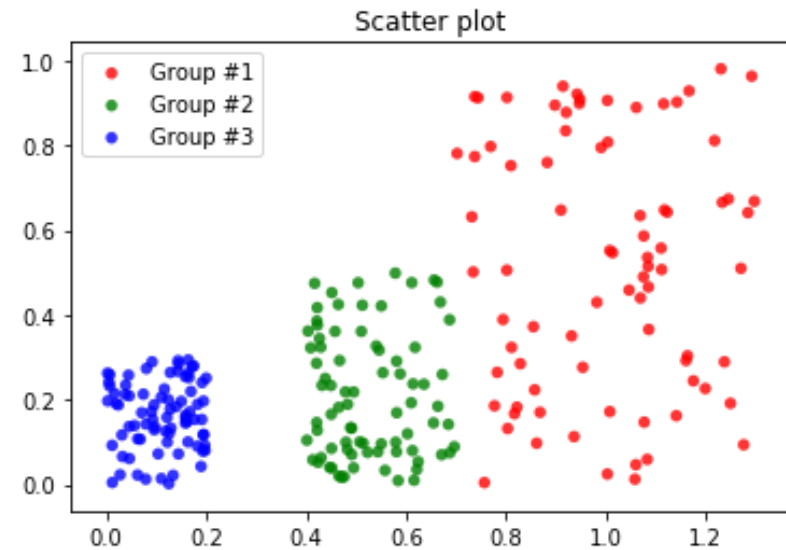
```
    x, y = data[group]
```

```
    plt.scatter(x, y, alpha=0.8, c=colors[group], edgecolors='none', s=30, label=groups[group])
```

```
plt.title('Scatter plot')
```

```
plt.legend(loc=2)
```

```
plt.show()
```

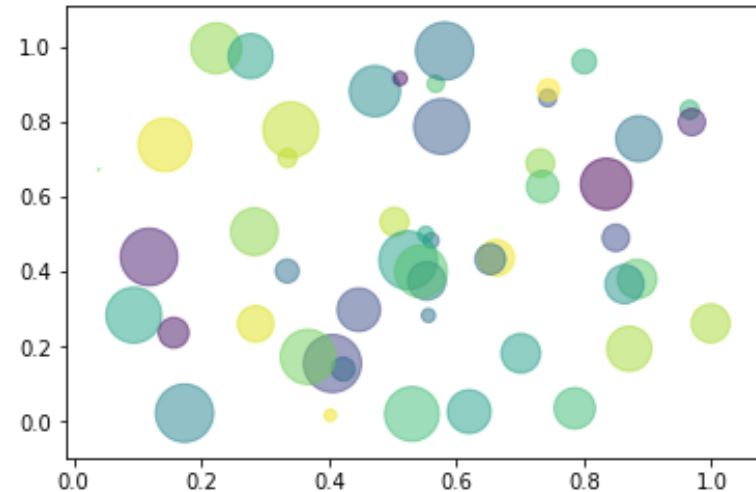


Scatter Plot → Bubble Chart (Extra dimension)

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

N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = (30 * np.random.rand(N))**2
# 0 to 15 point radii

plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```



Visualizing Linear Relationships with **Seaborn**

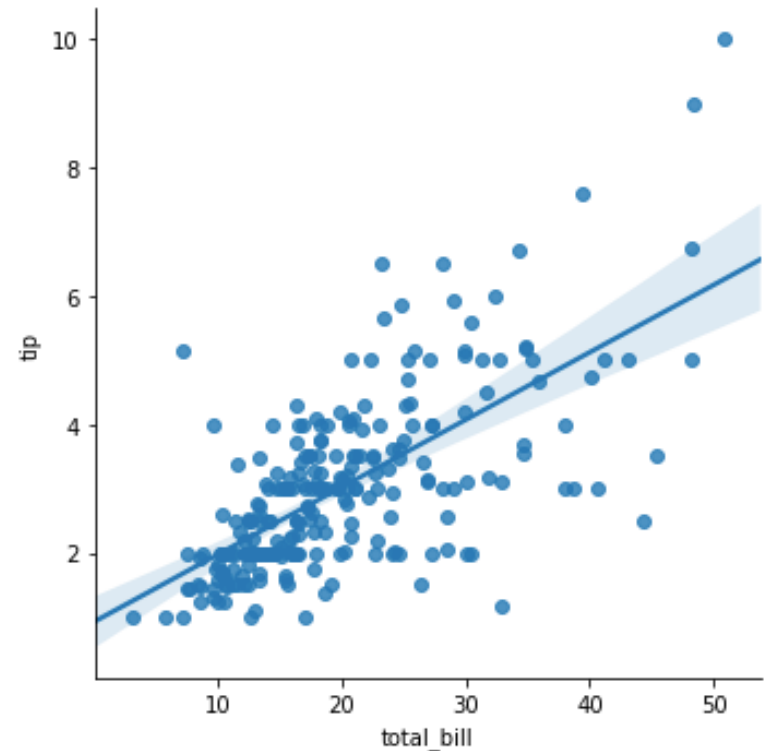
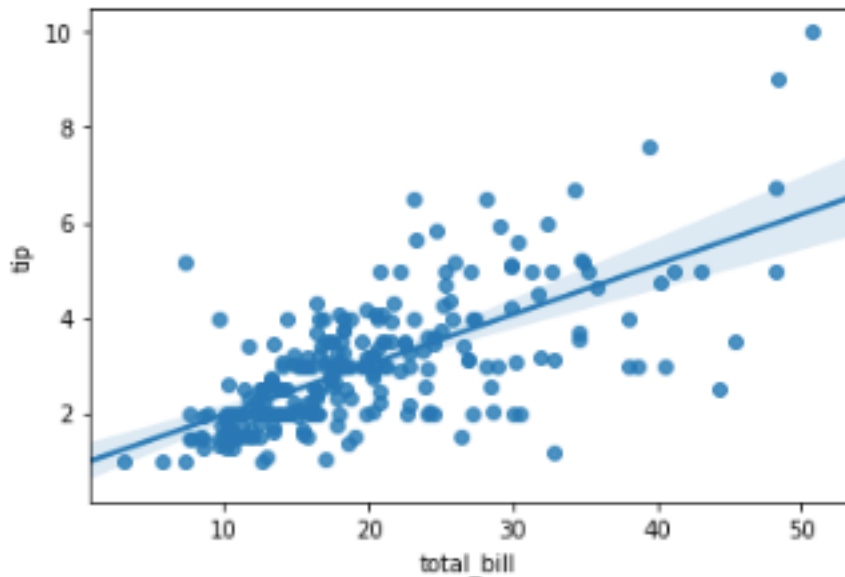
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Load a demo dataset
tips = sns.load_dataset("tips")
```

<https://seaborn.pydata.org/tutorial/regression.html>
<https://github.com/mwaskom/seaborn-data>

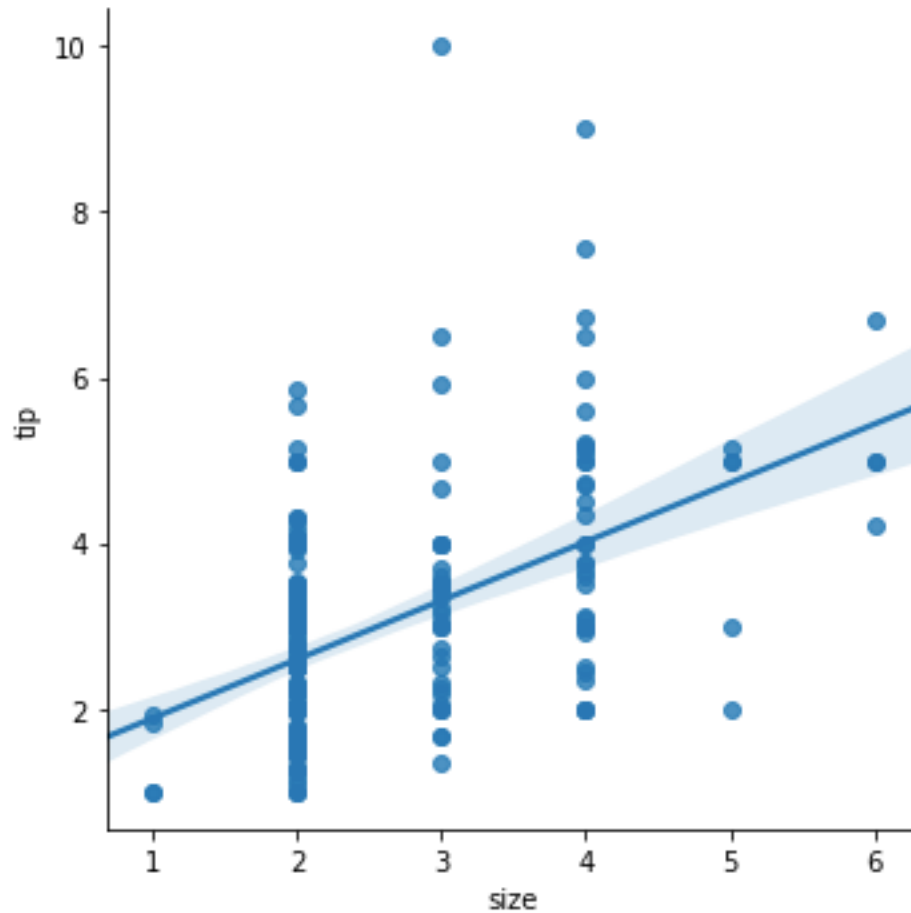
Functions to Draw Line Regression Models: Different in x, y data formats

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



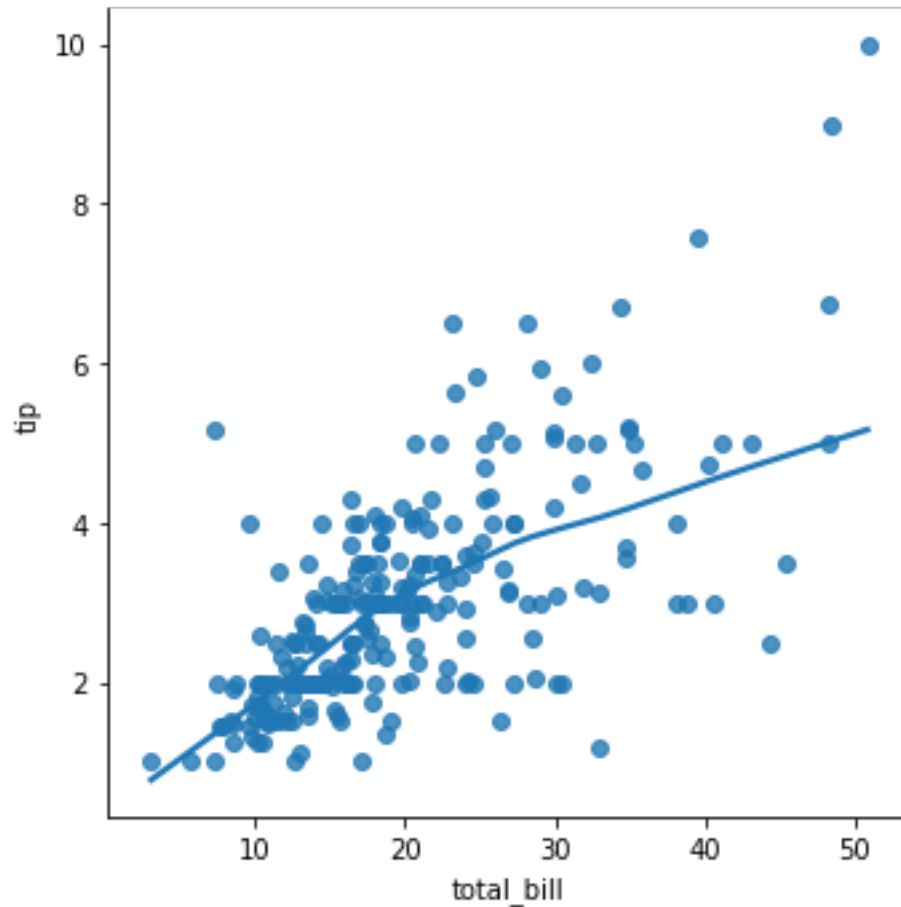
```
sns.lmplot(x="total_bill", y="tip", data=tips);
```

```
sns.lmplot(x="size", y="tip", data=tips);
```



Locally weighted scatterplot smoothing

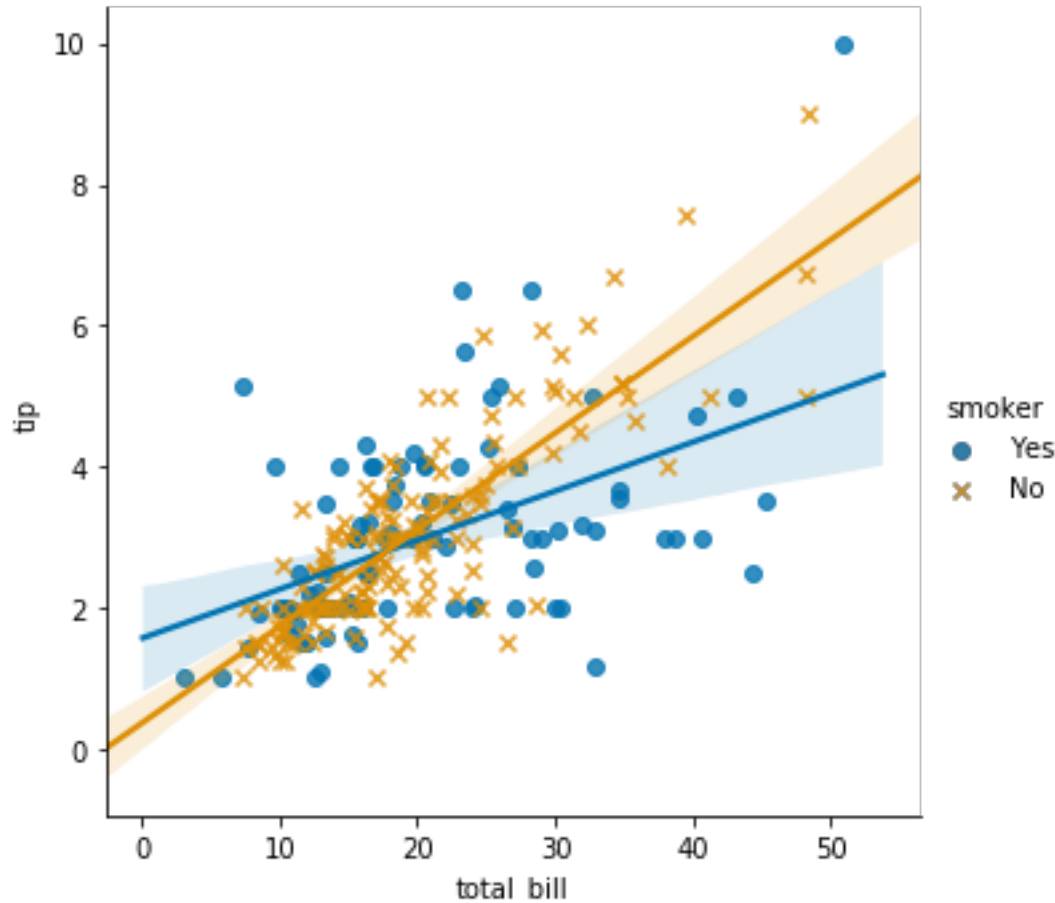
```
sns.lmplot(x="total_bill", y="tip", data=tips, lowess=True);
```



<https://seaborn.pydata.org/tutorial/regression.html>

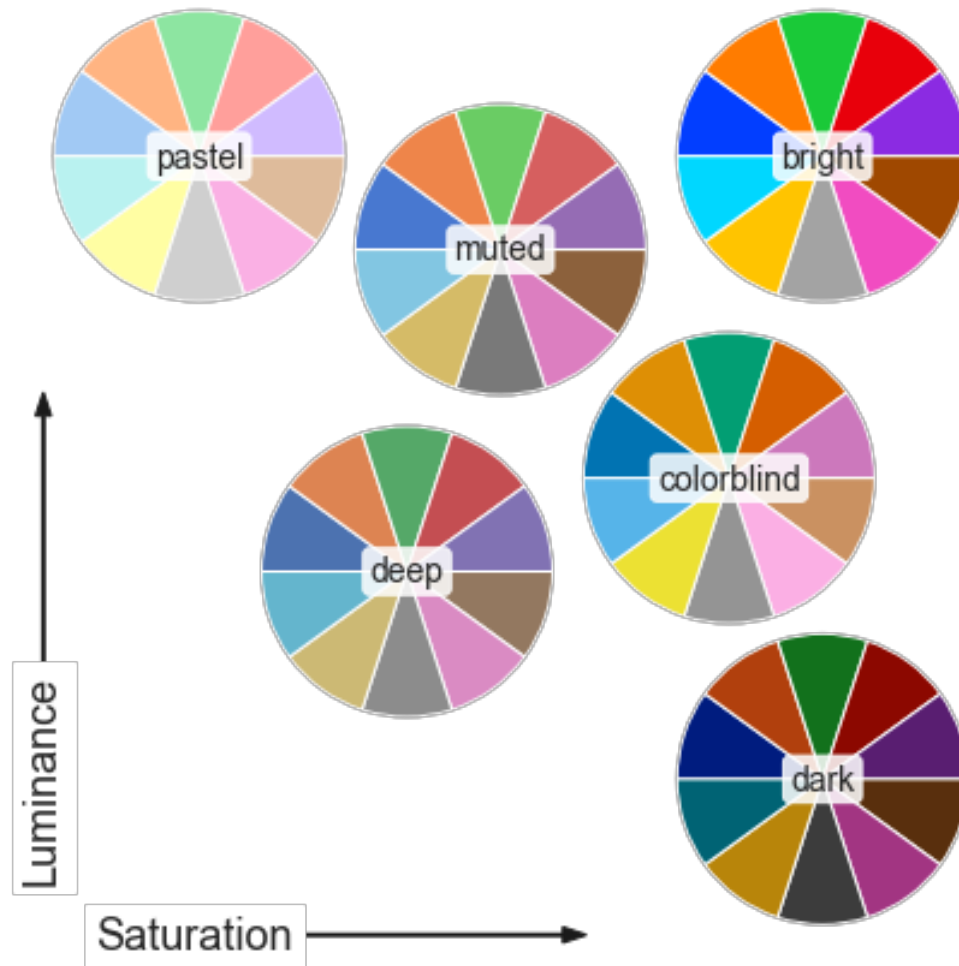
Conditioning on Other Variables

```
sns.lmplot(x="total_bill", y="tip", hue="smoker",  
data=tips, markers=["o", "x"], palette="colorblind");
```



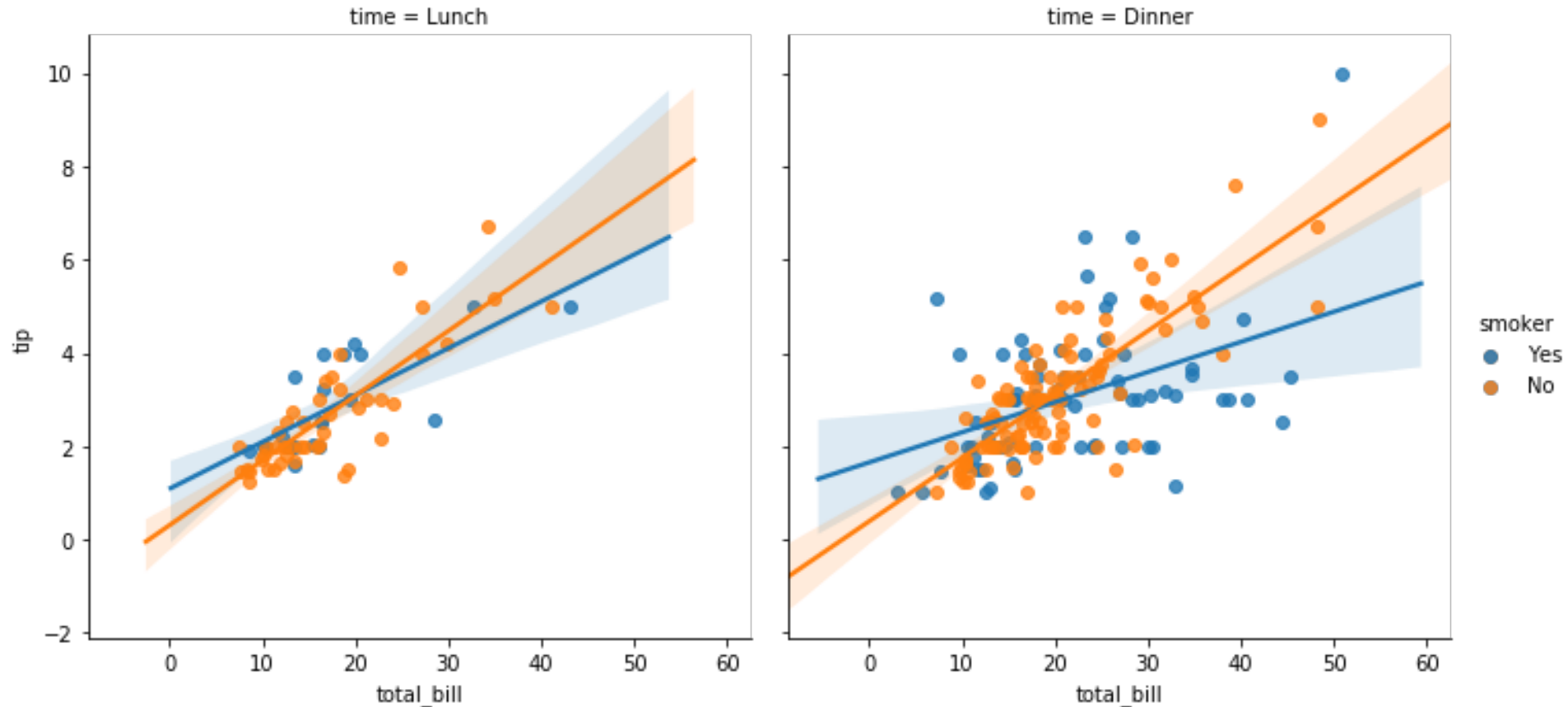
<https://seaborn.pydata.org/tutorial/regression.html>

Color Palettes in Seaborn



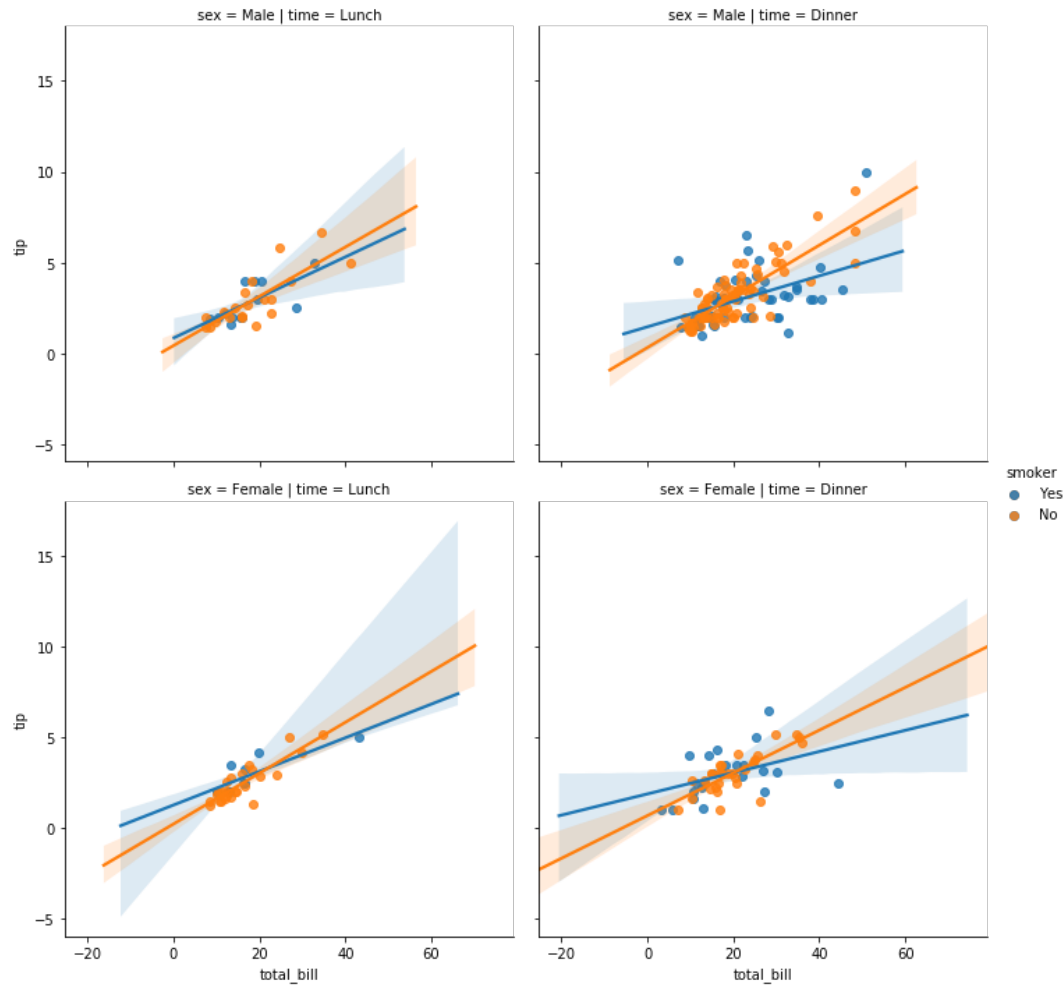
https://seaborn.pydata.org/tutorial/color_palettes.html

```
sns.lmplot(x="total_bill", y="tip",  
hue="smoker", col="time", data=tips);
```



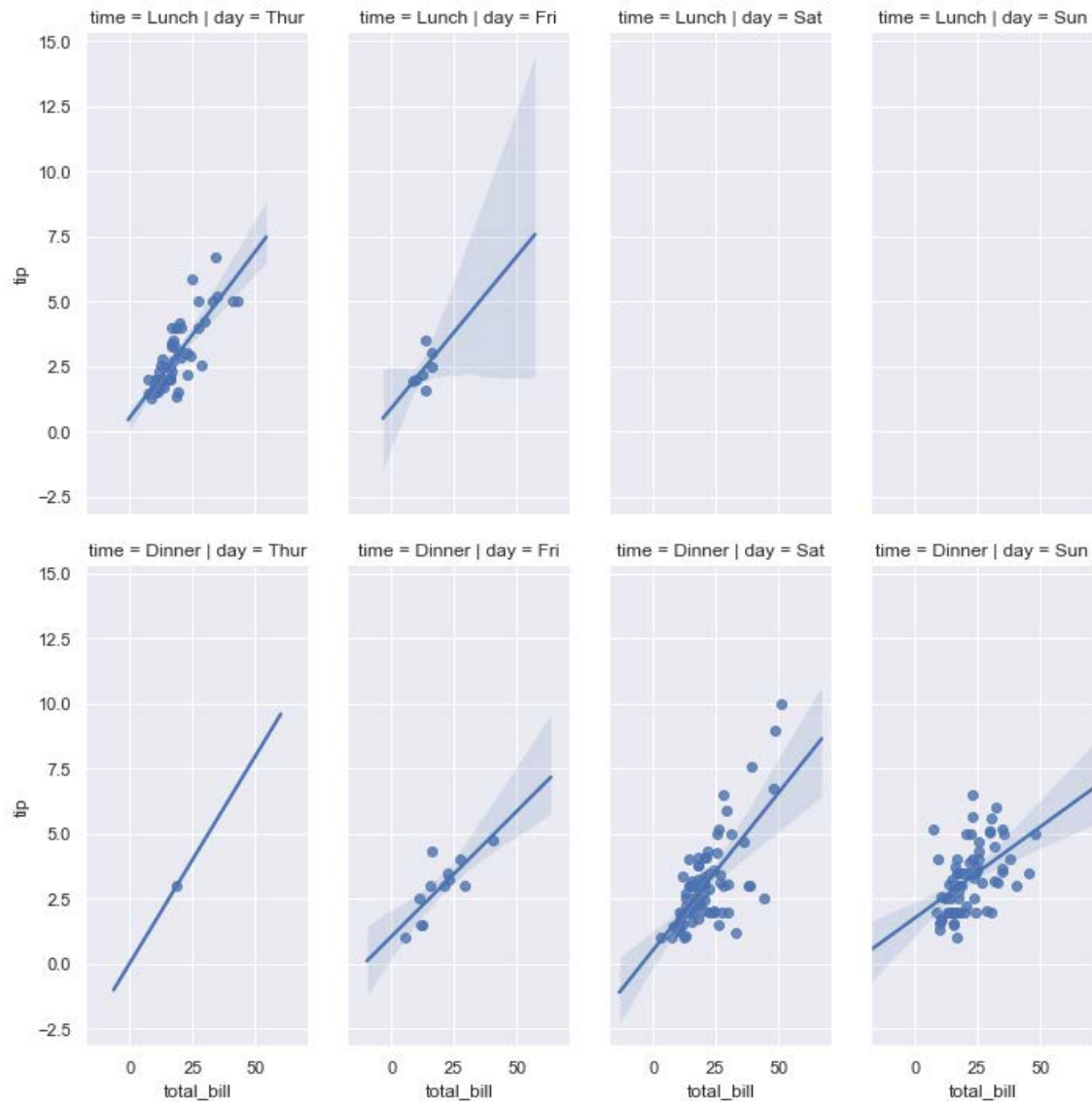
<https://seaborn.pydata.org/tutorial/regression.html>

```
sns.lmplot(x="total_bill", y="tip", hue="smoker",  
           col="time", row="sex", data=tips);
```

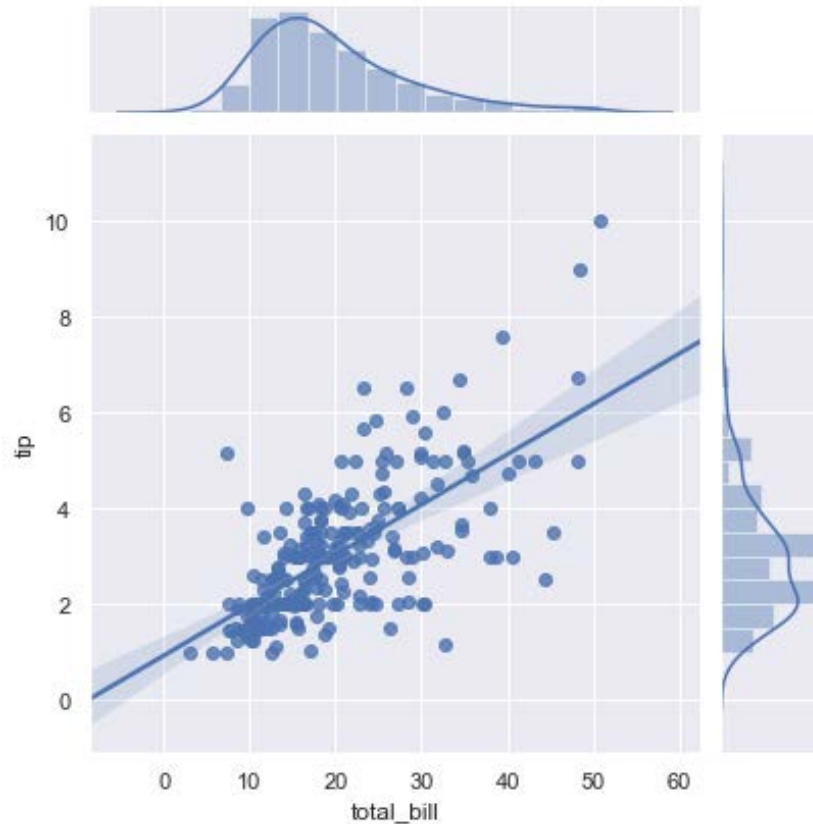


<https://seaborn.pydata.org/tutorial/regression.html>

```
sns.set(color_codes=True);  
sns.lmplot(x="total_bill", y="tip", col="day", row="time", data=tips,  
           aspect=.5);
```



Plotting a regression in other contexts

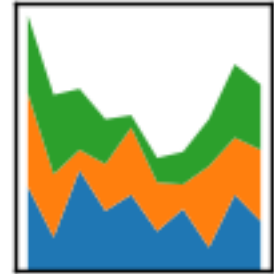
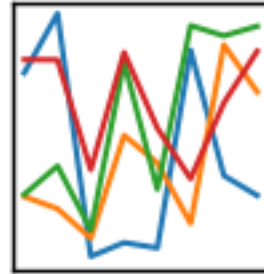
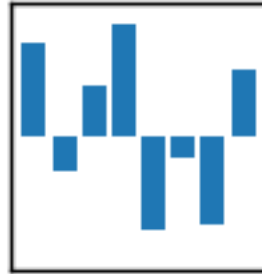


<https://seaborn.pydata.org/tutorial/regression.html>

<https://seaborn.pydata.org/generated/seaborn.jointplot.html#seaborn.jointplot>

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



```
import numpy as np  
import pandas as pd
```

Create a Series

```
s = pd.Series([1, 3, 5, np.nan, 7, 9])  
s  
  
0    1.0  
1    3.0  
2    5.0  
3    NaN  
4    7.0  
5    9.0  
dtype: float64
```

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

Create a DataFrame

```
dates = pd.date_range('20190901', periods=6)
df = pd.DataFrame(np.random.randn(6, 5), index=dates, columns=list('ABCDE'))
df
```

	A	B	C	D	E
2019-09-01	-1.053425	1.464377	-0.273853	-0.025258	-1.338681
2019-09-02	0.047744	-1.334018	-0.726689	1.322917	-0.425243
2019-09-03	-3.069065	0.081144	1.374411	1.903245	0.328674
2019-09-04	-1.436338	0.584854	0.630831	0.454311	-1.795015
2019-09-05	-0.695772	-0.282153	-0.653798	1.979850	-0.736357
2019-09-06	0.526961	-0.528157	-1.934245	0.125042	-0.753182

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

Create a **DataFrame** by passing a dictionary of objects that can be converted to series-like

```
df2 = pd.DataFrame({'A': 1.,
                    'B': pd.Timestamp('20190901'),
                    'C': pd.Series(1, index=list(range(4)), dtype='float32'),
                    'D': np.array([3] * 4, dtype='int32'),
                    'E': pd.Categorical(["test", "train", "test", "train"]),
                    'F': 'foo'})
df2
```

	A	B	C	D	E	F
0	1.0	2019-09-01	1.0	3	test	foo
1	1.0	2019-09-01	1.0	3	train	foo
2	1.0	2019-09-01	1.0	3	test	foo
3	1.0	2019-09-01	1.0	3	train	foo

```
df2.dtypes
```

```
A          float64
B    datetime64[ns]
C          float32
D          int32
E          category
F           object
dtype: object
```

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

View the top and bottom rows

```
df.head(3)
```

	A	B	C	D	E
2019-09-01	-1.053425	1.464377	-0.273853	-0.025258	-1.338681
2019-09-02	0.047744	-1.334018	-0.726689	1.322917	-0.425243
2019-09-03	-3.069065	0.081144	1.374411	1.903245	0.328674

```
df.tail(2)
```

	A	B	C	D	E
2019-09-05	-0.695772	-0.282153	-0.653798	1.979850	-0.736357
2019-09-06	0.526961	-0.528157	-1.934245	0.125042	-0.753182

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

Display index and columns

```
df.index
```

```
DatetimeIndex(['2019-09-01', '2019-09-02', '2019-09-03', '2019-09-04',  
               '2019-09-05', '2019-09-06'],  
              dtype='datetime64[ns]', freq='D')
```

```
df.columns
```

```
Index(['A', 'B', 'C', 'D', 'E'], dtype='object')
```

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

Convert to NumPy Arrays (No index and column label)

```
df.to_numpy()
```

```
array([[ -1.05342528,  1.46437732, -0.27385301, -0.02525847, -1.33868145],  
       [  0.04774417, -1.33401754, -0.72668916,  1.32291669, -0.42524277],  
       [-3.06906528,  0.08114368,  1.37441074,  1.9032455 ,  0.3286742 ],  
       [-1.43633805,  0.58485406,  0.63083128,  0.45431118, -1.79501536],  
       [-0.6957716 , -0.28215313, -0.65379802,  1.97985008, -0.73635676],  
       [  0.52696143, -0.52815725, -1.93424535,  0.12504231, -0.7531819 ]])
```

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

Quick statistic summary of data

```
df.describe()
```

	A	B	C	D	E
count	6.000000	6.000000	6.000000	6.000000	6.000000
mean	-0.946649	-0.002325	-0.263891	0.960018	-0.786634
std	1.263260	0.962411	1.152472	0.892763	0.734813
min	-3.069065	-1.334018	-1.934245	-0.025258	-1.795015
25%	-1.340610	-0.466656	-0.708466	0.207360	-1.192307
50%	-0.874598	-0.100505	-0.463826	0.888614	-0.744769
75%	-0.138135	0.458926	0.404660	1.758163	-0.503021
max	0.526961	1.464377	1.374411	1.979850	0.328674

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

Transpose Data

df.T

	2019-09-01 00:00:00	2019-09-02 00:00:00	2019-09-03 00:00:00	2019-09-04 00:00:00	2019-09-05 00:00:00	2019-09-06 00:00:00
A	-1.053425	0.047744	-3.069065	-1.436338	-0.695772	0.526961
B	1.464377	-1.334018	0.081144	0.584854	-0.282153	-0.528157
C	-0.273853	-0.726689	1.374411	0.630831	-0.653798	-1.934245
D	-0.025258	1.322917	1.903245	0.454311	1.979850	0.125042
E	-1.338681	-0.425243	0.328674	-1.795015	-0.736357	-0.753182

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

Sorting by Values

```
df.sort_values(by='C')
```

	A	B	C	D	E
2019-09-06	0.526961	-0.528157	-1.934245	0.125042	-0.753182
2019-09-02	0.047744	-1.334018	-0.726689	1.322917	-0.425243
2019-09-05	-0.695772	-0.282153	-0.653798	1.979850	-0.736357
2019-09-01	-1.053425	1.464377	-0.273853	-0.025258	-1.338681
2019-09-04	-1.436338	0.584854	0.630831	0.454311	-1.795015
2019-09-03	-3.069065	0.081144	1.374411	1.903245	0.328674

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

Getting

```
df['B']
```

```
2019-09-01    1.464377
2019-09-02   -1.334018
2019-09-03    0.081144
2019-09-04    0.584854
2019-09-05   -0.282153
2019-09-06   -0.528157
Freq: D, Name: B, dtype: float64
```

```
df[0:3]
```

	A	B	C	D	E
2019-09-01	-1.053425	1.464377	-0.273853	-0.025258	-1.338681
2019-09-02	0.047744	-1.334018	-0.726689	1.322917	-0.425243
2019-09-03	-3.069065	0.081144	1.374411	1.903245	0.328674

```
df['20190901':'20190904']
```

	A	B	C	D	E
2019-09-01	-1.053425	1.464377	-0.273853	-0.025258	-1.338681
2019-09-02	0.047744	-1.334018	-0.726689	1.322917	-0.425243
2019-09-03	-3.069065	0.081144	1.374411	1.903245	0.328674
2019-09-04	-1.436338	0.584854	0.630831	0.454311	-1.795015

Selection by Label

```
df.loc['20190902':'20190904', ['A', 'B']]
```

	A	B
2019-09-02	0.047744	-1.334018
2019-09-03	-3.069065	0.081144
2019-09-04	-1.436338	0.584854

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

Selection by Position

```
df.iloc[[1, 2, 4], [0, 2]]
```

	A	C
2019-09-02	0.047744	-0.726689
2019-09-03	-3.069065	1.374411
2019-09-05	-0.695772	-0.653798

```
df.iloc[1:3, :]
```

	A	B	C	D	E
2019-09-02	0.047744	-1.334018	-0.726689	1.322917	-0.425243
2019-09-03	-3.069065	0.081144	1.374411	1.903245	0.328674

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

Boolean Indexing

```
df[df.B > 0.5]
```

	A	B	C	D	E
2019-09-01	-1.053425	1.464377	-0.273853	-0.025258	-1.338681
2019-09-04	-1.436338	0.584854	0.630831	0.454311	-1.795015

```
df[df > 0.5]
```

	A	B	C	D	E
2019-09-01	NaN	1.464377	NaN	NaN	NaN
2019-09-02	NaN	NaN	NaN	1.322917	NaN
2019-09-03	NaN	NaN	1.374411	1.903245	NaN
2019-09-04	NaN	0.584854	0.630831	NaN	NaN
2019-09-05	NaN	NaN	NaN	1.979850	NaN
2019-09-06	0.526961	NaN	NaN	NaN	NaN

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

Setting

```
df2 = df.copy()  
df2[df2 > 0] = -df2  
df2
```

	A	B	C	D	E
2019-09-01	-1.053425	-1.464377	-0.273853	-0.025258	-1.338681
2019-09-02	-0.047744	-1.334018	-0.726689	-1.322917	-0.425243
2019-09-03	-3.069065	-0.081144	-1.374411	-1.903245	-0.328674
2019-09-04	-1.436338	-0.584854	-0.630831	-0.454311	-1.795015
2019-09-05	-0.695772	-0.282153	-0.653798	-1.979850	-0.736357
2019-09-06	-0.526961	-0.528157	-1.934245	-0.125042	-0.753182

https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html

Writing to and Reading from csv file

```
# writing to a CSV file
```

```
df.to_csv('week2.csv')
```

```
# reading from a csv file
```

```
df = pd.read_csv('week2.csv')
```

Writing to and Reading from excel file

```
# writing to excel file
```

```
df.to_excel('week2.xlsx',  
sheet_name='Sheet1')
```

```
# reading from excel file
```

```
df = pd.read_excel('week2.xlsx', 'Sheet1',  
index_col=None, na_values=['NA'])
```

Folium Library for Visualizing Geospatial Data

```
!conda install -c conda-forge folium --yes
```

```
import folium
```

```
print('Folium installed and imported!')
```

```
# define the world map
```

```
world_map = folium.Map()
```

```
# display world map
```

```
world_map
```

<https://labs.cognitiveclass.ai/tools/jupyterlab/lab/tree/labs/DV0101EN/DV0101EN-3-5-1-Generating-Maps-in-Python-py-v2.0.ipynb>

```
# create a Stamen Toner map of the world centered around HK
world_map = folium.Map(location=[22.3700556,114.1535941],
                        zoom_start=8, tiles='Stamen Toner')
```

```
# display map
world_map
```



```
# create a Stamen Terrain map of the world centered around HK
```

```
world_map = folium.Map(location=[22.3700556,114.1535941],  
zoom_start=10, tiles='Stamen Terrain')
```

```
# display map  
world_map
```



Add a Marker

```
# create a Stamen Toner map of the world centered around HK
```

```
hk_map = folium.Map(location=[22.3700556,114.1535941], zoom_start=10, tiles='Stamen Terrain')
```

```
## Add a marker to Kowloon Tong
```

```
# Create a feature group
```

```
KowloonTong = folium.map.FeatureGroup()
```

```
# style the feature group
```

```
KowloonTong.add_child(  
    folium.vector_layers.CircleMarker(  
        [22.33445,114.17803],radius = 5,  
        color = "yellow", fill_color="yellow"))
```

```
# add the feature group to the map
```

```
hk_map.add_child(KowloonTong)
```

```
# label the marker
```

```
folium.Marker([22.33445,114.17803],  
    popup='Kowloon Tong').add_to(hk_map)
```

```
# display map
```

```
hk_map
```

Choropleth Maps

```
import pandas as pd
```

```
url = 'https://raw.githubusercontent.com/python-visualization/folium/master/examples/data'
```

```
state_geo = f'{url}/us-states.json'
```

```
state_unemployment = f'{url}/US_Unemployment_Oct2012.csv'
```

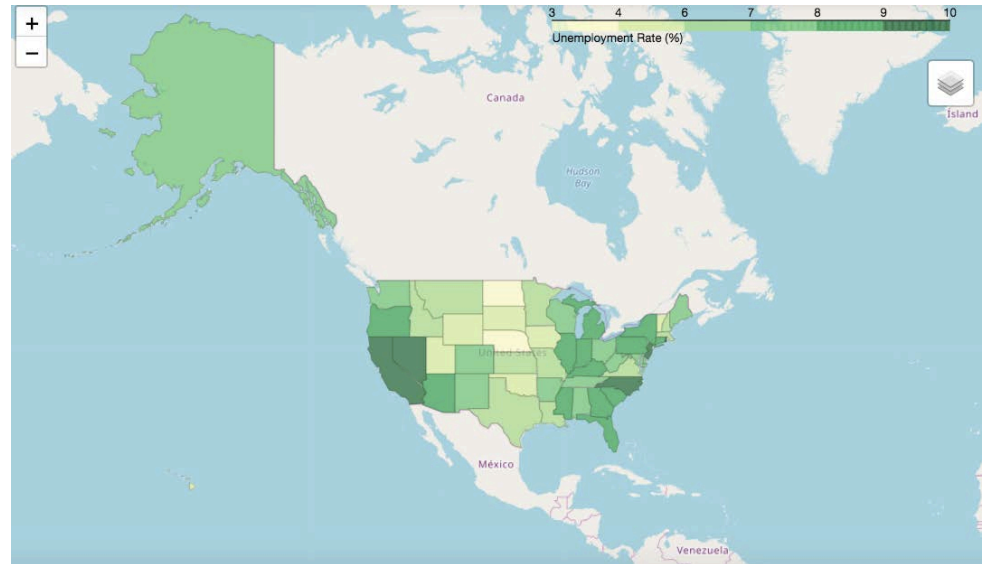
```
state_data = pd.read_csv(state_unemployment)
```

```
m = folium.Map(location=[48, -102], zoom_start=3)
```

```
folium.Choropleth(  
    geo_data=state_geo,  
    name='choropleth',  
    data=state_data,  
    columns=['State', 'Unemployment'],  
    key_on='feature.id',  
    fill_color='YlGn',  
    fill_opacity=0.7,  
    line_opacity=0.2,  
    legend_name='Unemployment Rate (%)'  
).add_to(m)
```

```
folium.LayerControl().add_to(m)
```

```
m
```



<https://python-visualization.github.io/folium/quickstart.html>

Word Cloud

https://github.com/amueller/word_cloud

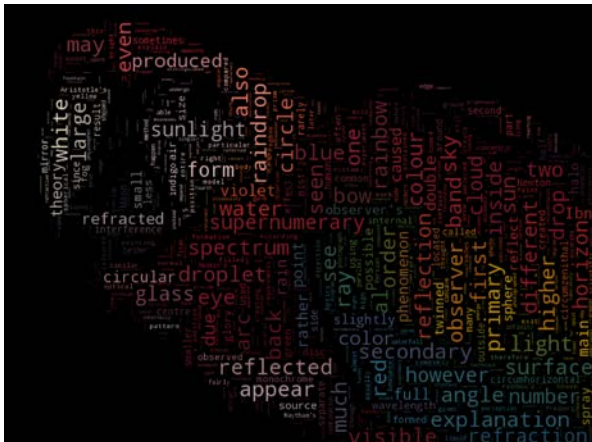
```
# install wordcloud
```

```
!conda install -c conda-forge wordcloud --yes
```

```
# import package and its set of stopwords
```

```
from wordcloud import WordCloud, STOPWORDS
```

```
print ('Wordcloud is installed and imported!')
```



```
import requests

# download file
url='http://www.gutenberg.org/cache/epub/10/pg10.txt'
myfile = requests.get(url)
open('pg10.txt', 'wb').write(myfile.content)
# open the file and read it into a variable
BibleText = open('pg10.txt', 'r').read()

print ('File downloaded and saved!')

stopwords = set(STOPWORDS)

# instantiate a word cloud object
Bible_wc = WordCloud(
    background_color='white',
    max_words=2000,
    stopwords=stopwords
)

# generate the word cloud
Bible_wc.generate(BibleText)
```

```
plt.show()
```

