Computer Science

CSC380: Principles of Data Science

Data Analysis, Collection, and Visualization 1

Chicheng Zhang

Announcements

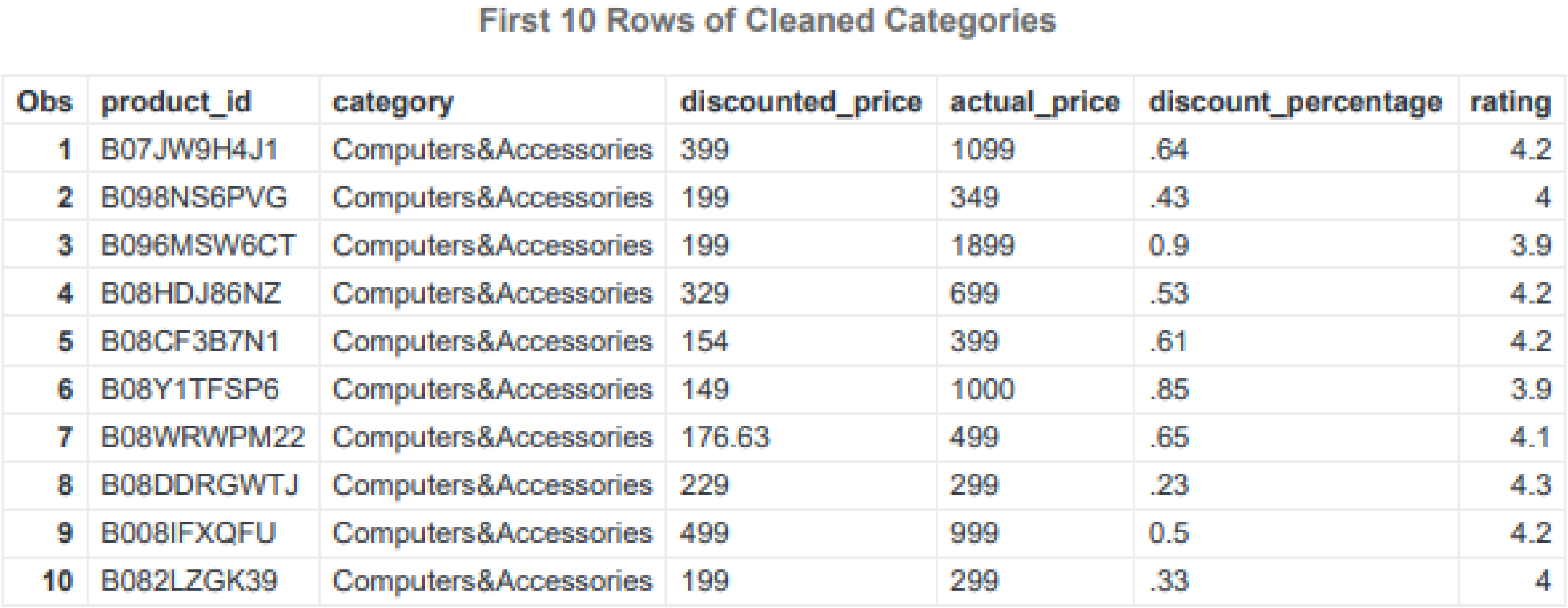
- Office hour times are decided
  - Our ASO is helping us finding TA offices
  - This week, we will do Zoom office hours
  - See D2L for time & place

- I created a form for self-reporting class participation
  - You can report all three activities (OH, in-class discussion, Piazza) there

- HW1 will be released today

Data is everywhere

Amazon Sales Dataset



First 10 Rows of Cleaned Categories

Obs	product_id	category	discounted_price	actual_price	discount_percentage	rating
1	B07JW9H4J1	Computers&Accessories	399	1099	.64	4.2
2	B098NS6PVG	Computers&Accessories	199	349	.43	4
3	B096MSW6CT	Computers&Accessories	199	1899	0.9	3.9
4	B08HDJ86NZ	Computers&Accessories	329	699	.53	4.2
5	B08CF3B7N1	Computers&Accessories	154	399	.61	4.2
6	B08Y1TFSP6	Computers&Accessories	149	1000	.85	3.9
7	B08WRWPM22	Computers&Accessories	176.63	499	.65	4.1
8	B08DDRGWTJ	Computers&Accessories	229	299	.23	4.3
9	B008IFXQFU	Computers&Accessories	499	999	0.5	4.2
10	B082LZGK39	Computers&Accessories	199	299	.33	4

How can we:

- get a sense of how users like the products?
- filter the highly-rated products and look at their common characteristics?

Today’s plan

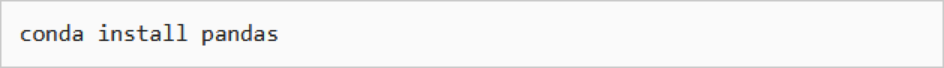
- Basic data processing using Pandas
  - Create dataframe
  - Access dataframe
  - Convert dataframe to other objects
- Descriptive statistics using Pandas
- Basic data visualization

Pandas

Pandas

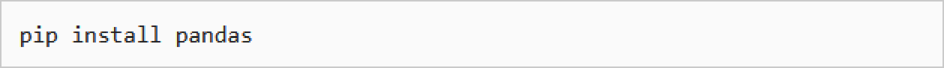
Open source library for data handling and manipulation in high-performance environments.

Installation If you are using Anaconda package manager,



```
conda install pandas
```

Or if you are using PyPi (pip) package manager,

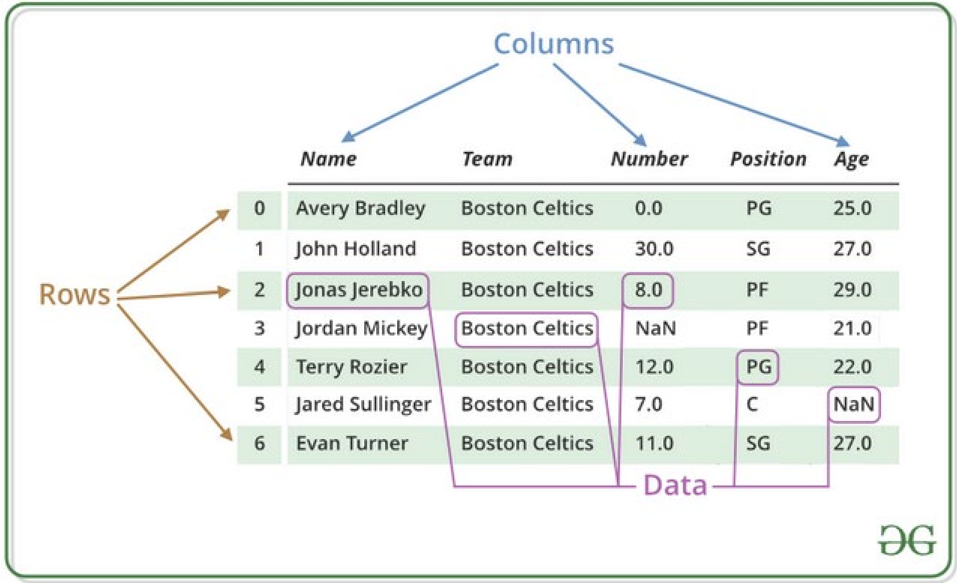


```
pip install pandas
```

See Pandas documentation for more detailed instructions
https://pandas.pydata.org/docs/getting_started/install.html

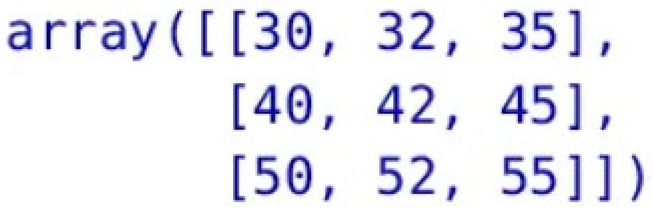
DataFrame

Primary data structure : Essentially a table



	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

Q: how is it different from an array?

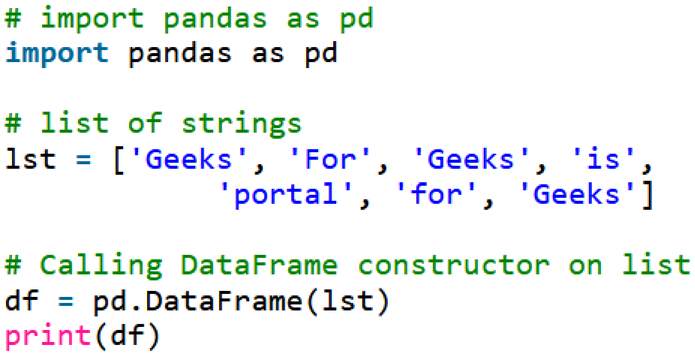


```
array([[30, 32, 35],
       [40, 42, 45],
       [50, 52, 55]])
```

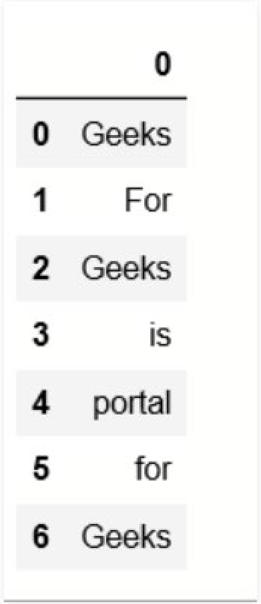
- Dataframes’ elements’ data types can be mixed; an array usually store elements of same type
- Dataframes’ rows and are labeled with indices; array indices are usually integers

DataFrame Example

Create and print an entire DataFrame



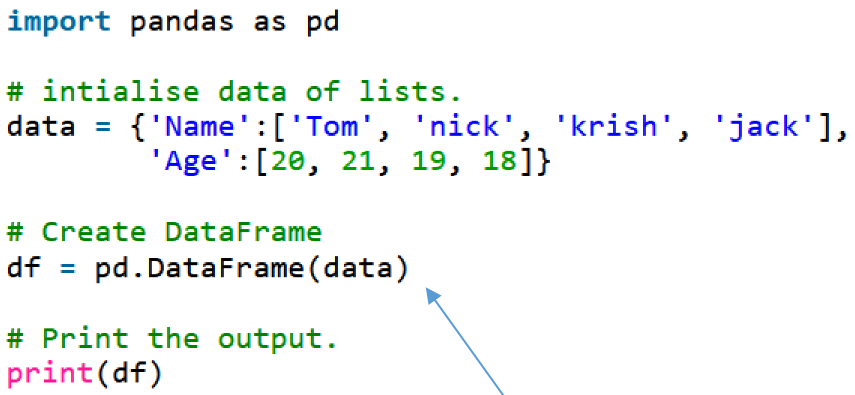
```python
# import pandas as pd
import pandas as pd

# list of strings
lst = ['Geeks', 'For', 'Geeks', 'is',
            'portal', 'for', 'Geeks']

# Calling DataFrame constructor on list
df = pd.DataFrame(lst)
print(df)
```



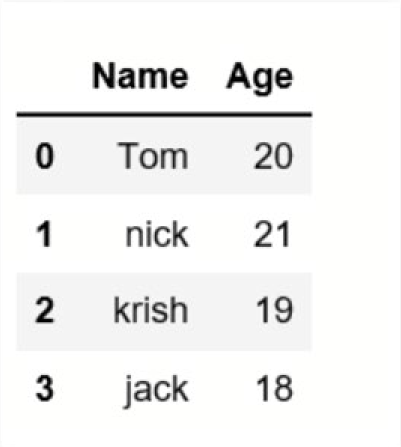
	0
0	Geeks
1	For
2	Geeks
3	is
4	portal
5	for
6	Geeks

DataFrame Example

Can create *named columns* using dictionary



```
import pandas as pd

# intialise data of lists.
data = {'Name':['Tom', 'nick', 'krish', 'jack'],
        'Age':[20, 21, 19, 18]}

# Create DataFrame
df = pd.DataFrame(data)

# Print the output.
print(df)
```

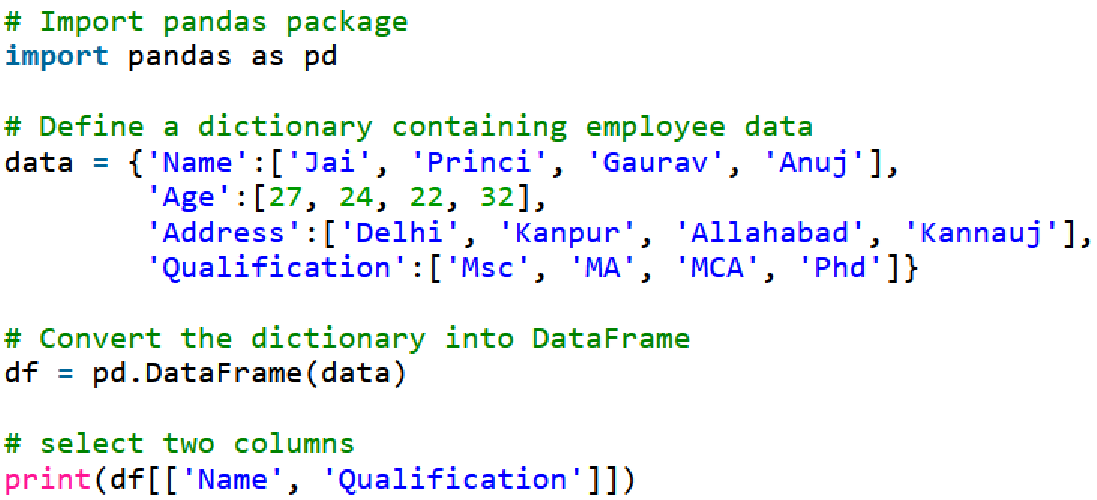


	Name	Age
0	Tom	20
1	nick	21
2	krish	19
3	jack	18

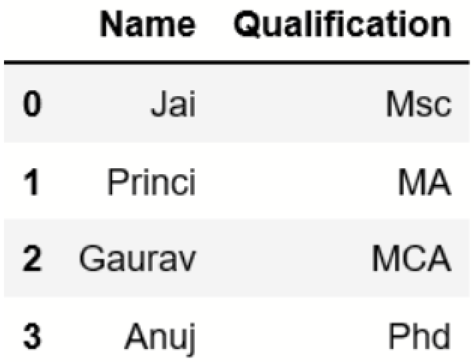
all data must have the same length

DataFrame : Selecting Columns

Select columns to print by name



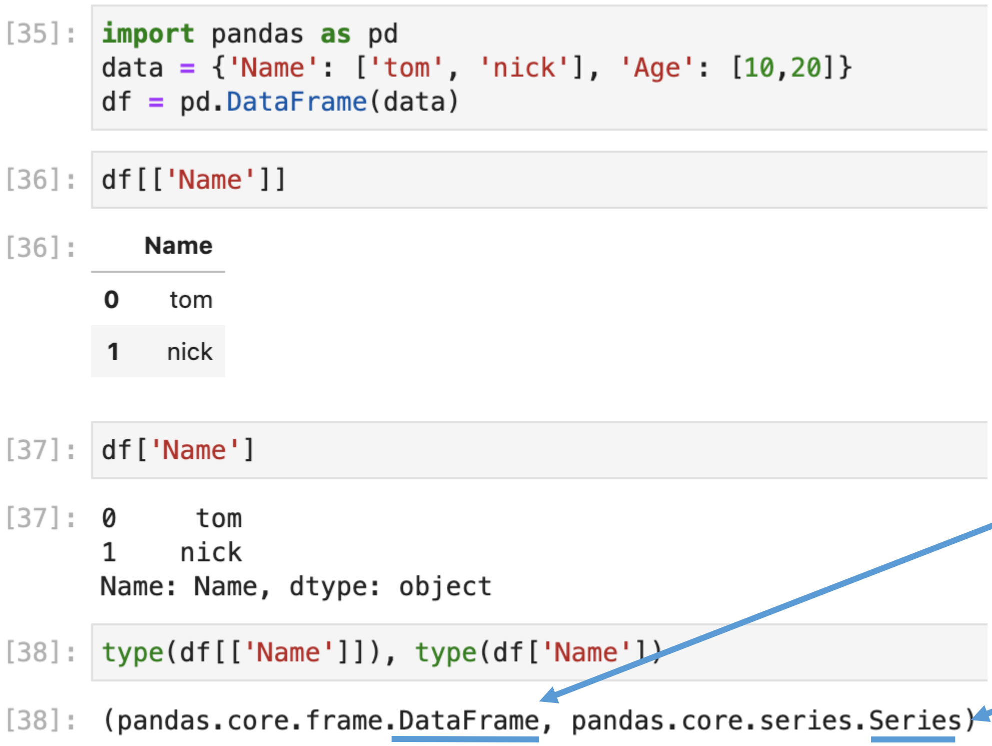
```
# Import pandas package
import pandas as pd

# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# select two columns
print(df[['Name', 'Qualification']])
```



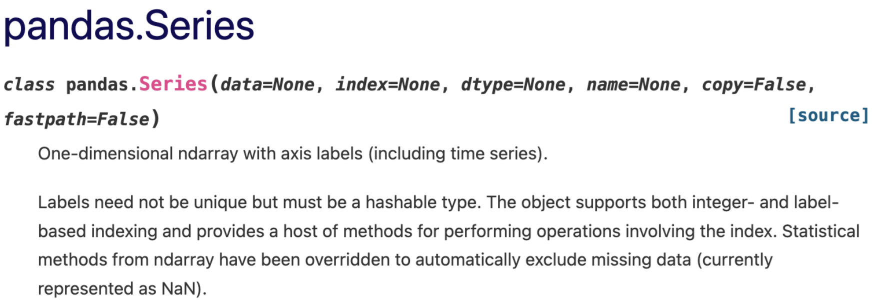
	Name	Qualification
0	Jai	Msc
1	Princi	MA
2	Gaurav	MCA
3	Anuj	Phd

access columns by name, not the column index!

DataFrame : Selecting Columns



```
[35]: import pandas as pd
      data = {'Name': ['tom', 'nick'], 'Age': [10,20]}
      df = pd.DataFrame(data)

[36]: df[['Name']]

[36]:      Name
       0   tom
       1   nick

[37]: df['Name']

[37]: 0     tom
      1    nick
      Name: Name, dtype: object

[38]: type(df[['Name']]), type(df['Name'])

[38]: (pandas.core.frame.DataFrame, pandas.core.series.Series)
```



pandas.Series

class pandas.**Series**(*data=None*, *index=None*, *dtype=None*, *name=None*, *copy=False*, *fastpath=False*) [source]

One-dimensional ndarray with axis labels (including time series).

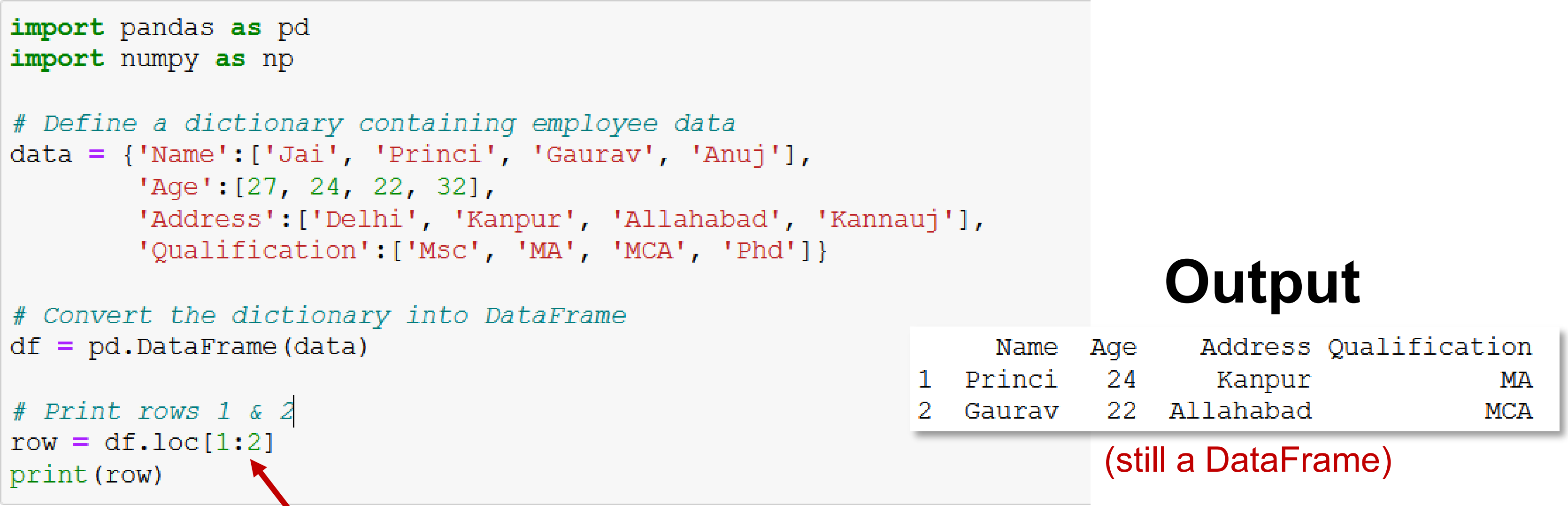
Labels need not be unique but must be a hashable type. The object supports both integer- and label-based indexing and provides a host of methods for performing operations involving the index. Statistical methods from ndarray have been overridden to automatically exclude missing data (currently represented as NaN).

still a DataFrame

essentially, a DataFrame's single row or column

DataFrame : Selecting Rows

Use df.loc to access certain rows



```
import pandas as pd
import numpy as np

# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# Print rows 1 & 2
row = df.loc[1:2]
print(row)
```

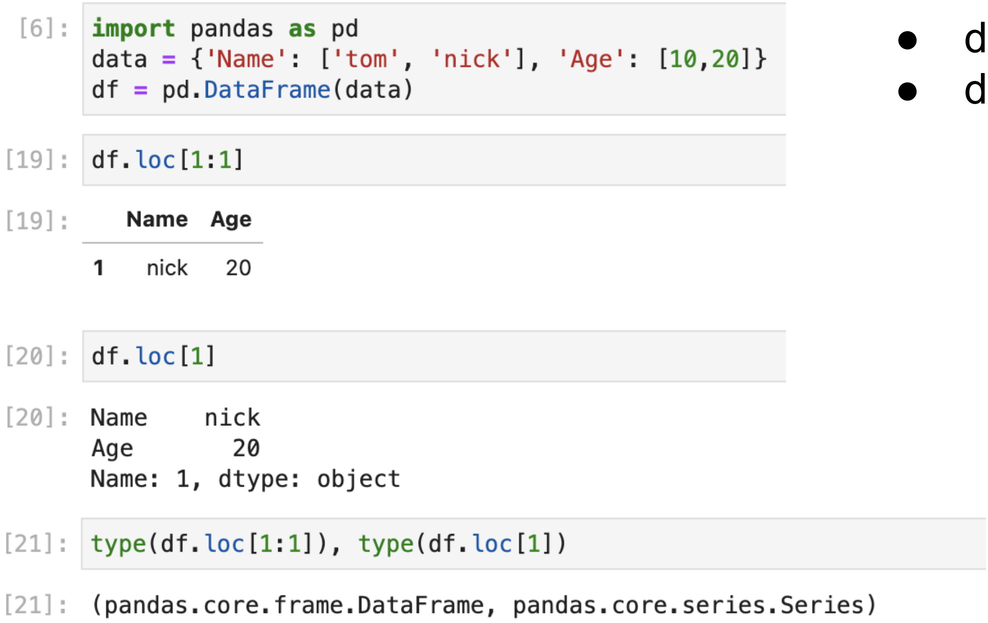
Output

```
     Name  Age    Address Qualification
1  Princi   24     Kanpur            MA
2  Gaurav   22  Allahabad           MCA
```

(still a DataFrame)

1:2 includes 2! This is different from Python array indexing

DataFrame : Selecting Rows



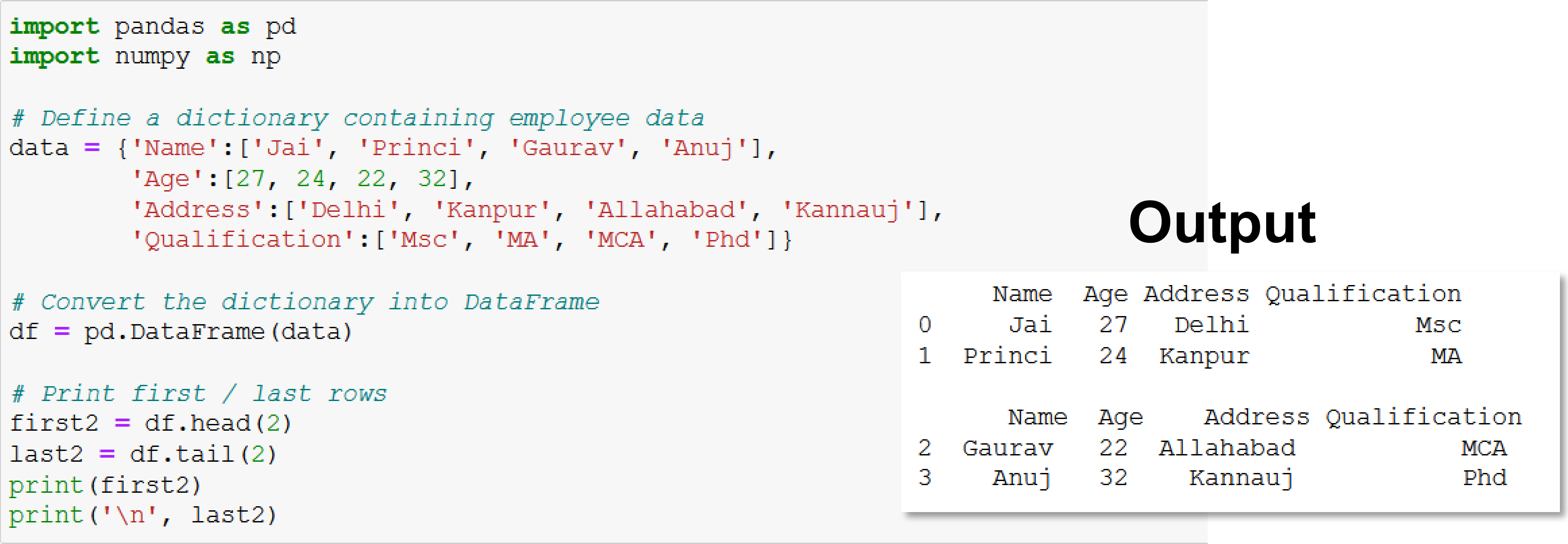
```
[6]:  import pandas as pd
      data = {'Name': ['tom', 'nick'], 'Age': [10,20]}
      df = pd.DataFrame(data)

[19]: df.loc[1:1]

[19]:     Name  Age
      1   nick   20

[20]: df.loc[1]

[20]: Name    nick
      Age       20
      Name: 1, dtype: object

[21]: type(df.loc[1:1]), type(df.loc[1])

[21]: (pandas.core.frame.DataFrame, pandas.core.series.Series)
```

- df.loc[1:1] is DataFrame object
- df.loc[1] is a series

DataFrame : Selecting Rows

`head()` and `tail()` select rows from beginning / end

handy when we would like to get a sense of what a big table looks like



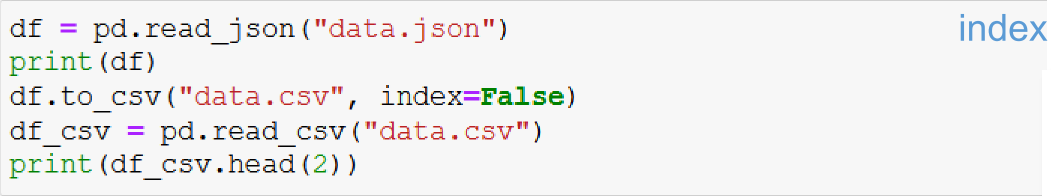
```
import pandas as pd
import numpy as np

# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)

# Print first / last rows
first2 = df.head(2)
last2 = df.tail(2)
print(first2)
print('\n', last2)
```

Output

```
     Name  Age Address Qualification
0     Jai   27   Delhi           Msc
1  Princi   24  Kanpur            MA

     Name  Age    Address Qualification
2  Gaurav   22  Allahabad           MCA
3    Anuj   32    Kannauj           Phd
```

Reading Data from Files

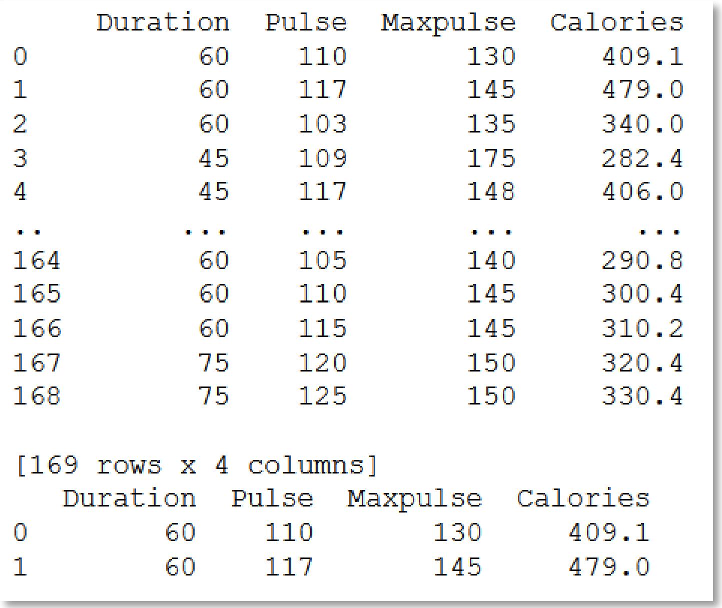
Easy reading / writing of standard formats



```
df = pd.read_json("data.json")
print(df)
df.to_csv("data.csv", index=False)
df_csv = pd.read_csv("data.csv")
print(df_csv.head(2))
```

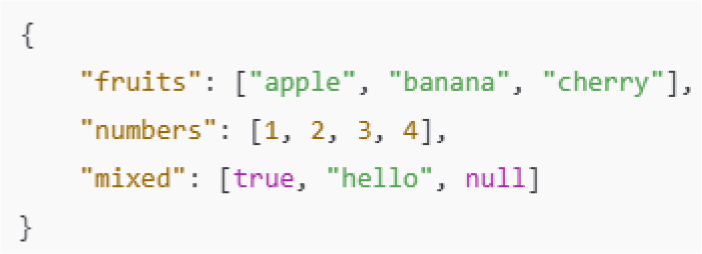
index ↓

Output



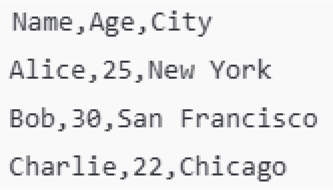
```
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
..        ...    ...       ...       ...
164        60    105       140     290.8
165        60    110       145     300.4
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4

[169 rows x 4 columns]
   Duration  Pulse  Maxpulse  Calories
0        60    110       130     409.1
1        60    117       145     479.0
```

Json format: e.g. X(twitter) API



```
{
    "fruits": ["apple", "banana", "cherry"],
    "numbers": [1, 2, 3, 4],
    "mixed": [true, "hello", null]
}
```

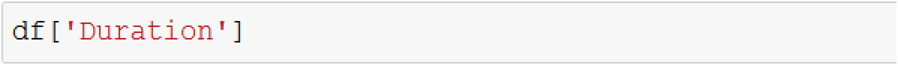
CSV format (comma separated values)



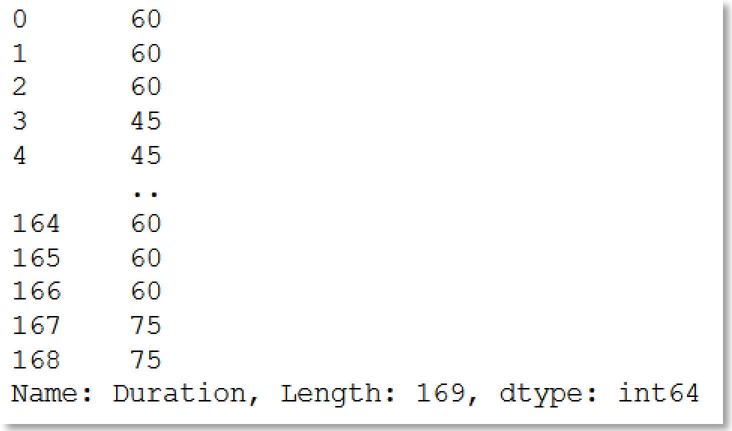
```
Name,Age,City
Alice,25,New York
Bob,30,San Francisco
Charlie,22,Chicago
```

Data Type Conversions

Working with DataFrames outside of Pandas can be tricky



```
df['Duration']
```

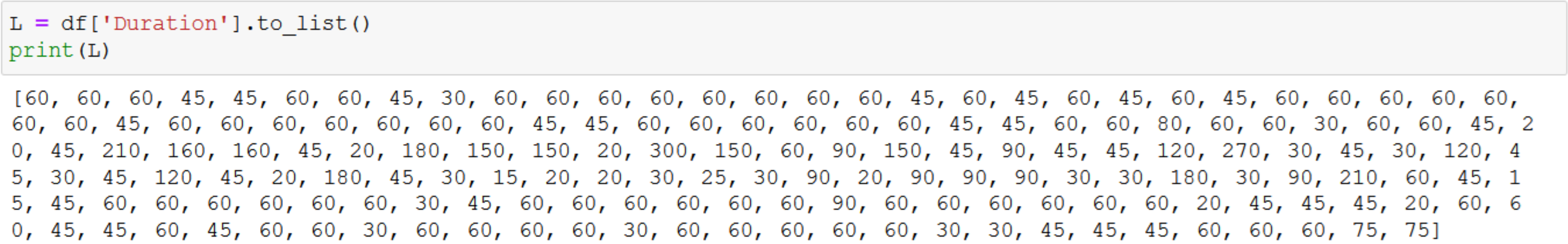


```
0       60
1       60
2       60
3       45
4       45
        ..
164     60
165     60
166     60
167     75
168     75
Name: Duration, Length: 169, dtype: int64
```

Q: is this a DataFrame object or Series object?

A: a Series object

We can easily convert a Series to built-in types, e.g., a list.

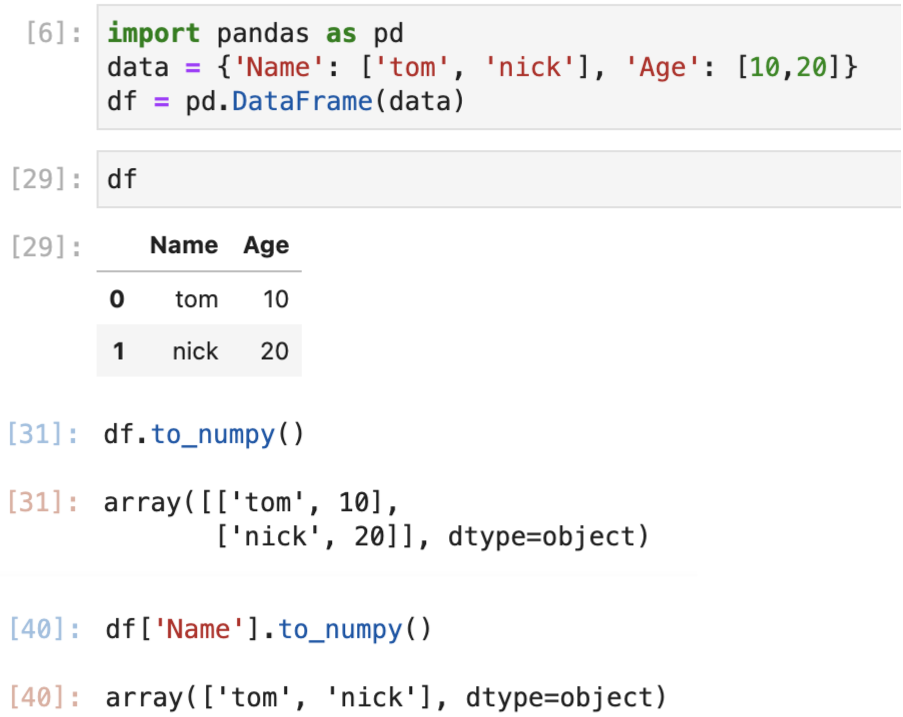


```
L = df['Duration'].to_list()
print(L)
```

```
[60, 60, 60, 45, 45, 60, 60, 45, 30, 60, 60, 60, 60, 60, 60, 60, 60, 45, 60, 45, 60, 45, 60, 45, 60, 60, 60, 60, 60,
60, 60, 45, 60, 60, 60, 60, 60, 60, 60, 45, 45, 60, 60, 60, 60, 60, 60, 45, 45, 60, 60, 80, 60, 60, 30, 60, 60, 45, 2
0, 45, 210, 160, 160, 45, 20, 180, 150, 150, 20, 300, 150, 60, 90, 150, 45, 90, 45, 45, 120, 270, 30, 45, 30, 120, 4
5, 30, 45, 120, 45, 20, 180, 45, 30, 15, 20, 20, 30, 25, 30, 90, 20, 90, 90, 90, 30, 30, 180, 30, 90, 210, 60, 45, 1
5, 45, 60, 60, 60, 60, 60, 60, 30, 45, 60, 60, 60, 60, 60, 60, 90, 60, 60, 60, 60, 60, 60, 20, 45, 45, 45, 20, 60, 6
0, 45, 45, 60, 45, 60, 60, 30, 60, 60, 60, 60, 30, 60, 60, 60, 60, 60, 30, 30, 45, 45, 45, 60, 60, 60, 75, 75]
```

Data Type Conversions

Or, to a numpy array



```
[6]: import pandas as pd
     data = {'Name': ['tom', 'nick'], 'Age': [10,20]}
     df = pd.DataFrame(data)

[29]: df

[29]:    Name  Age
      0   tom   10
      1  nick   20

[31]: df.to_numpy()

[31]: array([['tom', 10],
             ['nick', 20]], dtype=object)

[40]: df['Name'].to_numpy()

[40]: array(['tom', 'nick'], dtype=object)
```

`to_numpy()`: can take `Series` and `DataFrame` objects as input

Numpy: Python library for scientific computing

Descriptive Statistics (using Pandas)

Descriptive Statistics Overview

- Given a data array, oftentimes useful to summarize it using some of its key features
  - **Range**
  - **Histogram**
  - **Mean**
  - **Median**
  - **Mode**

Range

- Difference between highest (maximum) and lowest (minimum) values
- [min, max] is called the *range interval*

Example what is the range of the following dataset?

4, 7, 2, 9, 12

Max: 12

Min: 2

=> Range interval = [2, 12], Range = 12 – 2 = 10

Histogram

Split the range interval into equally-sized bins and report counts in each bin

Example Taking the ages of the presidents of the United States at the time of their inauguration (in total 44 points)

57,61,57,57,58,57,61,54,68,51, .. 47,70

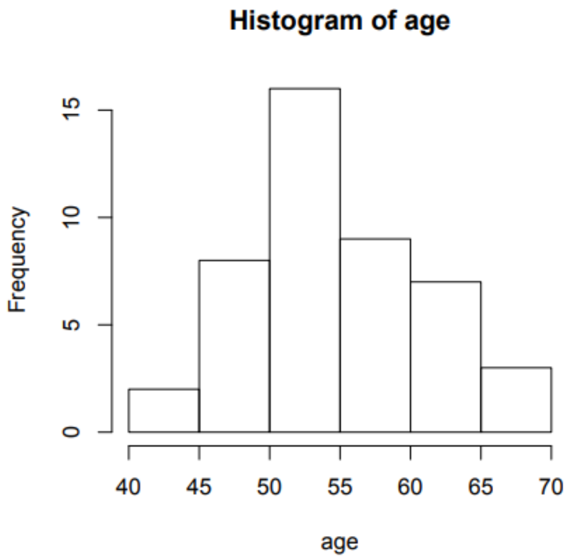
Bins: (40, 45], (45, 50], (50, 55], (55, 60], (60, 65], (65, 70]

Histogram

Counts in different bins

(40, 45]	(45, 50]	(50, 55]	(55, 60]	(60, 65]	(65, 70]
2	8	16	9	7	3

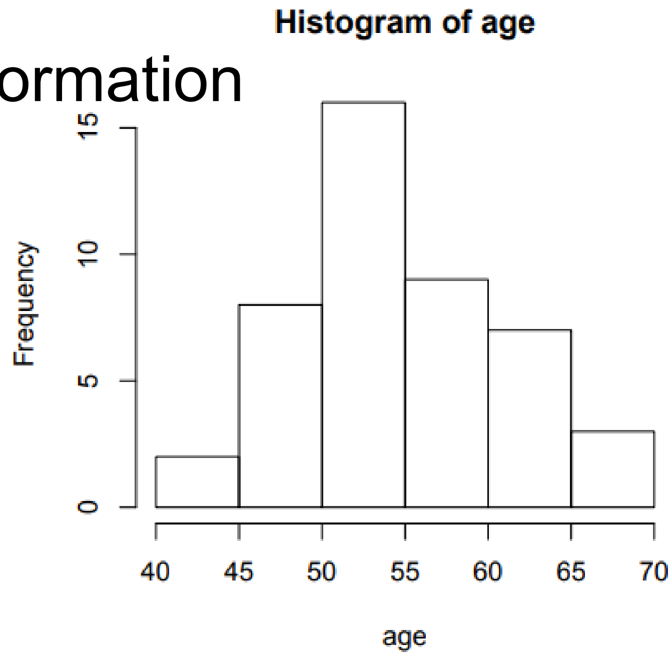
We can also visualize the histogram using a bar plot:



Histogram of age

Histogram

- A histogram reveals a lot of useful information
  - Typical values the data takes
  - The “spread” of data
  - …
- It is a *data visualization* method (more coming up)



Mean

- Average of the data $x_1, \ldots, x_n$
- In formula:

$$\bar{x} = \frac{1}{n}(x_1 + \cdots + x_n) =: \frac{1}{n}\sum_{i=1}^{n} x_i$$

Example heights of 3 students are 1.71, 1.84, 1.64 (m)

their average height $\bar{x} = \frac{1}{3}$(1.71+1.84+1.63) = 1.73 (m)

Median

For data $x_1, x_2, \ldots, x_N$ sort the data,

$$x_{(1)}, x_{(2)}, \ldots, x_{(n)}$$

- Notation $x_{(i)}$ means the i-th *lowest* value, e.g. $x_{(i-1)} \leq x_{(i)} \leq x_{(i+1)}$
- $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$ are called *order statistics* ←

not summary info, but rather a transformation

If n is **odd** then find the middle datapoint,

$$\text{median}(x_1, \ldots, x_n) = x_{((n+1)/2)}$$

If n is **even** then average between both middle datapoints,

$$\text{median}(x_1, \ldots, x_n) = \frac{1}{2}\left(x_{(n/2)} + x_{(n/2+1)}\right)$$

Median

What is the median of the following data?

1, 2, 3, 4, 5, 6, 8, 9 **4.5**

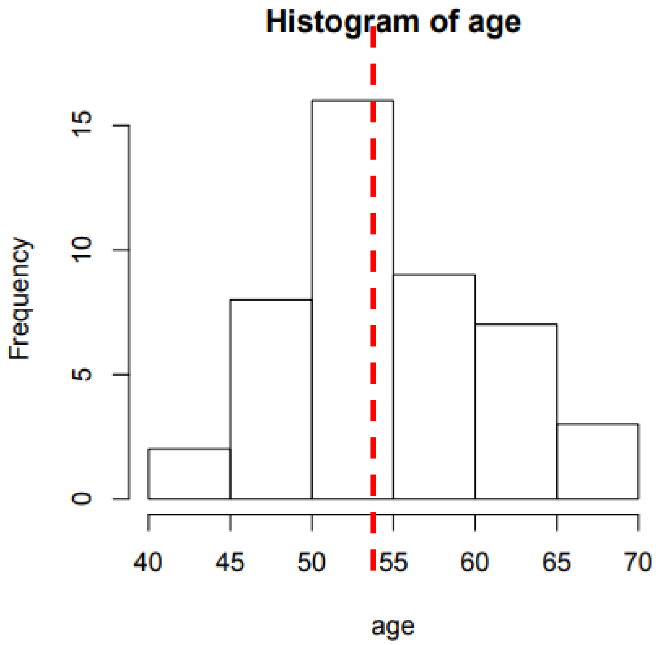
What is the median of the following data?

1, 2, 3, 4, 5, 6, 8, 100 **4.5**

Median is *robust* to outliers

Median

- *Roughly speaking*, median is the point where half of the population is below it and half of the population is above it



Mode

- Value of highest number of appearances

Example what is the mode of the following dataset?

1,1,2,3,7,8,8,8,9

Count of 8: 3

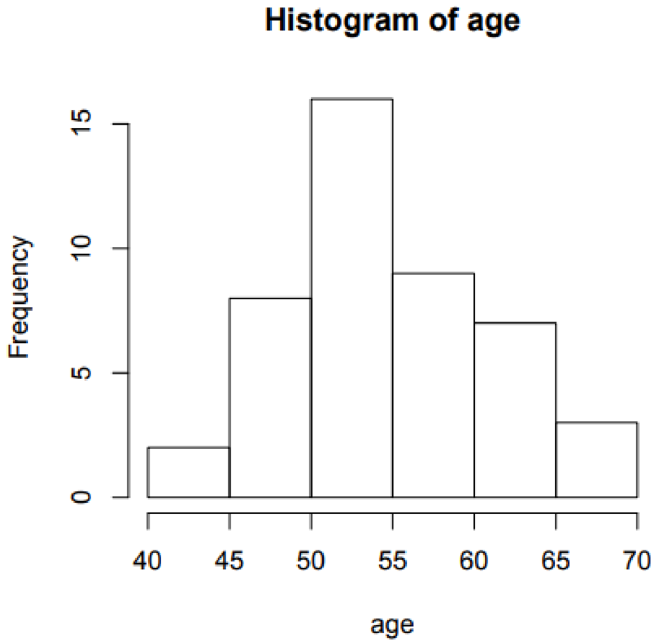
Count of 1: 2

Counts of other numbers: 1

=> Mode = 8

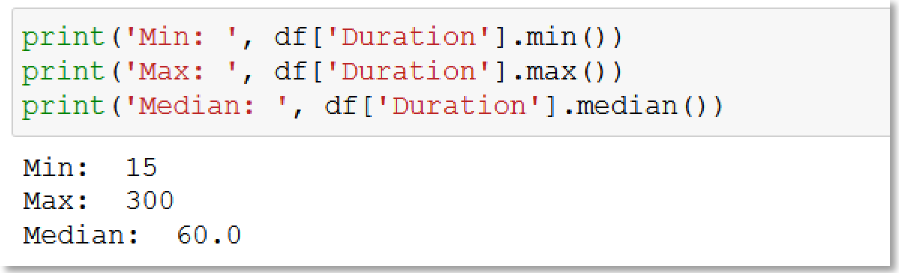
Mode

- *Roughly speaking*, mode is the location of the histogram with the tallest bar



Summary Statistics in Pandas

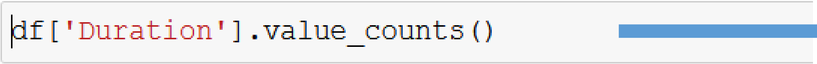
Compute summary statistics on Pandas Series



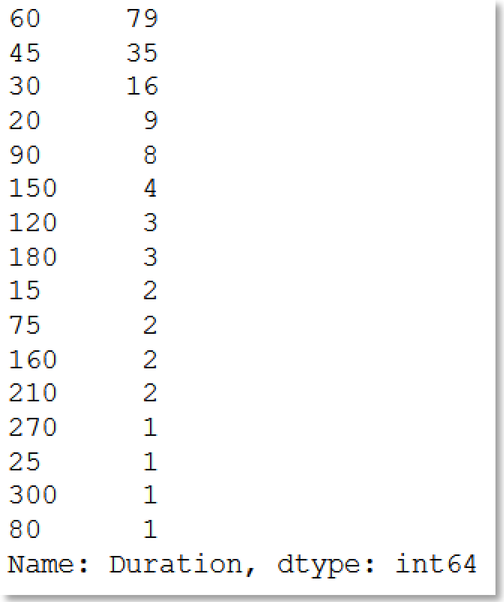
```
print('Min: ', df['Duration'].min())
print('Max: ', df['Duration'].max())
print('Median: ', df['Duration'].median())
```

```
Min:  15
Max:  300
Median:  60.0
```

Can also count occurrences of unique values,



```
df['Duration'].value_counts()
```

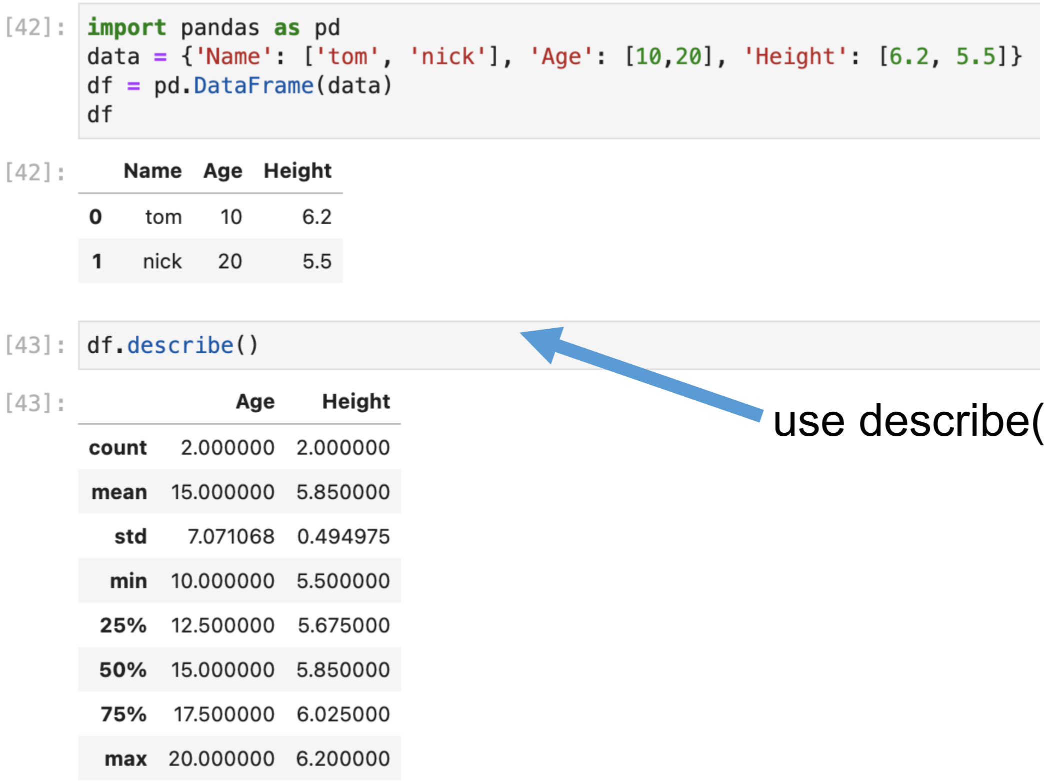


```
60      79
45      35
30      16
20       9
90       8
150      4
120      3
180      3
15       2
75       2
160      2
210      2
270      1
25       1
300      1
80       1
Name: Duration, dtype: int64
```

s = df['Duration'].value_counts()
s[60]=79.

Summary Statistics

Compute summary statistics on each column of Dataframe



```
[42]: import pandas as pd
      data = {'Name': ['tom', 'nick'], 'Age': [10,20], 'Height': [6.2, 5.5]}
      df = pd.DataFrame(data)
      df
```

```
[42]:      Name  Age  Height
       0    tom   10     6.2
       1   nick   20     5.5
```

```
[43]: df.describe()
```

```
[43]:              Age     Height
       count   2.000000  2.000000
        mean  15.000000  5.850000
         std   7.071068  0.494975
         min  10.000000  5.500000
         25%  12.500000  5.675000
         50%  15.000000  5.850000
         75%  17.500000  6.025000
         max  20.000000  6.200000
```

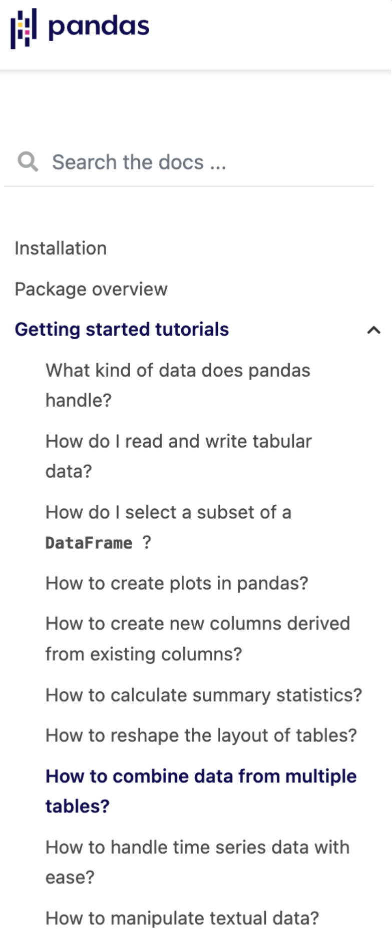
use describe() to get a summary of the data

More on Pandas

Many database operations are available

- You can specify index, which can speed up some operations
- You can do ‘join’
- You can do ‘where’ clause to filter the data
- You can do ‘group by’

More on Pandas



Installation

Package overview

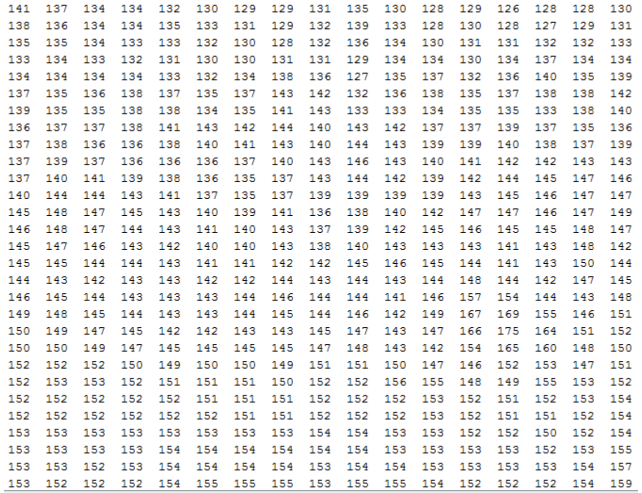
Getting started tutorials

- What kind of data does pandas handle?
- How do I read and write tabular data?
- How do I select a subset of a `DataFrame` ?
- How to create plots in pandas?
- How to create new columns derived from existing columns?
- How to calculate summary statistics?
- How to reshape the layout of tables?
- **How to combine data from multiple tables?**
- How to handle time series data with ease?
- How to manipulate textual data?

Doing it by yourself helps a lot!

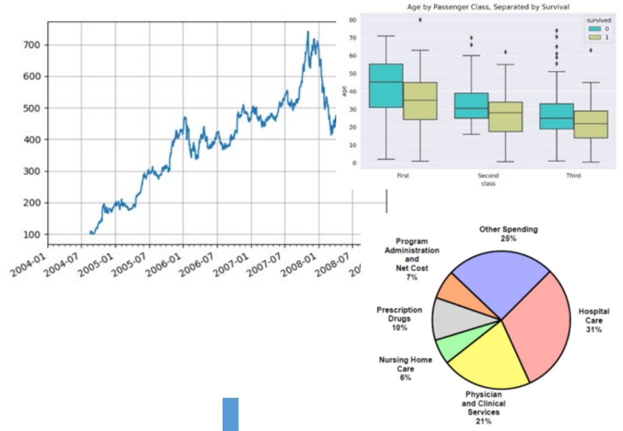
Data Visualization

Data Analysis, Exploration, and Visualization



141	137	134	134	132	130	129	129	131	135	130	128	129	126	128	128	130
138	136	134	134	135	133	131	129	132	139	133	128	130	128	127	129	131
135	135	134	133	133	132	130	128	132	136	134	130	131	131	132	132	133
133	134	133	132	131	130	130	131	131	129	134	134	130	134	137	134	134
134	134	134	134	133	132	134	138	136	127	135	137	132	136	140	135	139
137	135	136	138	137	135	137	143	142	132	136	138	135	137	138	138	142
139	135	135	138	138	134	135	141	143	133	133	134	135	135	133	138	140
136	137	137	138	141	143	142	144	140	143	142	137	137	139	137	135	136
137	138	136	136	138	140	141	143	140	144	143	139	139	140	138	137	139
137	139	137	136	136	136	137	140	143	146	143	140	141	142	142	143	143
137	140	141	139	138	136	135	137	143	144	142	139	142	144	145	147	146
140	144	144	143	141	137	135	137	139	139	139	139	143	145	146	147	147
145	148	147	145	143	140	139	141	136	138	140	142	147	147	146	147	149
146	148	147	144	143	141	140	143	137	139	142	145	146	145	145	148	147
145	147	146	143	142	140	140	143	138	140	143	143	143	141	143	148	142
145	145	144	144	143	141	141	142	142	145	146	145	144	141	143	150	144
144	143	142	143	143	142	142	144	143	144	143	144	148	144	142	147	145
146	145	144	143	143	143	144	146	144	144	141	146	157	154	144	143	148
149	148	145	144	143	143	144	145	144	146	142	149	167	169	155	146	151
150	149	147	145	142	142	143	143	145	147	143	147	166	175	164	151	152
150	150	149	147	145	145	145	145	147	148	143	142	154	165	160	148	150
152	152	152	150	149	150	150	149	151	151	150	147	146	152	153	147	151
152	153	153	152	151	151	151	150	152	152	156	155	148	149	155	153	152
152	152	152	152	152	151	151	151	152	152	152	153	152	151	152	153	154
152	152	152	152	152	152	151	151	152	152	152	153	152	151	151	152	154
153	153	153	153	153	153	153	153	154	154	153	153	152	152	150	152	154
153	153	153	153	154	154	154	154	154	154	153	153	153	153	152	153	155
153	153	152	153	154	154	154	154	153	154	154	153	153	153	153	154	157
153	152	152	152	154	155	155	155	153	155	155	154	152	152	152	154	159

Encoding



Iterate

Visual Perception

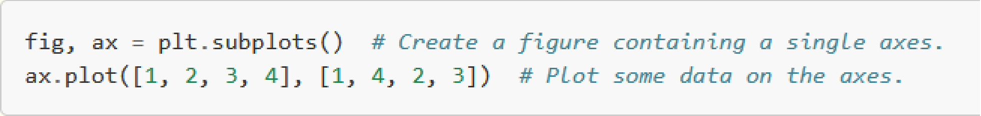
Understanding

Data visualization in Python…

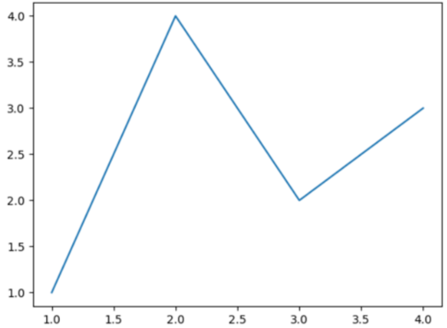
matplotlib

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

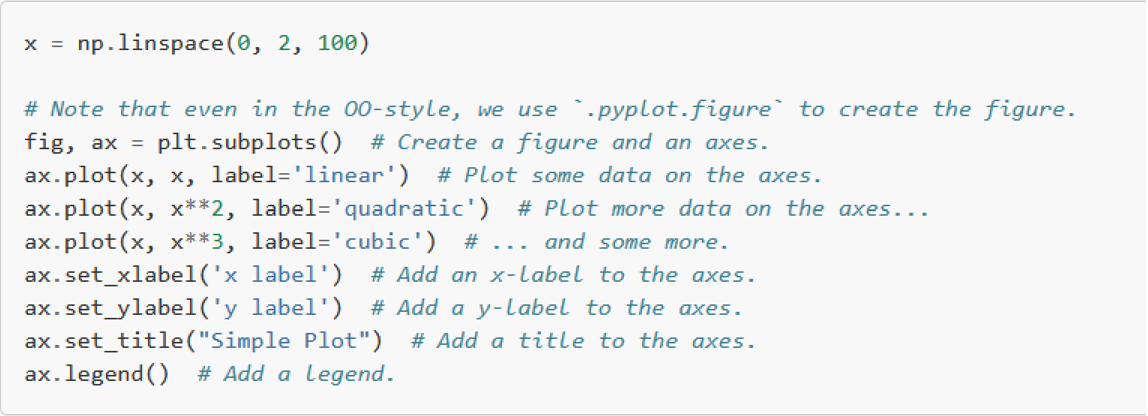
Create a simple figure with an axis object



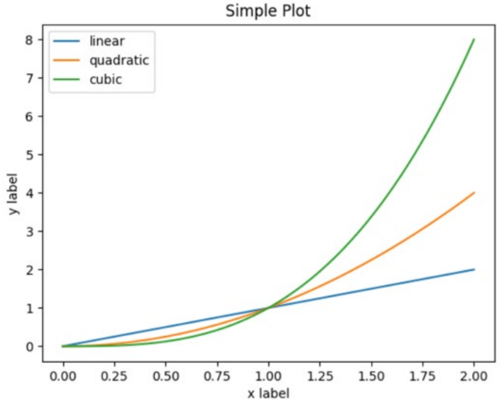
```
fig, ax = plt.subplots()  # Create a figure containing a single axes.
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])  # Plot some data on the axes.
```



A more complicated plot…

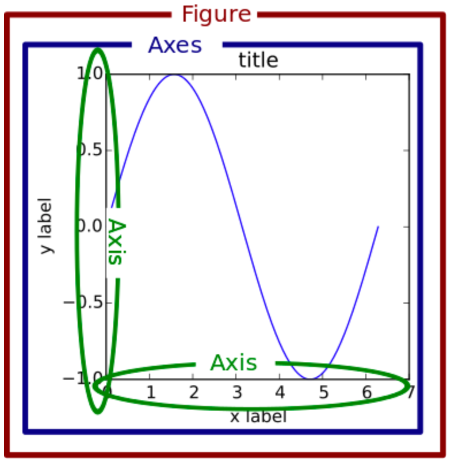


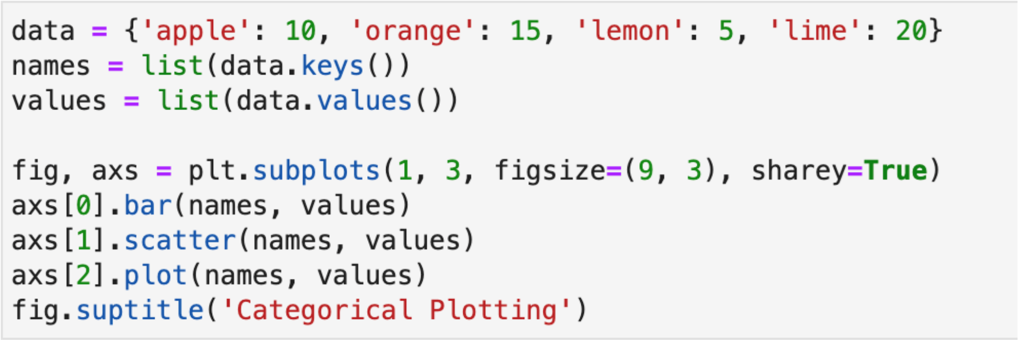
```
x = np.linspace(0, 2, 100)

# Note that even in the OO-style, we use `.pyplot.figure` to create the figure.
fig, ax = plt.subplots()  # Create a figure and an axes.
ax.plot(x, x, label='linear')  # Plot some data on the axes.
ax.plot(x, x**2, label='quadratic')  # Plot more data on the axes...
ax.plot(x, x**3, label='cubic')  # ... and some more.
ax.set_xlabel('x label')  # Add an x-label to the axes.
ax.set_ylabel('y label')  # Add a y-label to the axes.
ax.set_title("Simple Plot")  # Add a title to the axes.
ax.legend()  # Add a legend.
```



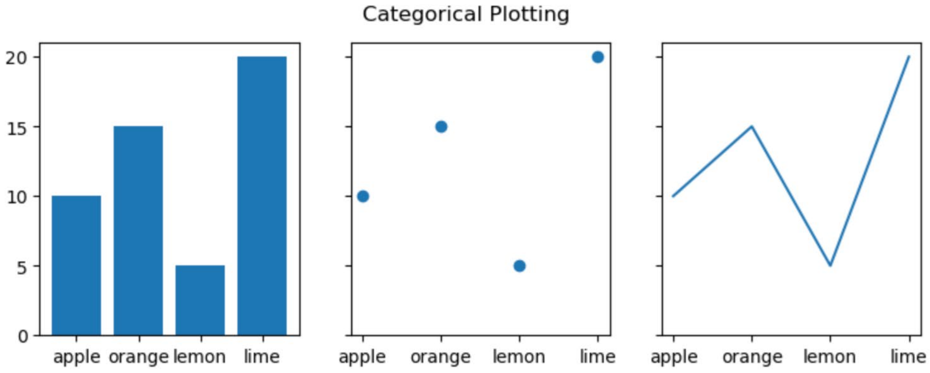
Axes: entire area of plot
Axis: horizontal or vertical (2d)

`subplot()` function: draw multiple plots in one figure

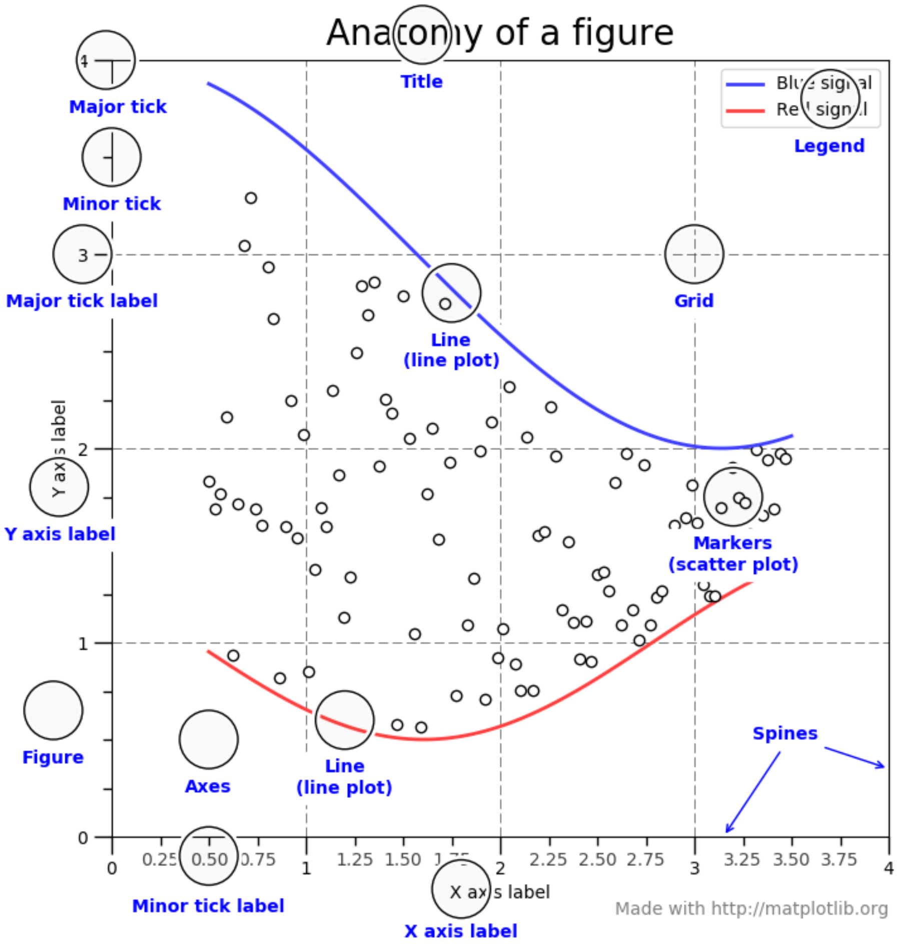




```python
data = {'apple': 10, 'orange': 15, 'lemon': 5, 'lime': 20}
names = list(data.keys())
values = list(data.values())

fig, axs = plt.subplots(1, 3, figsize=(9, 3), sharey=True)
axs[0].bar(names, values)
axs[1].scatter(names, values)
axs[2].plot(names, values)
fig.suptitle('Categorical Plotting')
```



matplotlib



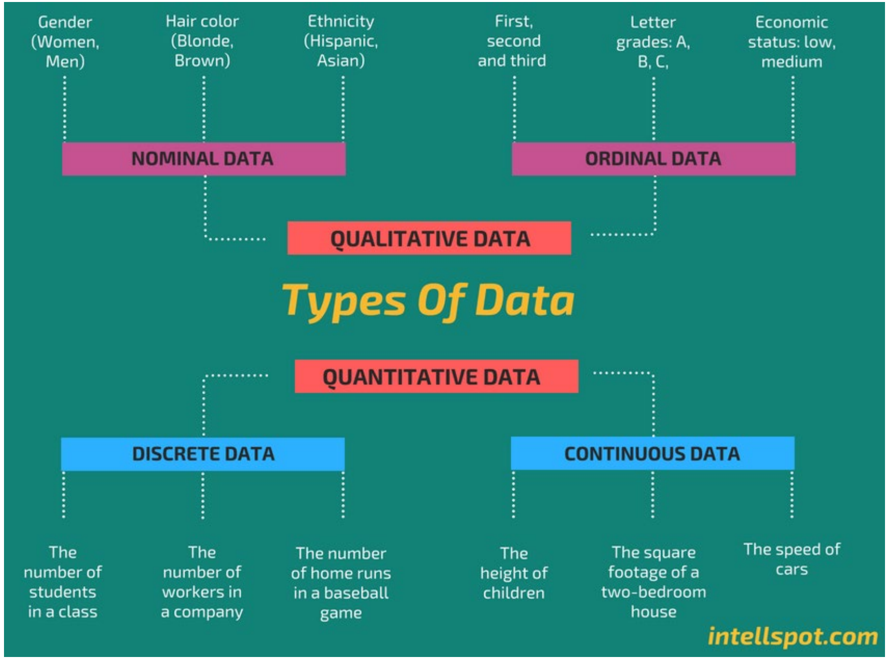
components of a Matplotlib figure

Documentation + tutorials:

https://matplotlib.org/

Types of Data

Data come in many forms, each requiring different approaches & models


Gender (Women, Men)
Hair color (Blonde, Brown)
Ethnicity (Hispanic, Asian)
First, second and third
Letter grades: A, B, C,
Economic status: low, medium
NOMINAL DATA
ORDINAL DATA
QUALITATIVE DATA
Types Of Data
QUANTITATIVE DATA
DISCRETE DATA
CONTINUOUS DATA
The number of students in a class
The number of workers in a company
The number of home runs in a baseball game
The height of children
The square footage of a two-bedroom house
The speed of cars
intellspot.com

Qualitative or **categorical**: can partition values into classes

Quantitative: can perform arithmetic operations (e.g., addition, subtraction, ordering)

*We often refer to different types of data as **variables***

Categorical Variables

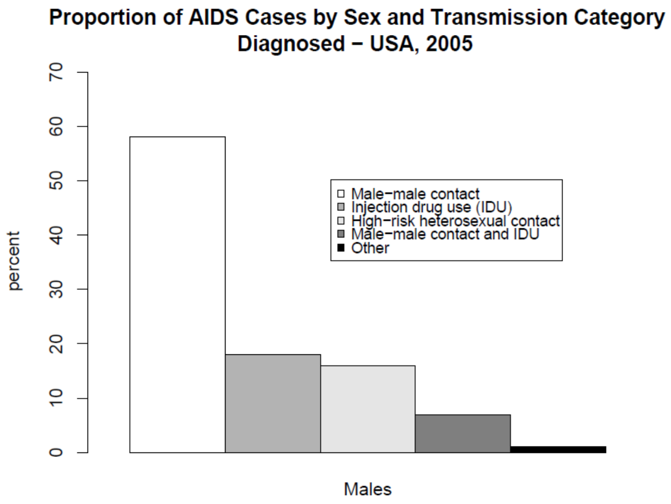
Examples

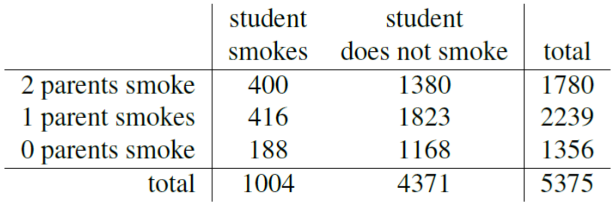
- Roll of a die: 1,2,3,4,5 or 6 ← **Numerical data can be categorical or quantitative depending on context**
- Blood Type: A, B, AB, or O
- Political Party: Democrat, Republican, etc.
- Type of Rock: Igneous, Sedimentary, or Metamorphic
- Word Identity: NP, VP, N, V, Adj, Adv, etc.

Conversion: Quantitative data can be converted to categorical by defining ranges:

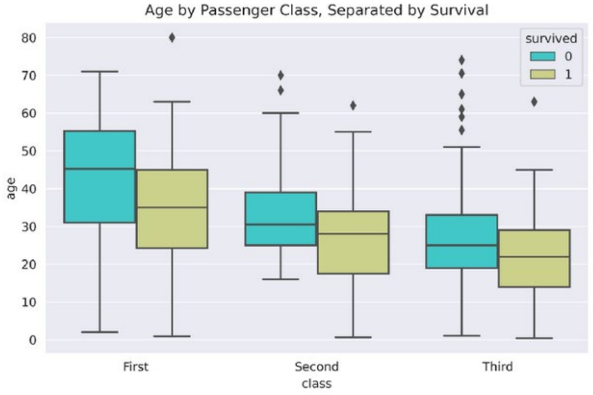
- Small [0, 10cm), Medium [10, 100cm), Large [100cm, 1m), XL [1m, -)
- Low [less than -100dB), Moderate [-100dB, -50dB), Loud [over -50dB)

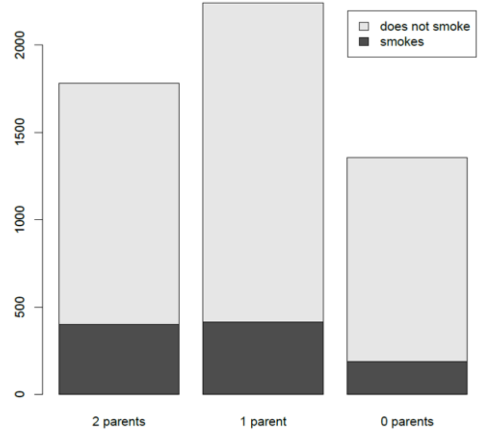
Visualizing Categorical Variables





	student smokes	student does not smoke	total
2 parents smoke	400	1380	1780
1 parent smokes	416	1823	2239
0 parents smoke	188	1168	1356
total	1004	4371	5375

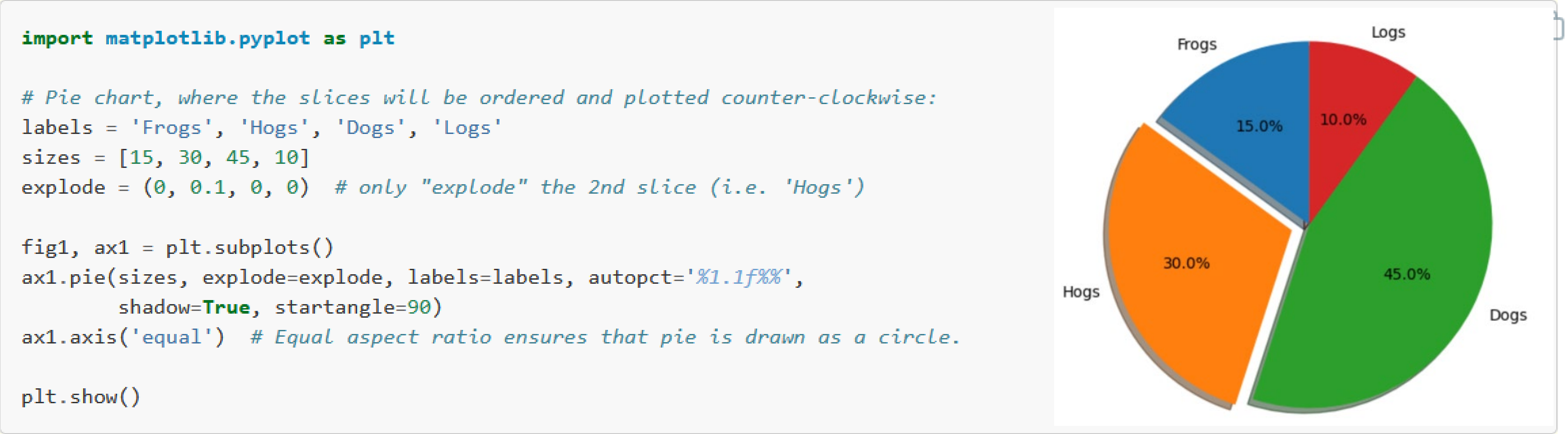




Pie Chart

Circular chart divided into sectors, illustrating relative magnitudes in frequencies or percentage.

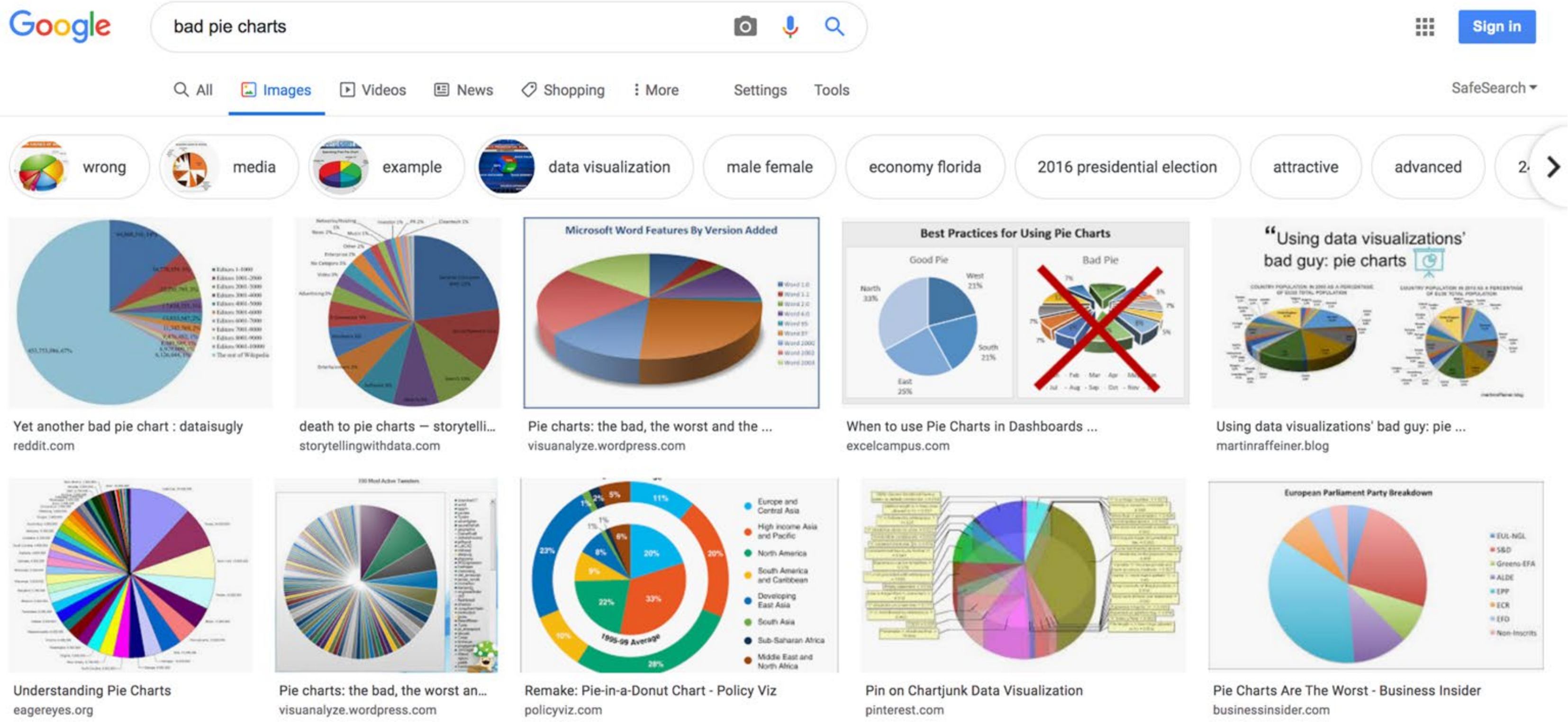
In a pie chart, *the area is proportional to the quantity it represents*



```python
import matplotlib.pyplot as plt

# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0)  # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()
ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

plt.show()
```

Be careful with using pie charts:

- Maybe unsuitable if too many sectors are present
- 3d charts can distort the sizes of the sectors; using 2d is recommended


Google
bad pie charts
Sign in
All
Images
Videos
News
Shopping
More
Settings
Tools
SafeSearch
wrong
media
example
data visualization
male female
economy florida
2016 presidential election
attractive
advanced
Microsoft Word Features By Version Added
Best Practices for Using Pie Charts
Good Pie
Bad Pie
"Using data visualizations' bad guy: pie charts
European Parliament Party Breakdown

Yet another bad pie chart : dataisugly
reddit.com

death to pie charts — storytelli...
storytellingwithdata.com

Pie charts: the bad, the worst and the ...
visuanalyze.wordpress.com

When to use Pie Charts in Dashboards ...
excelcampus.com

Using data visualizations' bad guy: pie ...
martinraffeiner.blog

Understanding Pie Charts
eagereyes.org

Pie charts: the bad, the worst an...
visuanalyze.wordpress.com

Remake: Pie-in-a-Donut Chart - Policy Viz
policyviz.com

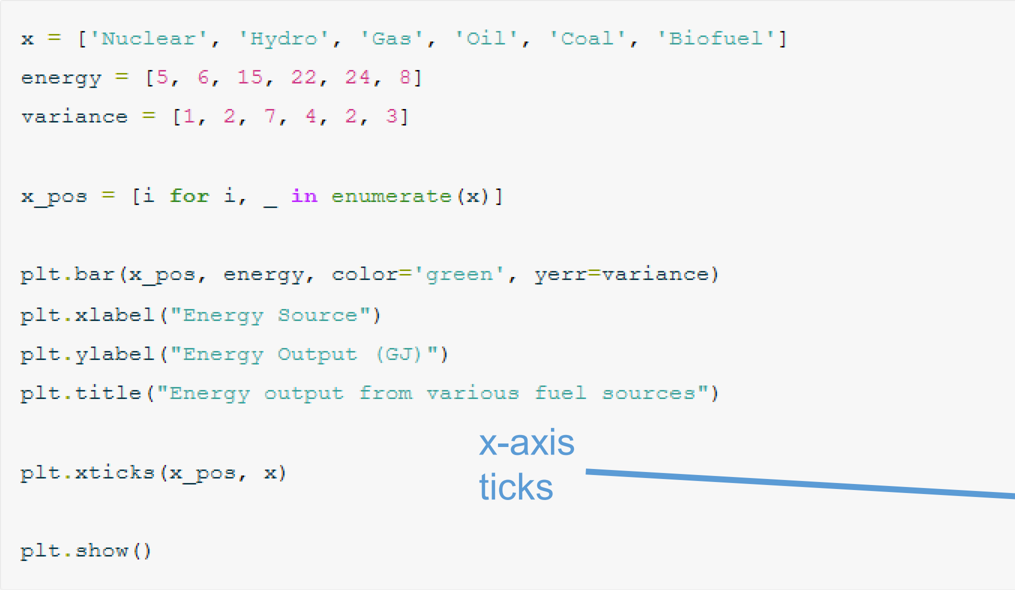
Pin on Chartjunk Data Visualization
pinterest.com

Pie Charts Are The Worst - Business Insider
businessinsider.com

Bar Chart

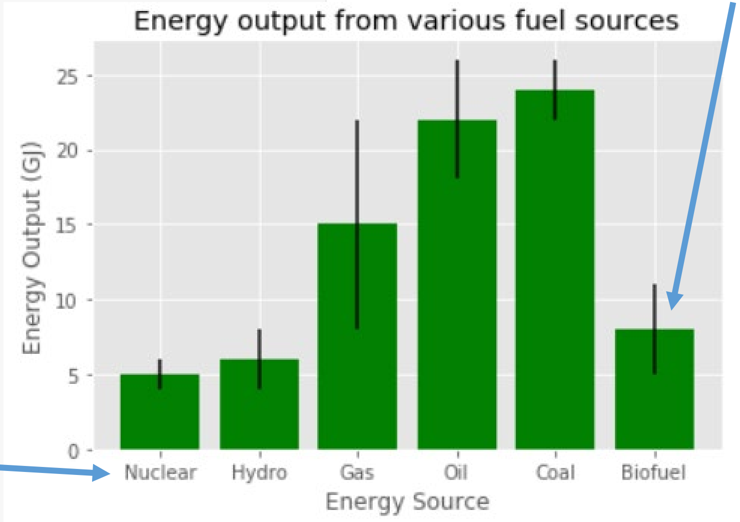
We perceive differences in height / length better than area…

`plt.bar()`



```
x = ['Nuclear', 'Hydro', 'Gas', 'Oil', 'Coal', 'Biofuel']
energy = [5, 6, 15, 22, 24, 8]
variance = [1, 2, 7, 4, 2, 3]

x_pos = [i for i, _ in enumerate(x)]

plt.bar(x_pos, energy, color='green', yerr=variance)
plt.xlabel("Energy Source")
plt.ylabel("Energy Output (GJ)")
plt.title("Energy output from various fuel sources")

plt.xticks(x_pos, x)

plt.show()
```

x-axis ticks



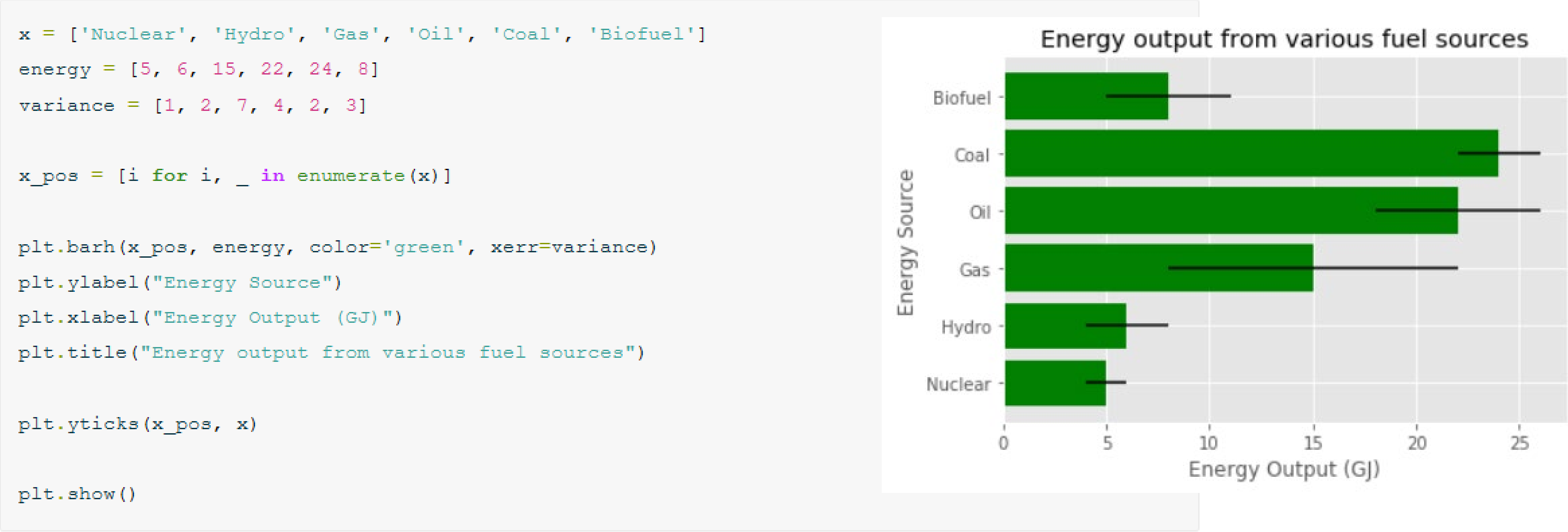
error bars

[Source: https://benalexkeen.com/bar-charts-in-matplotlib/]

Bar Chart

Horizontal version

`plt.barh()`

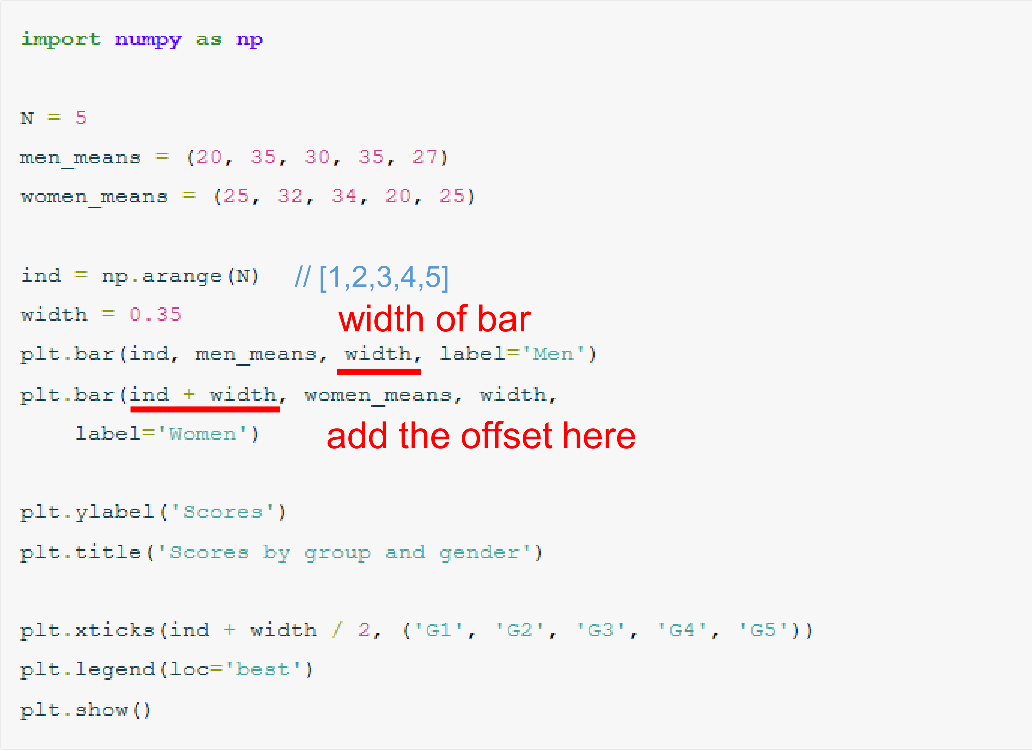


```
x = ['Nuclear', 'Hydro', 'Gas', 'Oil', 'Coal', 'Biofuel']
energy = [5, 6, 15, 22, 24, 8]
variance = [1, 2, 7, 4, 2, 3]

x_pos = [i for i, _ in enumerate(x)]

plt.barh(x_pos, energy, color='green', xerr=variance)
plt.ylabel("Energy Source")
plt.xlabel("Energy Output (GJ)")
plt.title("Energy output from various fuel sources")

plt.yticks(x_pos, x)

plt.show()
```

[Source: https://benalexkeen.com/bar-charts-in-matplotlib/]

Bar Chart

Multiple groups of bars…

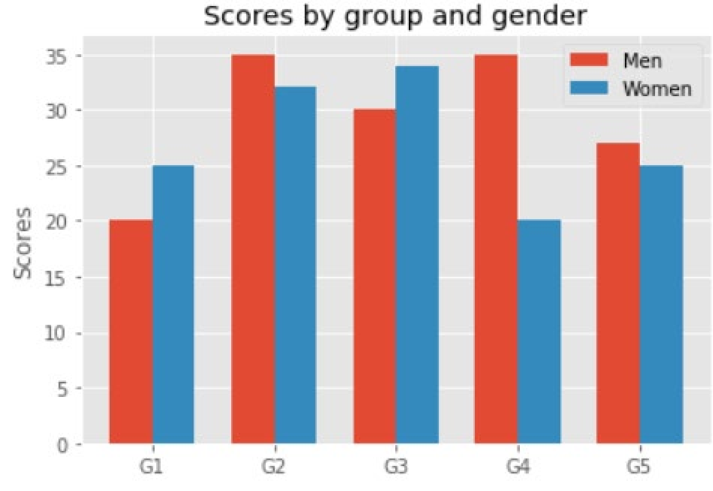


```
import numpy as np

N = 5
men_means = (20, 35, 30, 35, 27)
women_means = (25, 32, 34, 20, 25)

ind = np.arange(N)
width = 0.35
plt.bar(ind, men_means, width, label='Men')
plt.bar(ind + width, women_means, width,
    label='Women')

plt.ylabel('Scores')
plt.title('Scores by group and gender')

plt.xticks(ind + width / 2, ('G1', 'G2', 'G3', 'G4', 'G5'))
plt.legend(loc='best')
plt.show()
```

// [1,2,3,4,5]

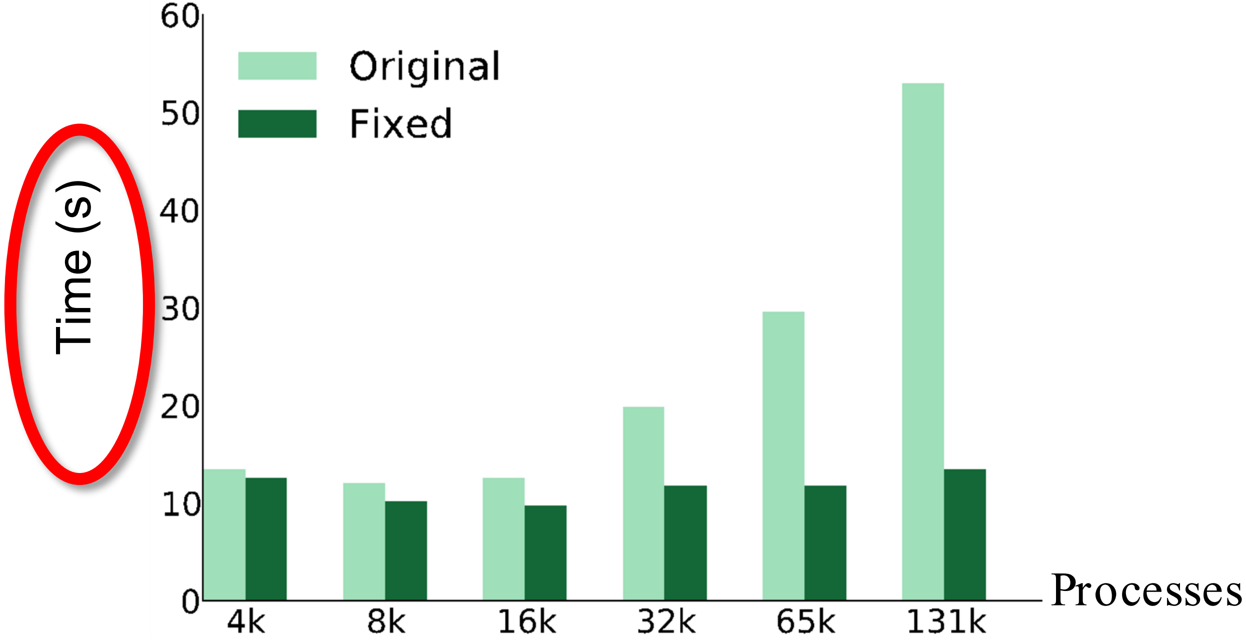
width of bar

add the offset here



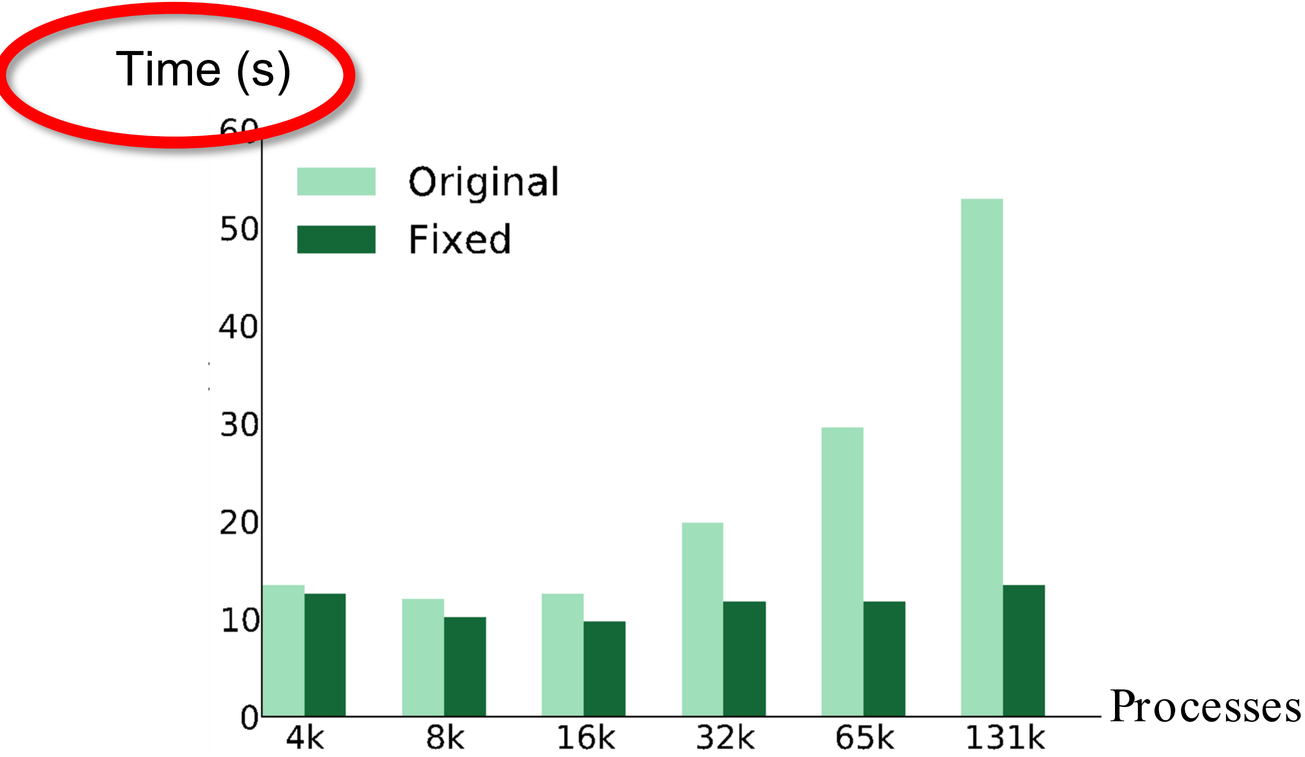
[Source: https://benalexkeen.com/bar-charts-in-matplotlib/]

Labels on the y-axis need not be vertical



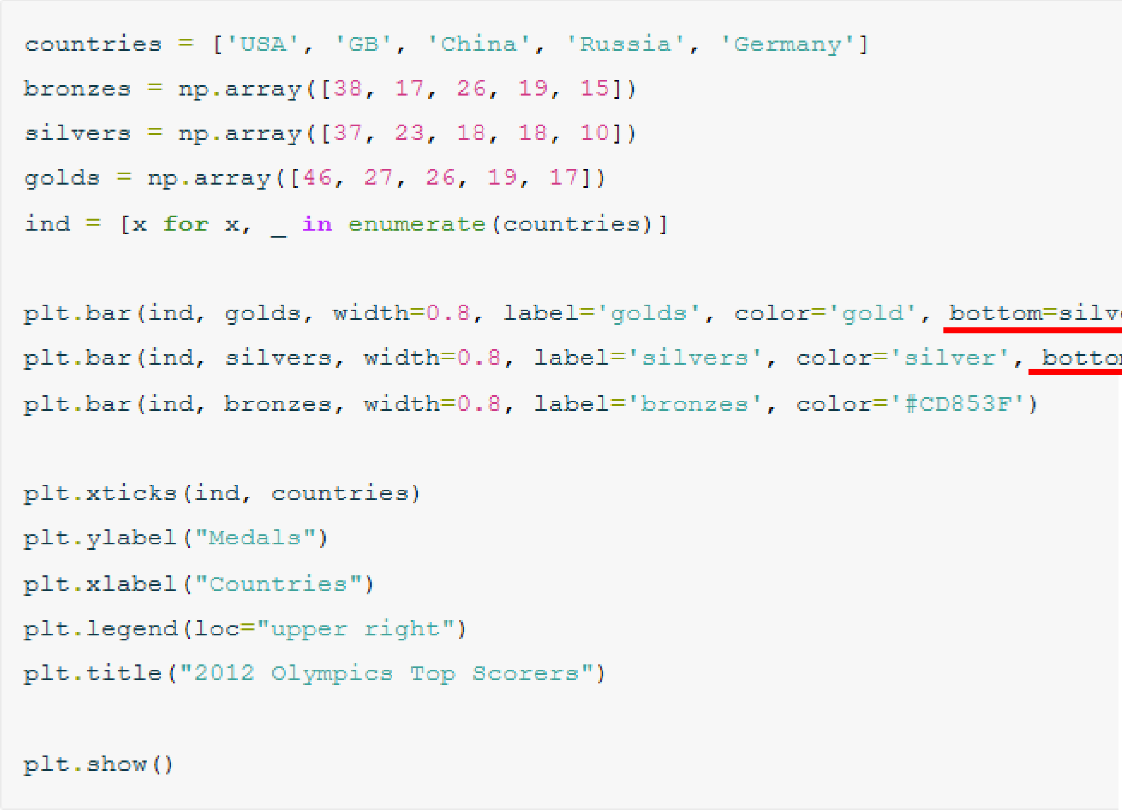
[Source: Kate Isaacs]

Labels on the y-axis need not be vertical

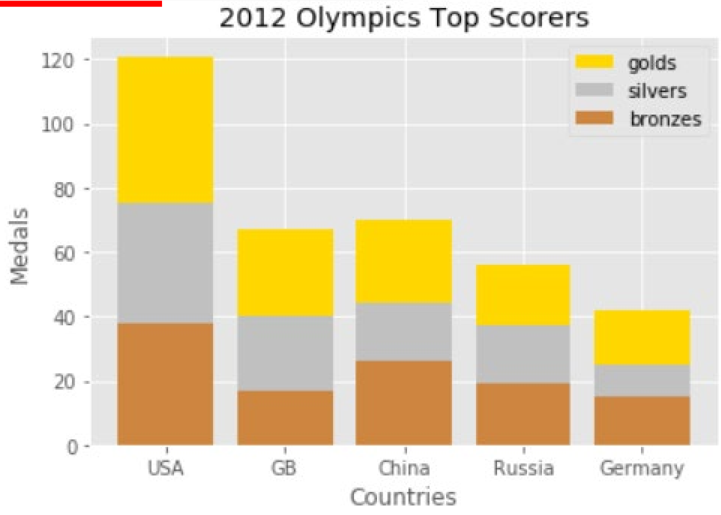


[Source: Kate Isaacs]

Stacked Bar Chart

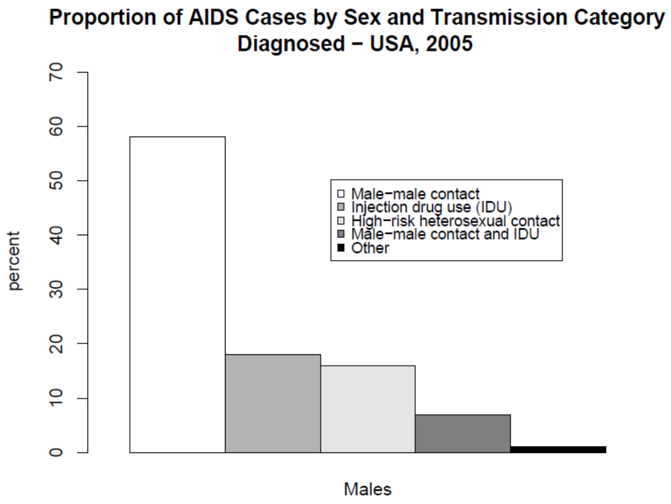


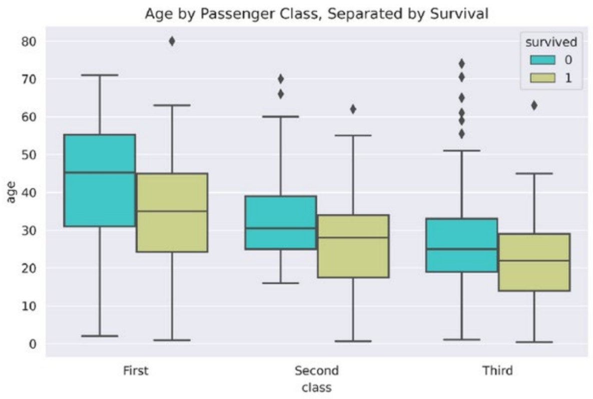
```
countries = ['USA', 'GB', 'China', 'Russia', 'Germany']
bronzes = np.array([38, 17, 26, 19, 15])
silvers = np.array([37, 23, 18, 18, 10])
golds = np.array([46, 27, 26, 19, 17])
ind = [x for x, _ in enumerate(countries)]

plt.bar(ind, golds, width=0.8, label='golds', color='gold', bottom=silvers+bronzes)
plt.bar(ind, silvers, width=0.8, label='silvers', color='silver', bottom=bronzes)
plt.bar(ind, bronzes, width=0.8, label='bronzes', color='#CD853F')

plt.xticks(ind, countries)
plt.ylabel("Medals")
plt.xlabel("Countries")
plt.legend(loc="upper right")
plt.title("2012 Olympics Top Scorers")

plt.show()
```

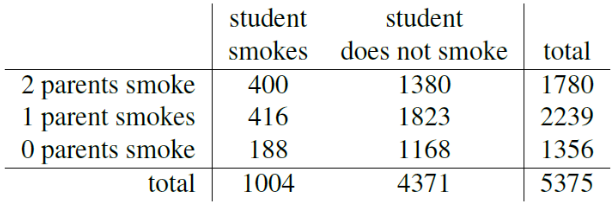


[Source: https://benalexkeen.com/bar-charts-in-matplotlib/]

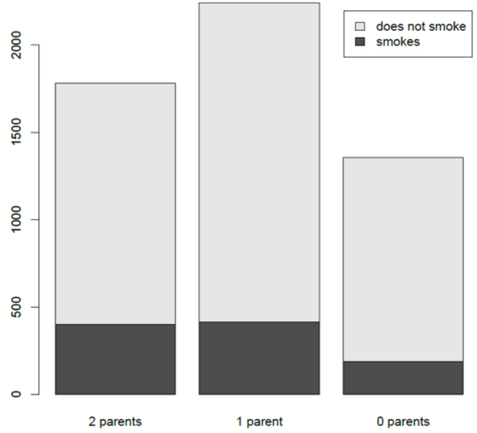
Visualizing Categorical Variables







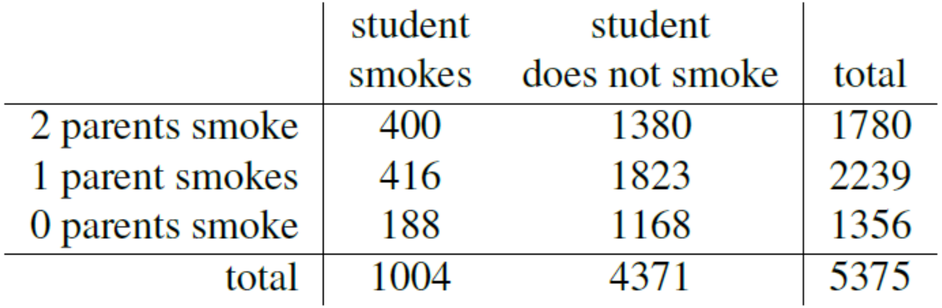
	student smokes	student does not smoke	total
2 parents smoke	400	1380	1780
1 parent smokes	416	1823	2239
0 parents smoke	188	1168	1356
total	1004	4371	5375



Two-Way Table

Also called contingency table or cross tabulation table…

Example We asked 5375 students and collected their smoking status and their parents’ smoking status, and summarize it as:

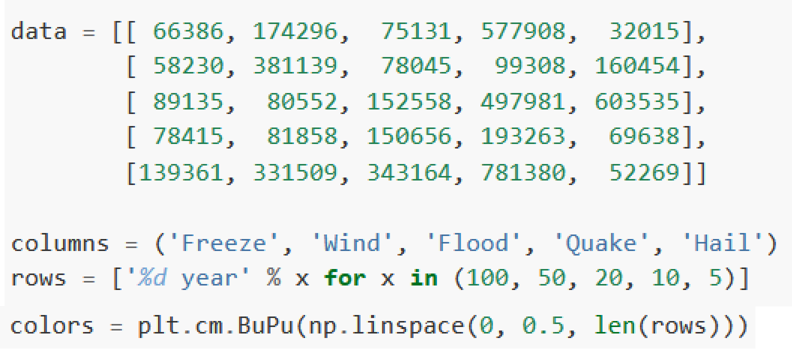


	student smokes	student does not smoke	total
2 parents smoke	400	1380	1780
1 parent smokes	416	1823	2239
0 parents smoke	188	1168	1356
total	1004	4371	5375

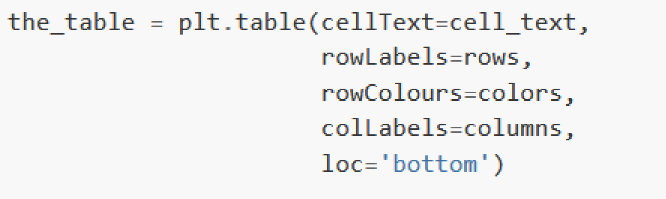
Q: is there any correlation between parents’ and child’s smoking statuses?

E.g. are students with 2 parents smoking more likely to smoke (compared with general students)?

Two-Way Table

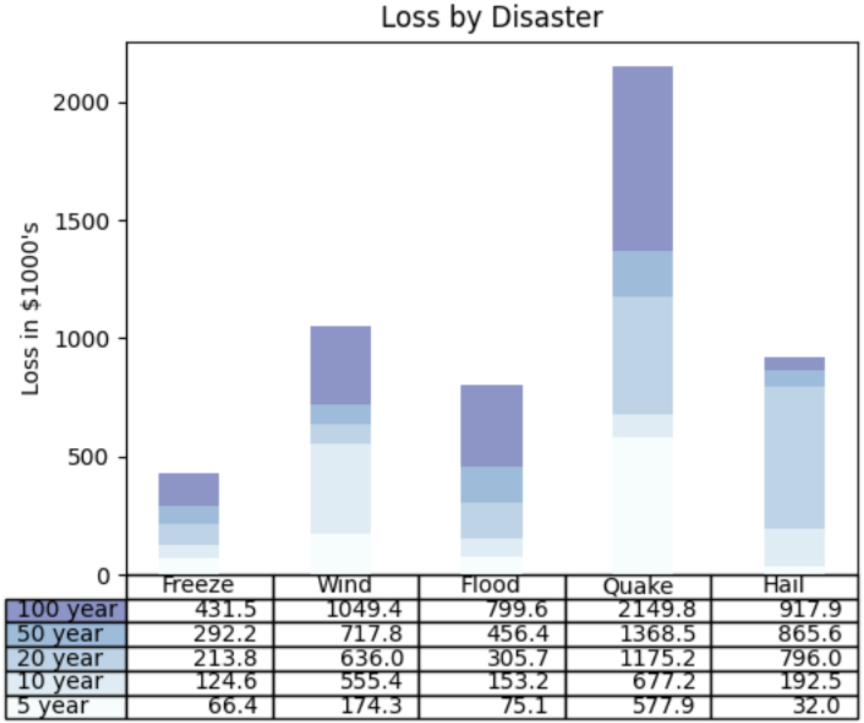


```
data = [[ 66386, 174296,  75131, 577908,  32015],
        [ 58230, 381139,  78045,  99308, 160454],
        [ 89135,  80552, 152558, 497981, 603535],
        [ 78415,  81858, 150656, 193263,  69638],
        [139361, 331509, 343164, 781380,  52269]]

columns = ('Freeze', 'Wind', 'Flood', 'Quake', 'Hail')
rows = ['%d year' % x for x in (100, 50, 20, 10, 5)]
colors = plt.cm.BuPu(np.linspace(0, 0.5, len(rows)))
```



```
the_table = plt.table(cellText=cell_text,
                      rowLabels=rows,
                      rowColours=colors,
                      colLabels=columns,
                      loc='bottom')
```

Adding stacked bars requires more steps, full code here: https://matplotlib.org/stable/gallery/misc/table_demo.html



	Freeze	Wind	Flood	Quake	Hail
100 year	431.5	1049.4	799.6	2149.8	917.9
50 year	292.2	717.8	456.4	1368.5	865.6
20 year	213.8	636.0	305.7	1175.2	796.0
10 year	124.6	555.4	153.2	677.2	192.5
5 year	66.4	174.3	75.1	577.9	32.0

Quiz

Quiz

- What are the mean, median, mode, range interval of the following dataset?

4, 9, 10, 6, 6

Measuring Spread: sample variance

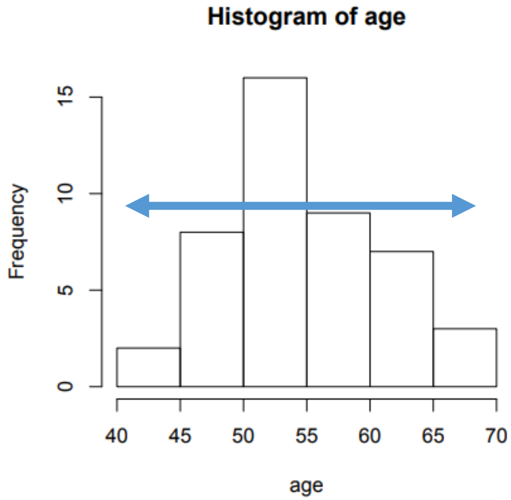
Another way to measure the spread is the sample variance,

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 \qquad s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

Biased

Sample mean

Unbiased



Sample Variance

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

Example calculate the sample variance of sample

4, 9, 10, 6, 6

Sample mean: $\bar{x} = \frac{4+9+10+6+6}{5} = 7$

5 terms in the summation:

$$(4-7)^2, (9-7)^2, (10-7)^2, (6-7)^2, (6-7)^2$$

$$9, \quad 4, \quad 9, \quad 1, \quad 1$$

$$\sigma^2 = \frac{1}{5}(9+4+9+1+1) = 4.8$$

Sample variance

- When is the variance of a sample zero?

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

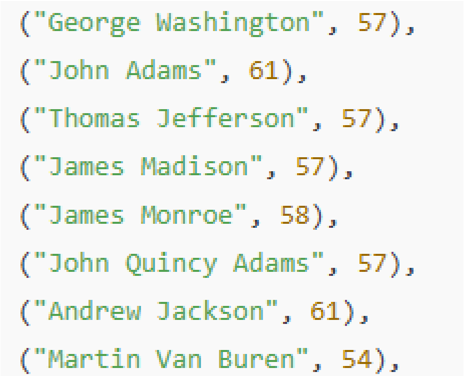
- Variance of a sample is zero if all x_i's are identical, e.g.

5, 5, .., 5

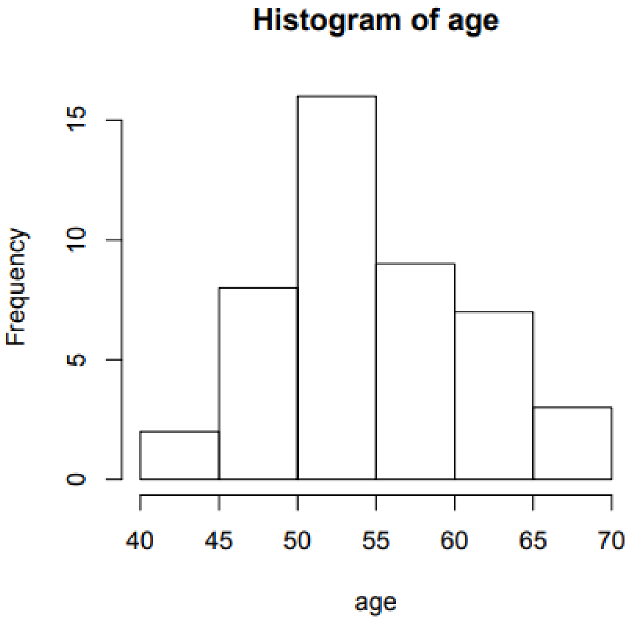
- Variance measures the degree of "fluctuations" in the data
- The square root of variance, σ, is called the *standard deviation*

Histogram

Example US presidents’ ages at inauguration



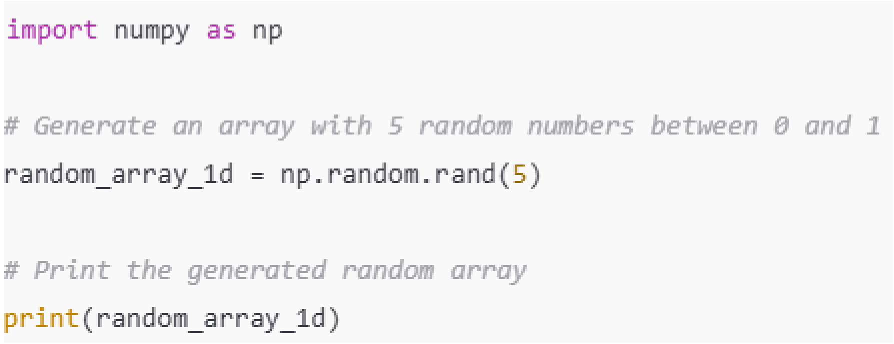
```
("George Washington", 57),
("John Adams", 61),
("Thomas Jefferson", 57),
("James Madison", 57),
("James Monroe", 58),
("John Quincy Adams", 57),
("Andrew Jackson", 61),
("Martin Van Buren", 54),
```



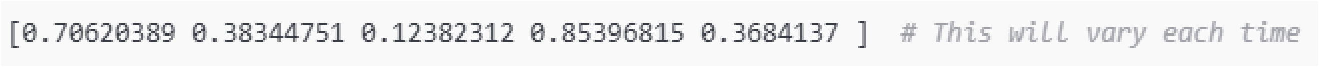
Aside: generating random data

- Numpy: Python lib for scientific computing

- It has general-purpose random number generator *rand*

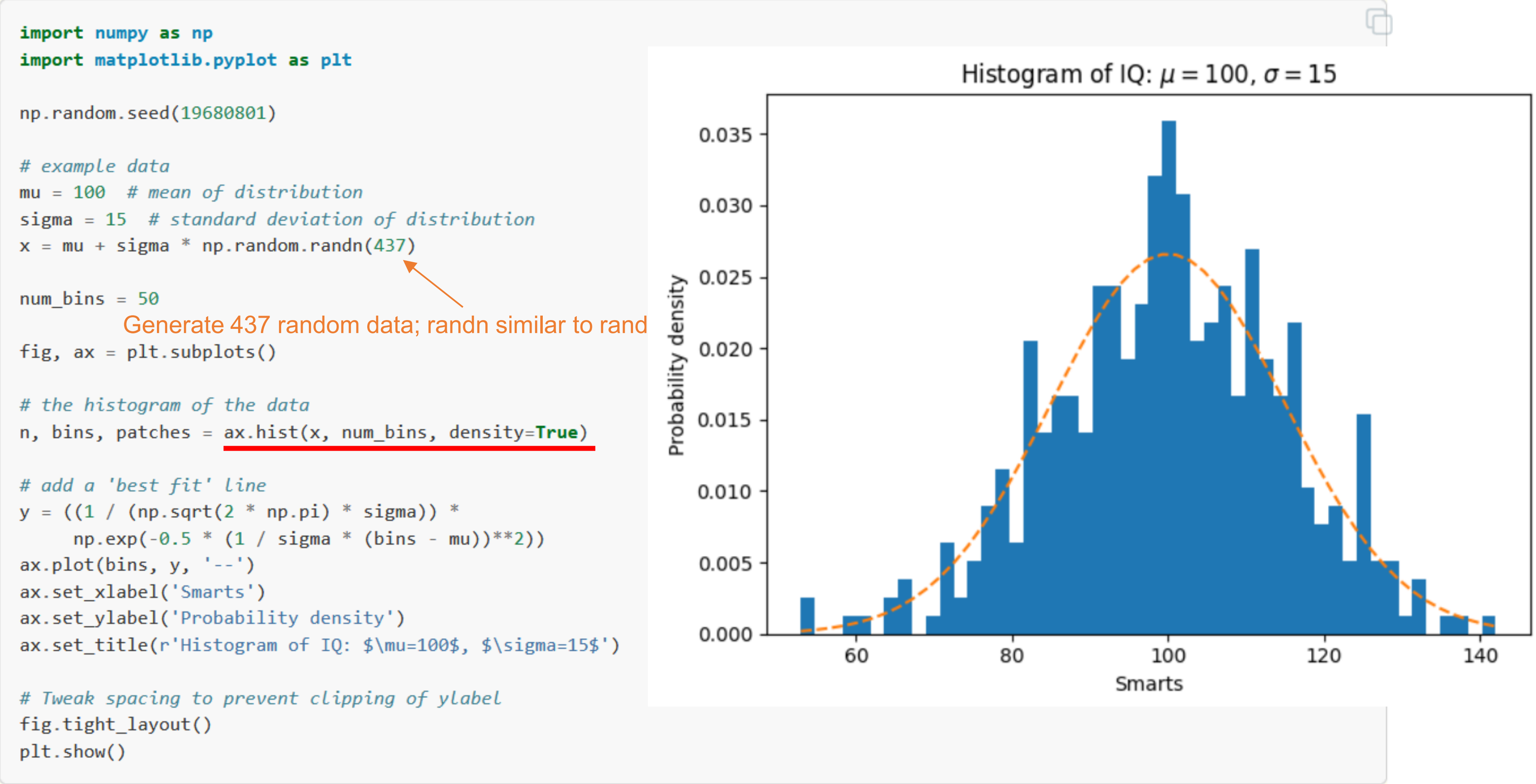


```
import numpy as np

# Generate an array with 5 random numbers between 0 and 1
random_array_1d = np.random.rand(5)

# Print the generated random array
print(random_array_1d)
```



```
[0.70620389 0.38344751 0.12382312 0.85396815 0.3684137 ]  # This will vary each time
```

Histogram

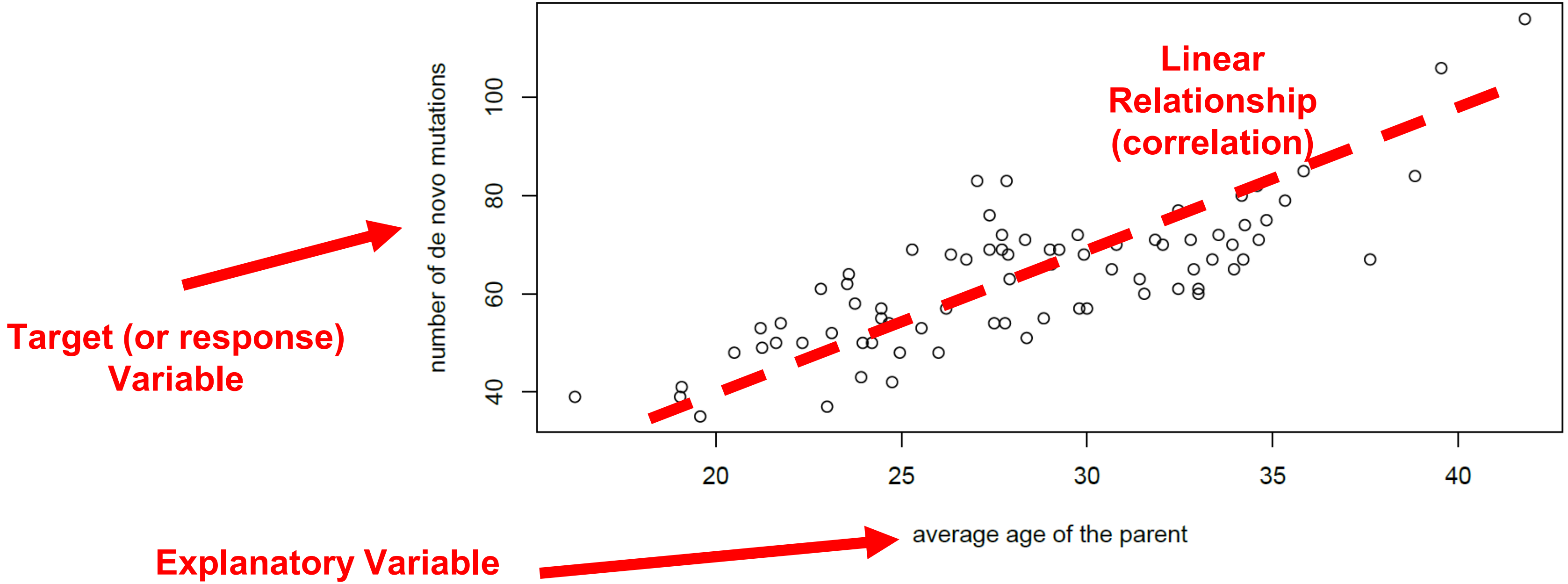


```
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(19680801)

# example data
mu = 100  # mean of distribution
sigma = 15  # standard deviation of distribution
x = mu + sigma * np.random.randn(437)

num_bins = 50

fig, ax = plt.subplots()

# the histogram of the data
n, bins, patches = ax.hist(x, num_bins, density=True)

# add a 'best fit' line
y = ((1 / (np.sqrt(2 * np.pi) * sigma)) *
     np.exp(-0.5 * (1 / sigma * (bins - mu))**2))
ax.plot(bins, y, '--')
ax.set_xlabel('Smarts')
ax.set_ylabel('Probability density')
ax.set_title(r'Histogram of IQ: $\mu=100$, $\sigma=15$')

# Tweak spacing to prevent clipping of ylabel
fig.tight_layout()
plt.show()
```

Generate 437 random data; randn similar to rand

Scatterplot

Compares relationship between two quantitative variables…



Target (or response) Variable

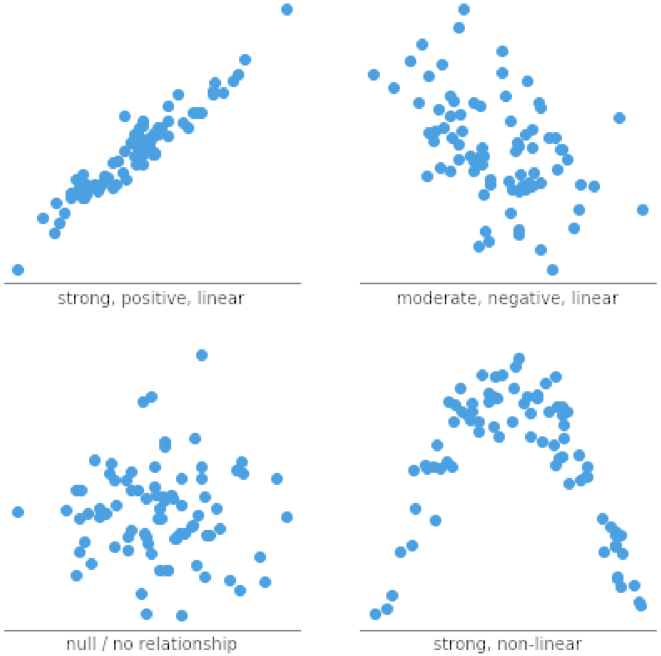
Explanatory Variable

Linear Relationship (correlation)

Useful for many prediction tasks:
e.g. house price prediction, salary prediction, stock price prediction, etc.

Scatterplot

Compares relationship between two quantitative variables…



strong, positive, linear

moderate, negative, linear

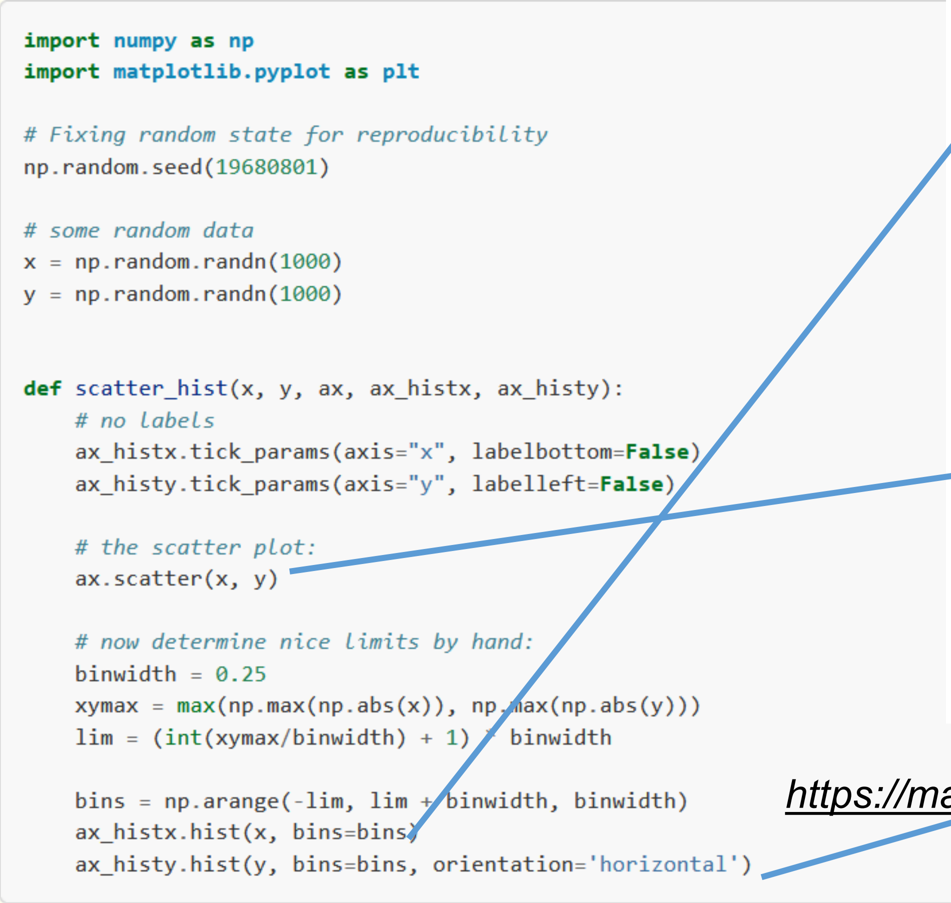
null / no relationship

strong, non-linear

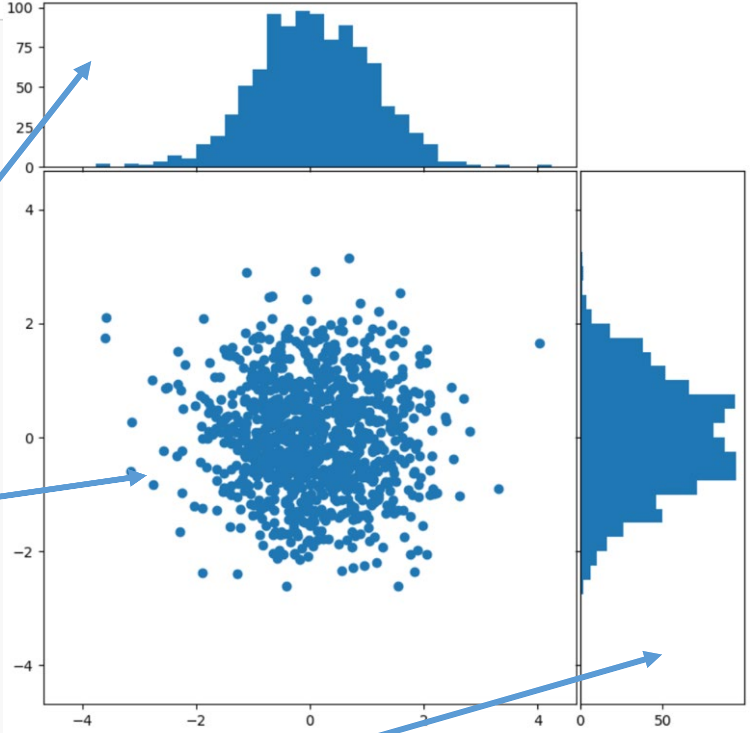
Relationship can also be:

- Nonlinear (e.g. “curvy”)
- Clustered or grouped

Scatterplot + Histogram



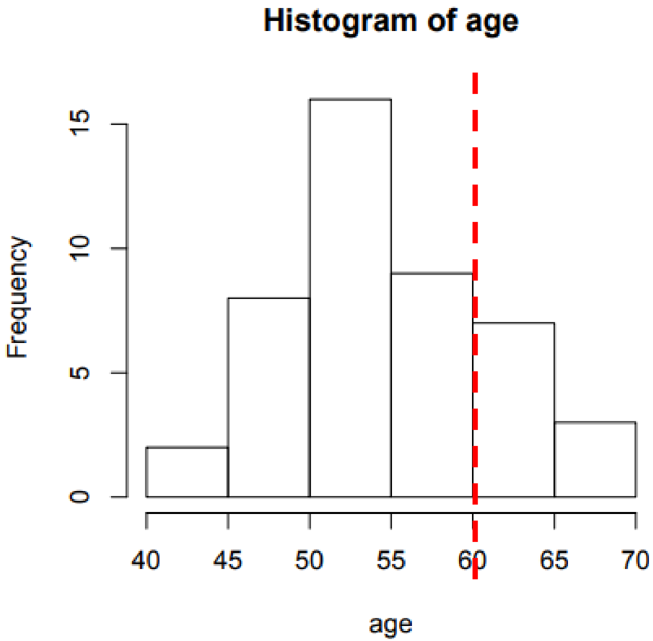
```python
import numpy as np
import matplotlib.pyplot as plt

# Fixing random state for reproducibility
np.random.seed(19680801)

# some random data
x = np.random.randn(1000)
y = np.random.randn(1000)


def scatter_hist(x, y, ax, ax_histx, ax_histy):
    # no labels
    ax_histx.tick_params(axis="x", labelbottom=False)
    ax_histy.tick_params(axis="y", labelleft=False)

    # the scatter plot:
    ax.scatter(x, y)

    # now determine nice limits by hand:
    binwidth = 0.25
    xymax = max(np.max(np.abs(x)), np.max(np.abs(y)))
    lim = (int(xymax/binwidth) + 1) * binwidth

    bins = np.arange(-lim, lim + binwidth, binwidth)
    ax_histx.hist(x, bins=bins)
    ax_histy.hist(y, bins=bins, orientation='horizontal')
```



Full Code:
https://matplotlib.org/stable/gallery/lines_bars_and_markers/scatter_hist.html

Percentile / Quartile

Question Is 60yrs old for a US president? Why or why not?



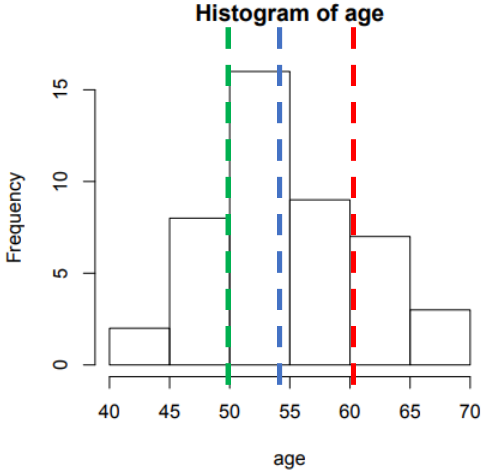
The number of presidents <60: 33
Total number of presidents: 44

About 75% of presidents younger than 60yrs old
=> 60yrs old = 0.75 Quantile or 75th Percentile

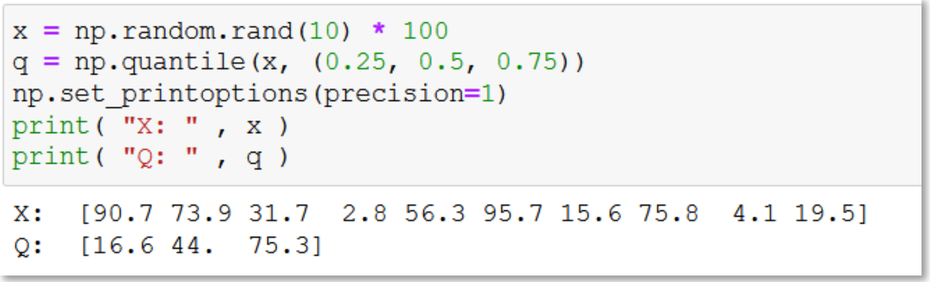
Measuring Spread

Quartile divide data into 4 equally-sized bins,

- **1st Quartile** : Lowest 25% of data
- **2nd Quartile** : Median (lowest 50% of data)
- **3rd Quartile** : 75% of data is below 3rd quartile
- **4th Quartile** : The maximum value



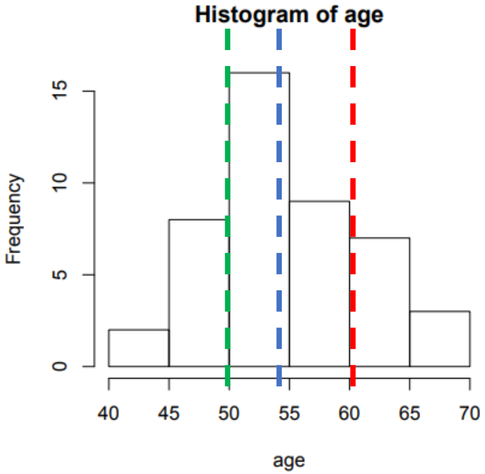
Compute using `np.quantile()` :



```
x = np.random.rand(10) * 100
q = np.quantile(x, (0.25, 0.5, 0.75))
np.set_printoptions(precision=1)
print( "X: " , x )
print( "Q: " , q )
```

```
X:  [90.7 73.9 31.7  2.8 56.3 95.7 15.6 75.8  4.1 19.5]
Q:  [16.6 44.  75.3]
```

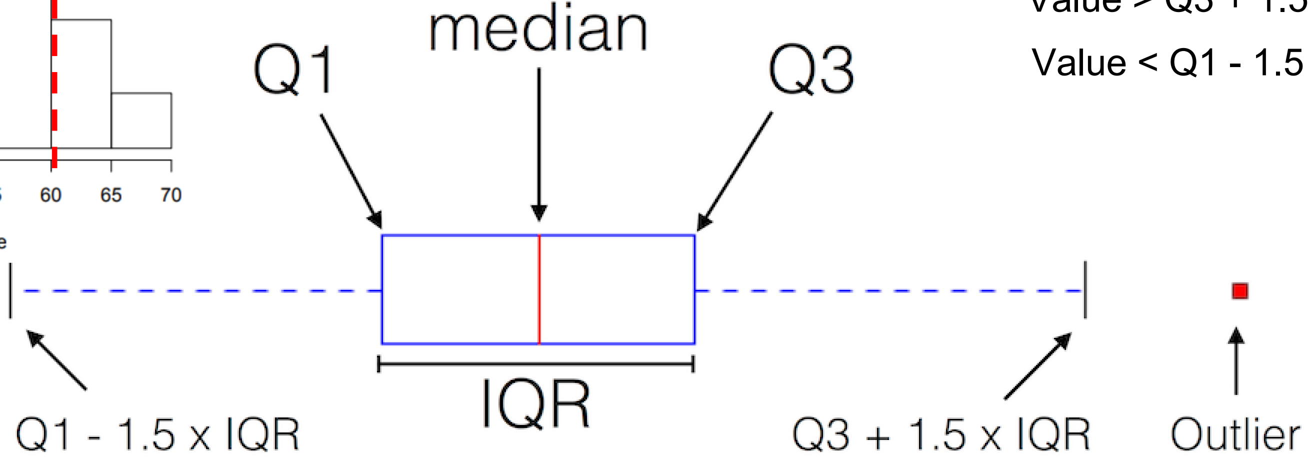
Box Plot



Outliers are atypical data whose

Value > Q3 + 1.5 x IQR

Value < Q1 - 1.5 x IQR



Q1

median

Q3

IQR

Q1 - 1.5 x IQR

Q3 + 1.5 x IQR

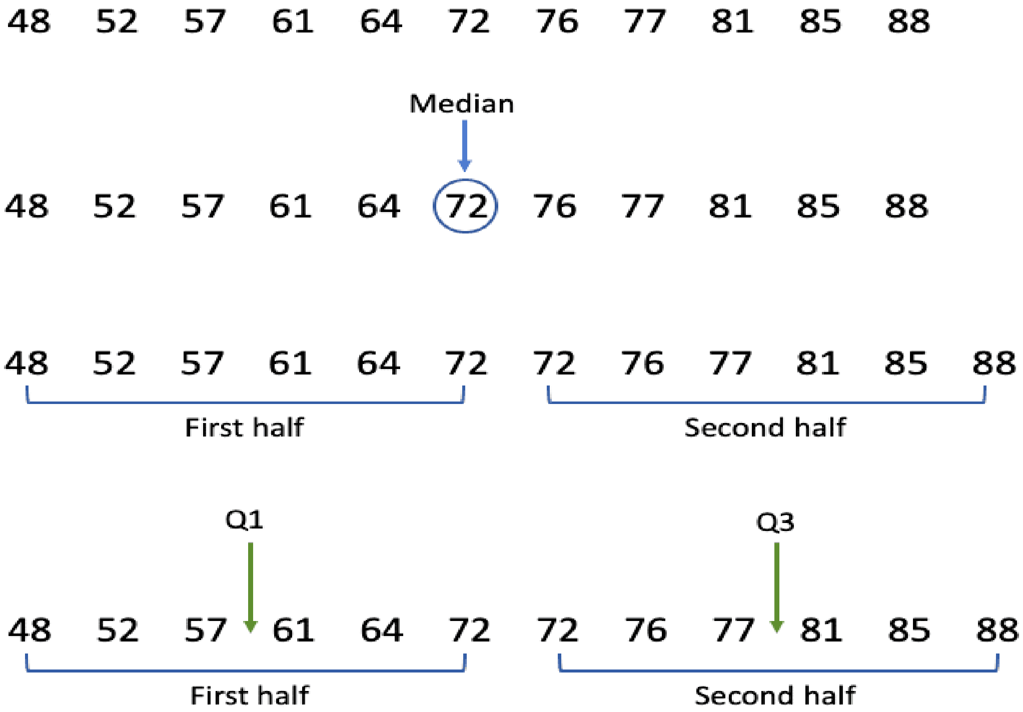
Outlier

Interquartile-Range (IQR) Measures interval containing 50% of data

IQR = Q3 – Q1

Region of *typical* data

Box Plot



48 52 57 61 64 72 76 77 81 85 88

Median

48 52 57 61 64 72 76 77 81 85 88

48 52 57 61 64 72 72 76 77 81 85 88

First half

Second half

Q1

Q3

48 52 57 61 64 72 72 76 77 81 85 88

First half

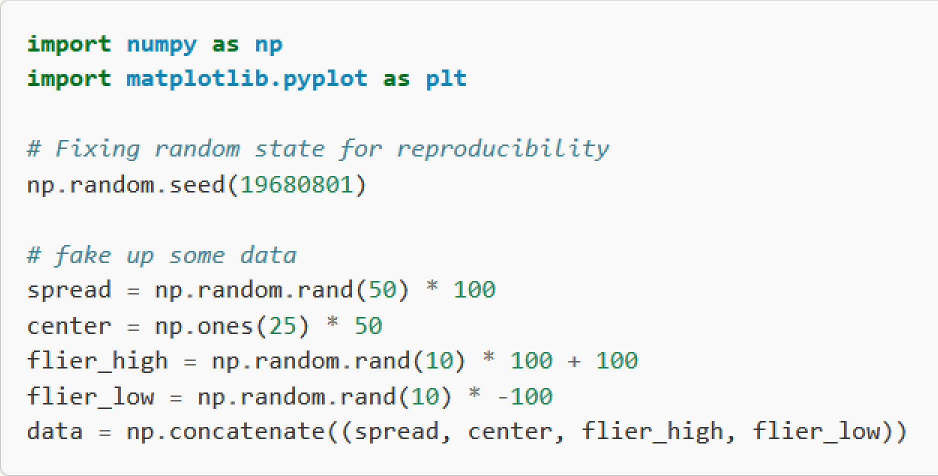
Second half

$$Q1 = \frac{57+61}{2} = 59$$

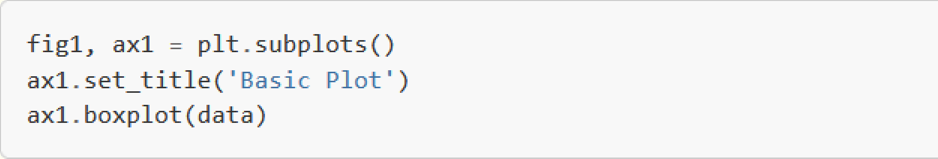
$$Q3 = \frac{77+81}{2} = 79$$

$$IQR = Q3 - Q1$$
$$IQR = 79 - 59 = 20$$

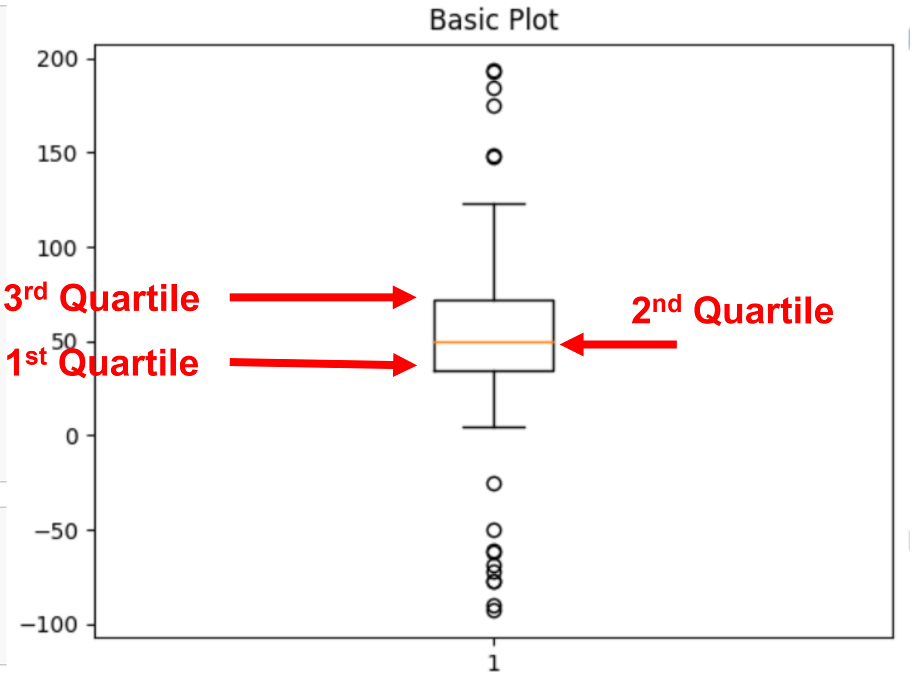
Box Plot



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

# Fixing random state for reproducibility
np.random.seed(19680801)

# 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))
```



```
fig1, ax1 = plt.subplots()
ax1.set_title('Basic Plot')
ax1.boxplot(data)
```



3rd Quartile

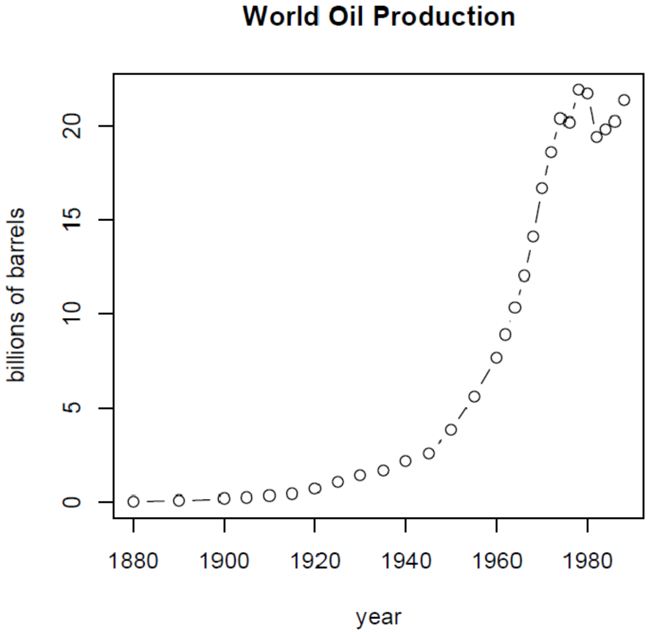
1st Quartile

2nd Quartile

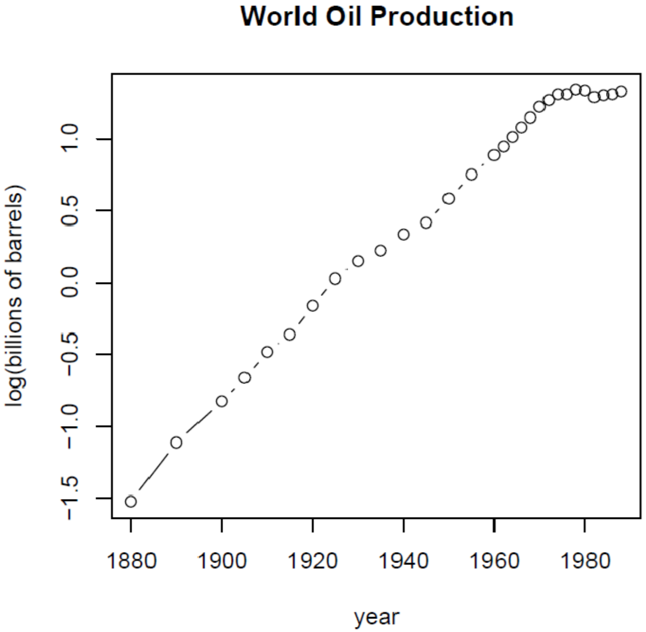
Logarithm Scale

Changing limits and base of y-scale highlights different aspects…

if y = e^x, then log(y) = x => becomes linear in x
if y = b^x, then log(y) = log(b)*x



World Oil Production

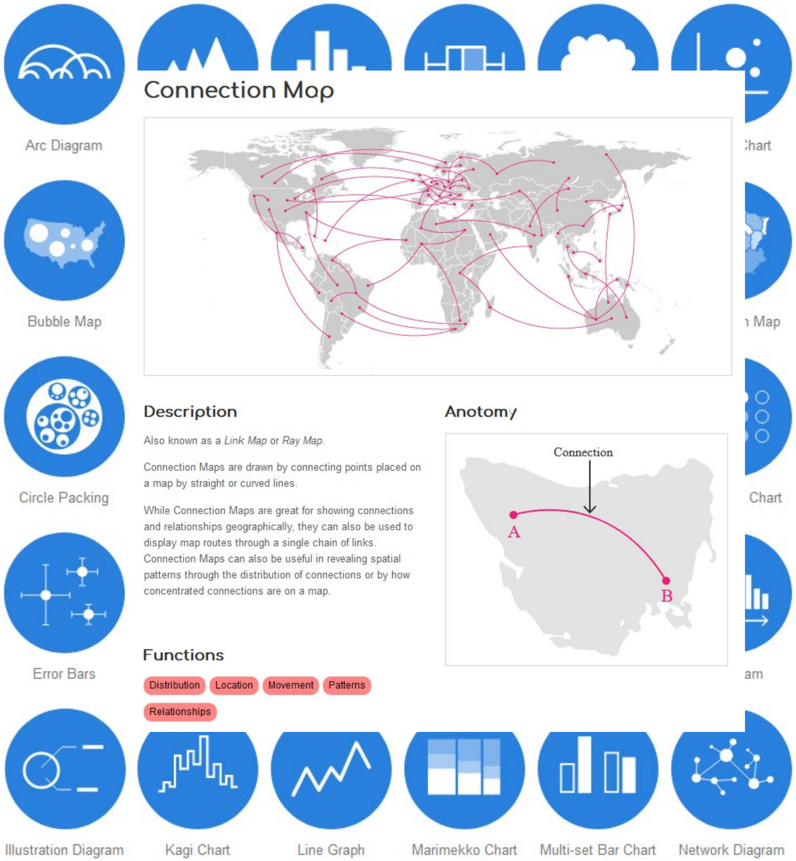


World Oil Production

…log-scale emphasizes relative changes in smaller quantities

More Visualization Resources

datavizcatalogue.com



Arc Diagram

Bubble Map

Circle Packing

Error Bars

hart

n Map

Chart

am

Connection Map

Description

Also known as a *Link Map* or *Ray Map*.

Connection Maps are drawn by connecting points placed on a map by straight or curved lines.

While Connection Maps are great for showing connections and relationships geographically, they can also be used to display map routes through a single chain of links. Connection Maps can also be useful in revealing spatial patterns through the distribution of connections or by how concentrated connections are on a map.

Anatomy

Functions

Distribution | Location | Movement | Patterns | Relationships

Illustration Diagram

Kagi Chart

Line Graph

Marimekko Chart

Multi-set Bar Chart

Network Diagram

matplotlib.org

scikit-learn.org

Next lecture

- Readings this & next lecture: WJ Chap. 1, 2

- We will have a quiz next class (1/27)
- The quiz can be done in pairs

Backup

Measuring Location

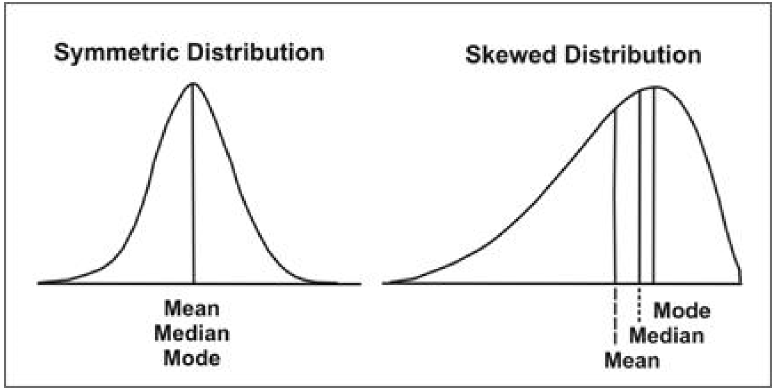
Three common measures of the distribution location…

Mean Average (expected value) of the data distribution

Median Midpoint – 50% of the probability is below and 50% above

Mode Value of highest probability (mass or density)

E.g., [1,2,3] vs [0,10,11] compute mean and median



…align with symmetric distributions, but diverge with asymmetry

Median

For data $x_1, x_2, \ldots, x_N$ sort the data,

$$x_{(1)}, x_{(2)}, \ldots, x_{(n)}$$

- Notation $x_{(i)}$ means the i-th *lowest* value, e.g. $x_{(i-1)} \leq x_{(i)} \leq x_{(i+1)}$
- $x_{(1)}, x_{(2)}, \ldots, x_{(n)}$ are called *order statistics*

not summary info, but rather a transformation

If n is **odd** then find the middle datapoint,

$$\text{median}(x_1, \ldots, x_n) = x_{((n+1)/2)}$$

If n is **even** then average between both middle datapoints,

$$\text{median}(x_1, \ldots, x_n) = \frac{1}{2}\left(x_{(n/2)} + x_{(n/2+1)}\right)$$

Sample Mean

For any real-valued function h(x) we can compute the mean as,

$$\overline{h(x)} = \frac{1}{N}\sum_{i=1}^{N} h(x_i)$$

Note $\overline{h(x)} \neq h(\bar{x})$ in general.

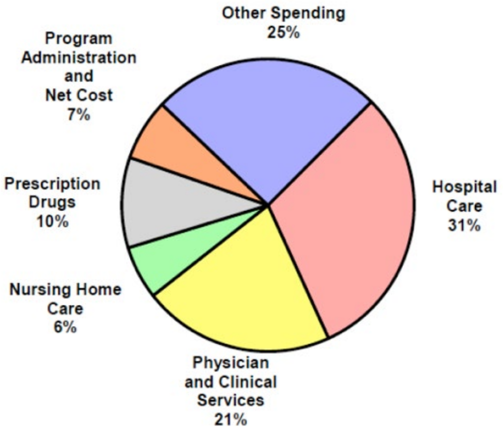
Example Compute the average of the square of values,

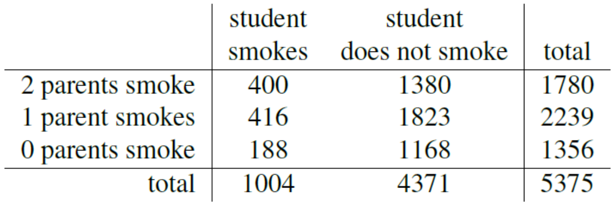
{ 1, 2, 3, 4, 5, 5, 6 }

$$\overline{x^2} = \frac{1}{7}(1 + 2^2 + 3^3 + 4^2 + 2(5^2) + 6^2) \approx 16.57$$

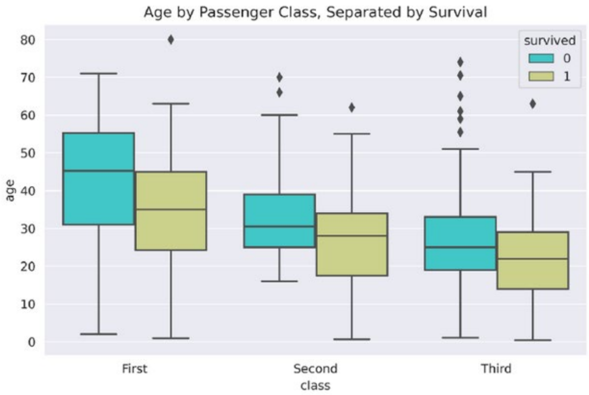
$(\bar{x})^2 \approx 13.80$

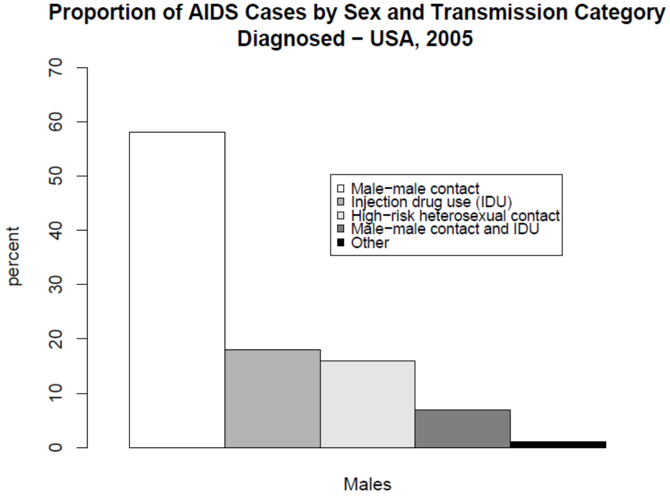
Visualizing Categorical Variables

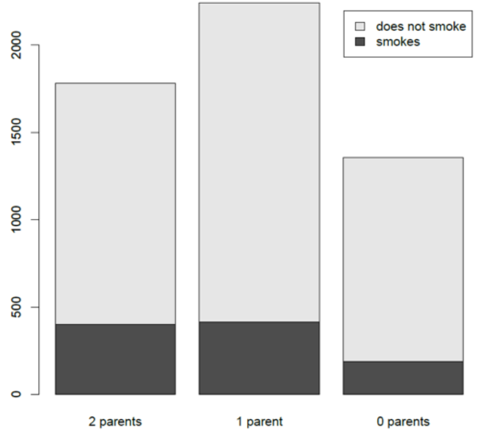




	student smokes	student does not smoke	total
2 parents smoke	400	1380	1780
1 parent smokes	416	1823	2239
0 parents smoke	188	1168	1356
total	1004	4371	5375

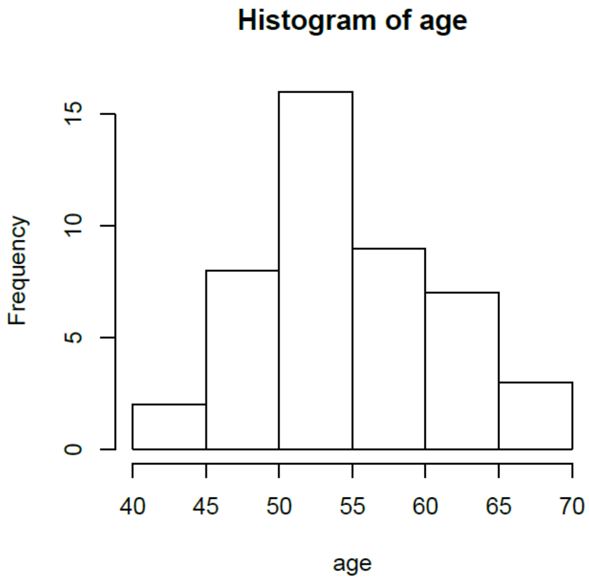


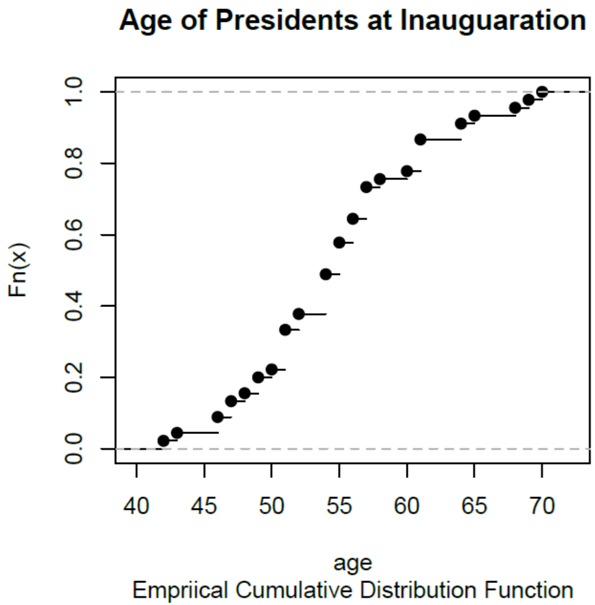




Histogram

Empirical approximation of (quantitative) data generating distribution





Empirical CDF for each x gives P(X<x),

$$F_n(x) = \frac{1}{n} \#(\text{observations less than or equal to x})$$

Sample Mean

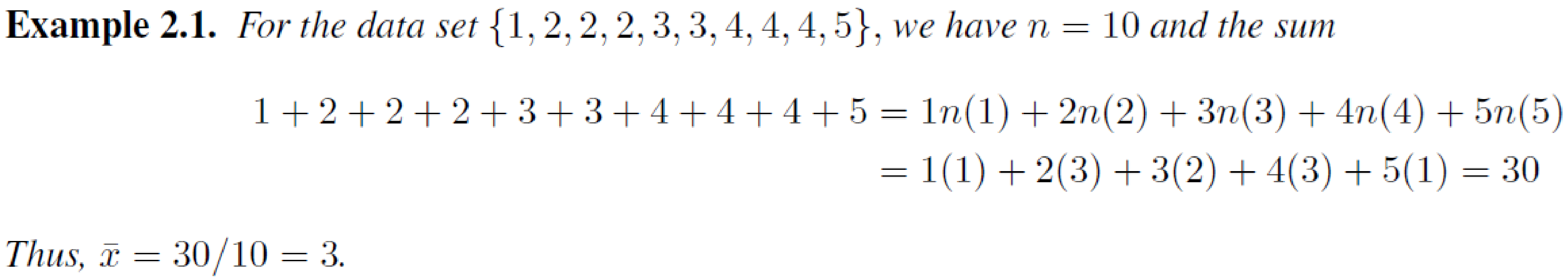
Empirical estimate of the true mean of the data distribution,

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Alternative definition: if the value x occurs n(x) times in the data then,

$$\bar{x} = \frac{1}{N}\underbrace{\sum_{x}}_{\text{for the unique values of } \{x_1, \dots, x_N\}} xn(x) = \sum_{x} xp(x) \quad \text{where} \quad \underbrace{p(x) = \frac{n(x)}{N}}_{\text{Empirical Distribution}}$$

Sample Mean



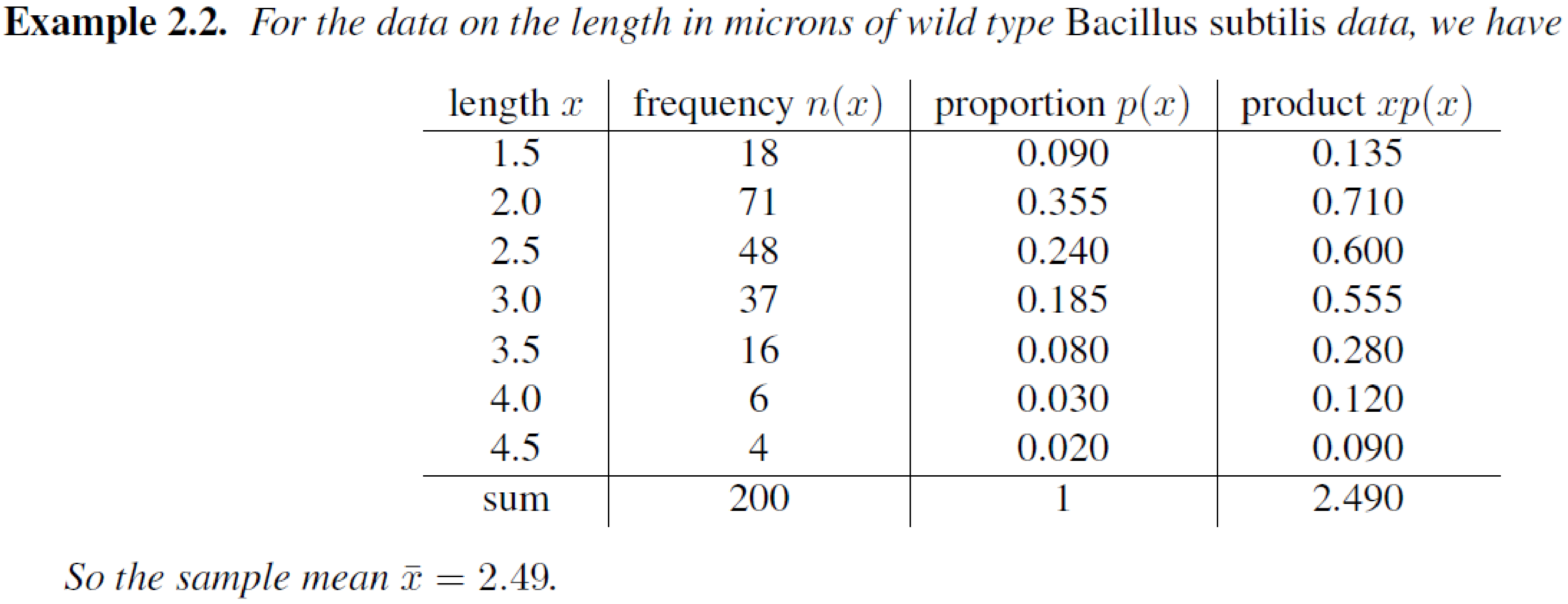
Example 2.1. *For the data set* $\{1, 2, 2, 2, 3, 3, 4, 4, 4, 5\}$, *we have* $n = 10$ *and the sum*

$$\begin{aligned} 1+2+2+2+3+3+4+4+4+5 &= 1n(1)+2n(2)+3n(3)+4n(4)+5n(5) \\ &= 1(1)+2(3)+3(2)+4(3)+5(1) = 30 \end{aligned}$$

Thus, $\bar{x} = 30/10 = 3$.

Sample Mean

↓
(bacterium)



Example 2.2. *For the data on the length in microns of wild type* Bacillus subtilis *data, we have*

length x	frequency $n(x)$	proportion $p(x)$	product $xp(x)$
1.5	18	0.090	0.135
2.0	71	0.355	0.710
2.5	48	0.240	0.600
3.0	37	0.185	0.555
3.5	16	0.080	0.280
4.0	6	0.030	0.120
4.5	4	0.020	0.090
sum	200	1	2.490

So the sample mean $\bar{x} = 2.49$.

Weighted Mean

In some cases we may weigh data differently,

$$\sum_{i=1}^{N} w_i x_i \quad \text{where} \quad \sum_{i=1}^{N} w_i = 1 \qquad 0 \leq w_i \text{ for } i = 1, \ldots, N$$

For example, grades in a class:

$$\text{Grade} = 0.2 \cdot x_{\text{midterm}} + 0.2 \cdot x_{\text{final}} + 0.6 \cdot x_{\text{homework}}$$

Grading Breakdown (example)

- Homework: 60%
- Midterm: 20%
- Final: 20%

Measuring Spread

We have seen estimates of spread via the sample variance,

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 \qquad s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

Biased **Unbiased**

But you might be interested in more detailed information about the spread.

For example, fraction of people with heights <= 5 feet