



CSC380: Principles of Data Science

Data Analysis, Collection, and Visualization 1

1

1

Pandas

3

1

Pandas

4

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

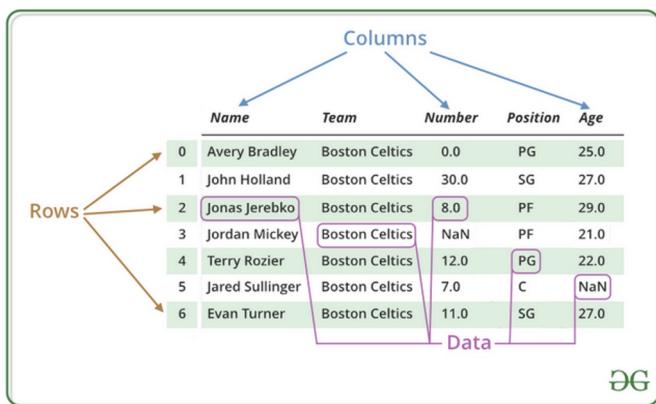
https://pandas.pydata.org/docs/getting_started/install.html

4

DataFrame

5

Primary data structure : Essentially a table



Q: how is it different from numpy array?

- Numpy arrays are more efficient
- Pandas dataframes are more flexible

5

DataFrame Example

6

Create and print an entire DataFrame

```
# 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 |

6

DataFrame Example

7

Can create named columns using dictionary

```
import pandas as pd

# initialise 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

7

DataFrame : Selecting Columns

8

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!

8

DataFrame : Selecting Columns

9

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

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).

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

| | Name |
|---|------|
| 0 | tom |
| 1 | nick |

```
[37]: df['Name']
```

still a DataFrame

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

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

essentially, a 'named' array

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

9

DataFrame : Selecting Rows

10

```

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)

```

2nd and 3rd row!

Output

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

(still a DataFrame)

1:2 includes 2! annoying! this is not python standard!!!

10

DataFrame : Selecting Rows

11

```

[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

11

Data Structure Conversions

13

Working with DataFrames outside of Pandas can be tricky,

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

Q: does it return a DataFrame object or Series object?

```
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
```

We can easily convert to built-in types, for example to a list.

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

13

Data Structure Conversions

14

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(): defined on Index, Series, and DataFrame objects

14

Summary Statistics

15

Easily compute summary statistics on data

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

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

| | |
|-----|----|
| 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.
```

Can also count occurrences of unique values,

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

15

Summary Statistics

16

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

| | Name | Age | Height |
|---|------|-----|--------|
| 0 | tom | 10 | 6.2 |
| 1 | nick | 20 | 5.5 |

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

| | 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

16

More on Pandas

17

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’

17

More on Pandas

18



Search the docs ...

Doing it by yourself helps a lot!

[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?](#)

18

Data Visualization

19

Outline

20

- Data Visualization
- Data Summarization
- Data Collection and Sampling

20

Data Analysis, Exploration, and Visualization

21

```

141 137 131 131 132 130 129 129 121 133 130 129 129 129 129 126 125 122 130
138 136 134 134 135 133 131 129 132 139 133 128 130 128 127 127 129 131
135 138 134 133 133 133 132 130 131 131 129 134 134 130 131 131 132 132 133
133 134 134 132 131 130 130 131 131 129 134 134 130 131 131 137 134 134
134 134 134 134 134 133 132 130 131 131 129 134 134 130 131 131 132 132 133
137 135 136 138 137 135 135 135 135 135 135 136 136 138 137 137 136 138 138 142
139 135 136 136 136 136 136 136 136 136 136 136 136 138 138 138 138 138 138 142
136 137 137 138 141 143 142 144 140 143 142 137 137 137 139 137 135 136
137 138 136 136 140 141 143 140 144 143 139 139 140 137 137 137 139 137 139
137 139 137 136 136 136 137 140 143 146 144 143 143 140 141 142 142 143 143
137 140 140 141 139 138 136 136 137 143 144 142 139 142 142 148 149 147 146
140 144 140 141 142 143 143 140 141 142 143 140 139 140 141 142 146 146 147
145 147 145 145 145 145 145 145 145 145 145 145 145 145 145 145 145 145 145 149
146 148 147 144 143 141 140 143 137 139 142 145 146 146 145 145 145 146 147
145 147 146 143 142 140 140 143 138 140 143 143 143 143 143 143 143 148 142
145 148 144 143 143 142 142 142 145 146 145 145 146 145 145 145 145 150 144
144 143 142 142 143 142 141 141 144 143 144 143 144 144 148 144 142 147 145
146 145 144 143 143 142 142 141 144 144 144 144 144 144 144 144 144 143 146
145 145 144 144 143 143 142 142 141 144 144 144 144 144 144 144 144 143 146
150 149 147 145 142 142 143 143 145 147 143 147 166 175 164 151 152
150 150 149 147 145 145 145 147 148 143 142 151 161 160 149 150
152 152 152 150 149 150 150 149 151 151 150 147 149 152 153 147 151
152 159 153 152 151 151 150 152 152 156 155 149 155 153 152
152 152 152 152 152 152 152 152 152 152 152 152 152 152 152 152 152 154
153 155 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153 154
153 153 153 153 153 153 153 154 154 154 154 154 154 154 154 154 154 155
153 153 152 153 153 154 154 154 153 154 154 154 154 154 154 154 154 157
153 152 152 152 152 154 155 155 155 155 155 155 155 154 152 152 154 159

```

Encoding



Iterate



Visual Perception



Understanding



21

Data visualization in Python...

matplotlib²²

```

import matplotlib.pyplot as plt
import numpy as np

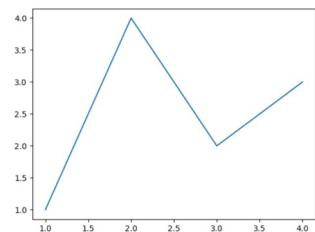
```

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.

```



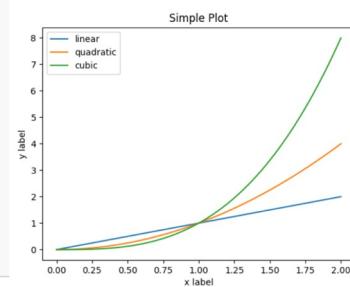
A more complicated plot...

```

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.

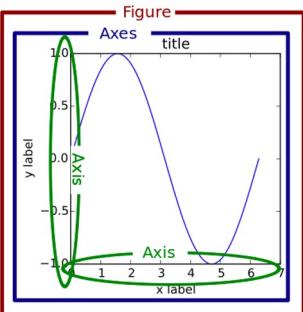
```



22

matplotlib²³

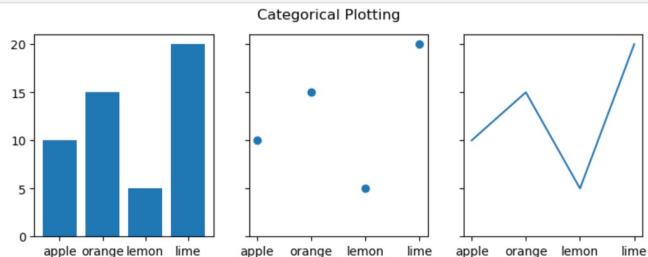
Axes vs Axis



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

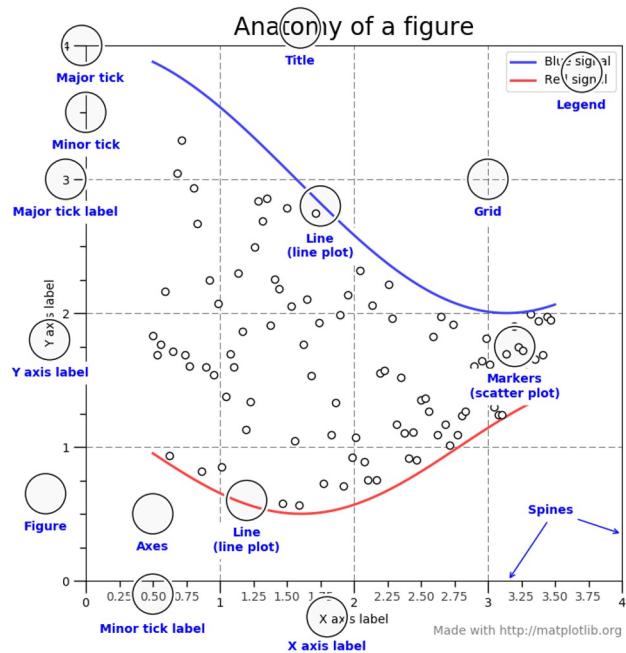
```
data = {'apple': 10, 'orange': 15, 'lemon': 5, 'lime': 20}
names = list(data.keys())
values = list(data.values())

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



23

matplotlib²⁴



components of a Matplotlib figure

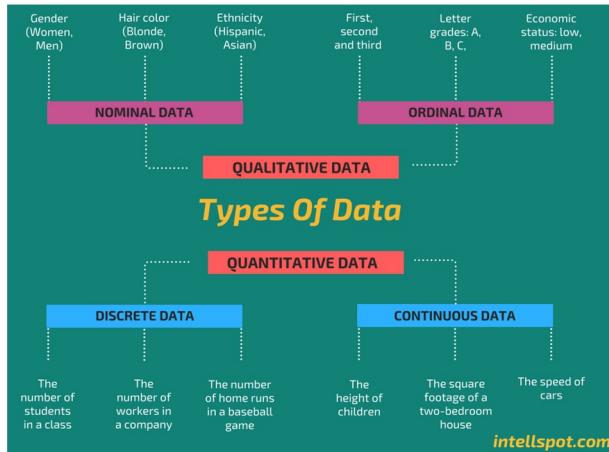
Documentation + tutorials:
<https://matplotlib.org/>

24

Types of Data

25

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



Qualitative or categorical : can partition data into classes

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

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

25

Categorical Variables

26

Examples

- Roll of a die: 1,2,3,4,5 or 6
 - 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.

Numerical data can be categorical or quantitative depending on context

Numerical data can be categorical or quantitative depending on context

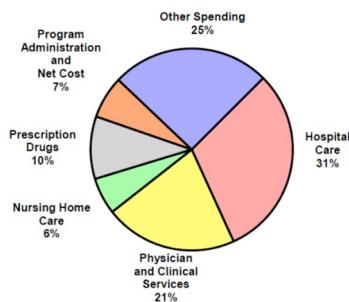
Conversion: Quantitative data can be converted to categorical by defining ranges:

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

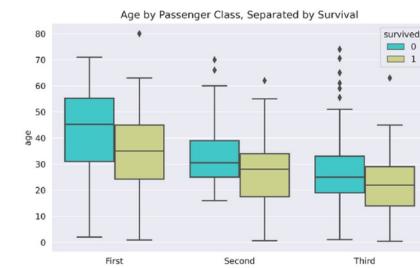
26

Visualizing Categorical Variables

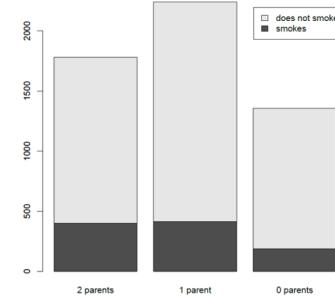
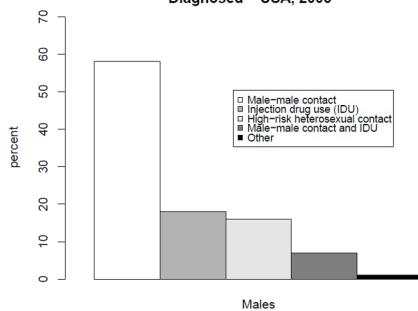
27



| | 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 |



Proportion of AIDS Cases by Sex and Transmission Category
Diagnosed – USA, 2005



27

Pie Chart

28

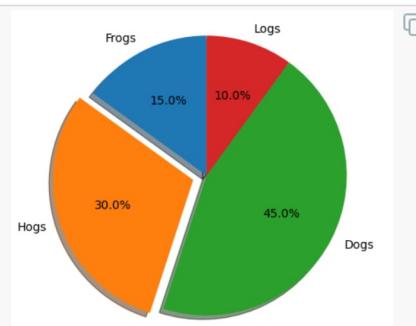
Circular chart divided into sectors, illustrating relative magnitudes in frequencies or percentage. In a pie chart, the area is proportional to the quantity it represents.

```
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.

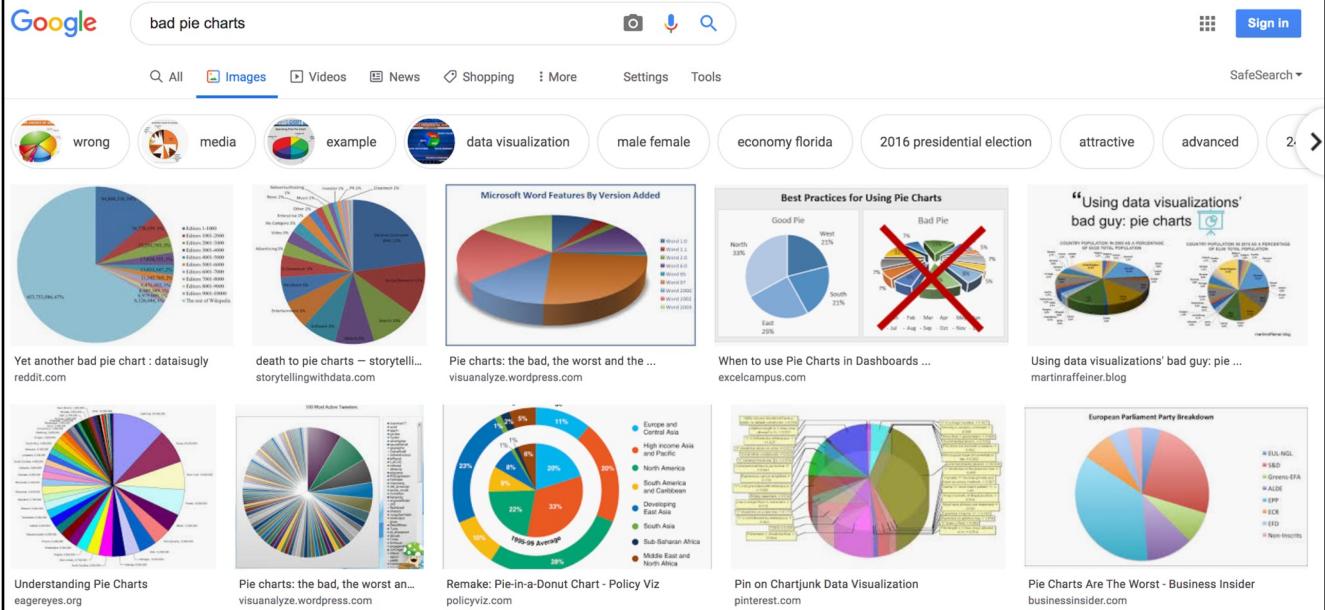
plt.show()
```



28

13

Maybe the biggest problem with pie charts is that they have been so often done poorly...



29

Bar Chart

30

We perceive differences in height / length better than area...

`plt.bar()`

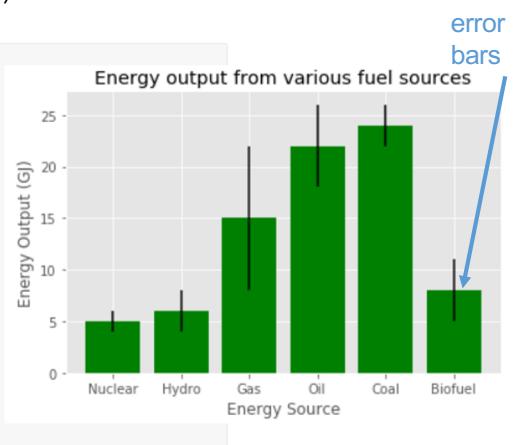
```
x = ['Nuclear', 'Hydro', 'Gas', 'Oil', 'Coal', 'Biofuel']
energy = [5, 6, 15, 22, 24, 8]
variance = [1, 2, 7, 4, 2, 3]

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()
```



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

30

Bar Chart

31

Horizontal version.

```
plt.bart()
```

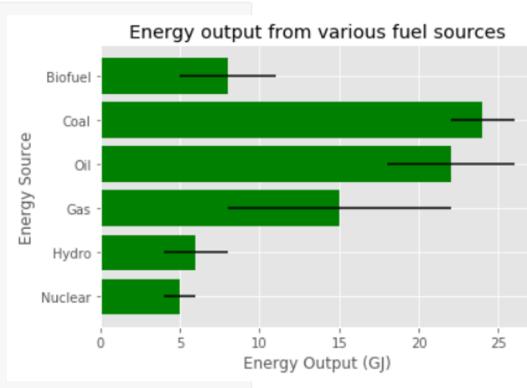
```
x = ['Nuclear', 'Hydro', 'Gas', 'Oil', 'Coal', 'Biofuel']
energy = [5, 6, 15, 22, 24, 8]
variance = [1, 2, 7, 4, 2, 3]

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

plt.bart(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/>]

31

Bar Chart

32

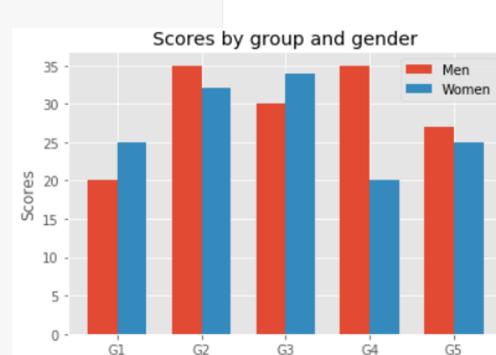
Multiple groups of bars...

```
import numpy as np

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

ind = np.arange(N) // [1,2,3,4,5]
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')

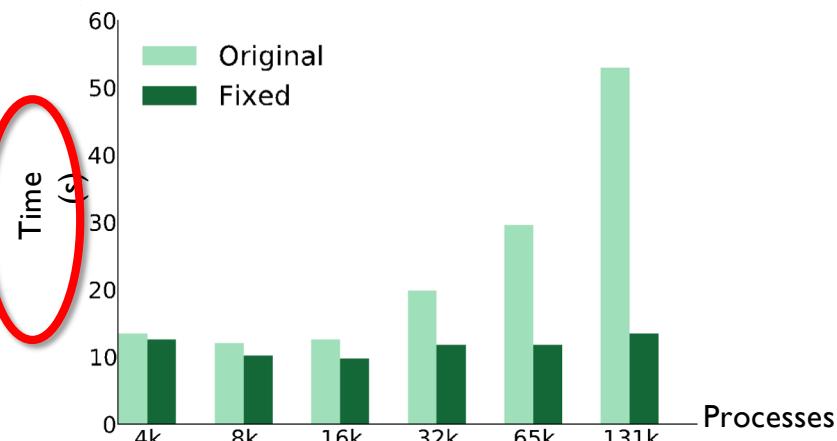
plt.xticks(ind + width / 2, ('G1', 'G2', 'G3', 'G4', 'G5'))
plt.legend(loc='best')
plt.show()
```



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

32

Labels on the y-axis need not be vertical

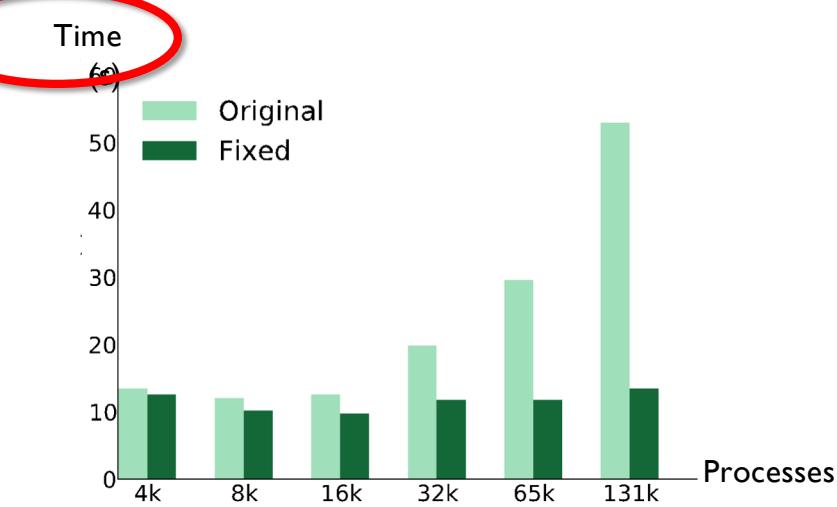


[Source: Kate Isaacs]

33

33

Labels on the y-axis need not be vertical



[Source: Kate Isaacs]

34

34

Stacked Bar Chart

35

```

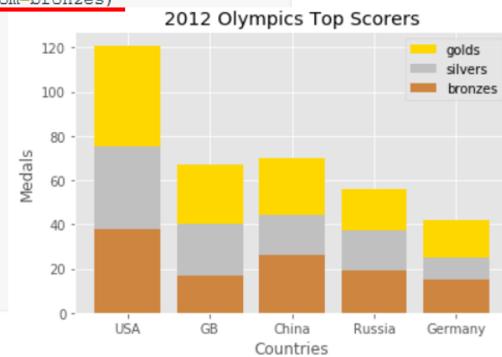
countries = ['USA', 'GB', 'China', 'Russia', 'Germany']
bronzes = np.array([38, 17, 26, 19, 15])
silvers = np.array([37, 23, 18, 18, 10])
golds = np.array([46, 27, 26, 19, 17])
ind = [x for x, _ in enumerate(countries)]

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/>]

35

Parallel Coordinates

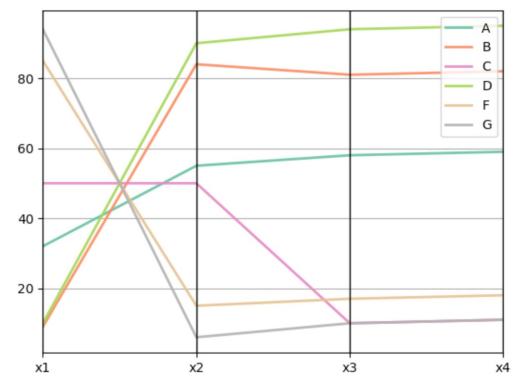
- How to visualize high-dim data (for example, points in 4D) say $\{p_1 \dots p_n\}$
- Idea: Draw 4, parallel vertical segments, for x_1 , for x_2 .. for x_4 .
- Each point becomes a polyline with a specific color
- The path of p_i crosses the j-axis at height p_{ij} (that is, the j'th coordinate of the point.)

| | x_1 | x_2 | x_3 | x_4 | Type |
|----|-------|-------|-------|-------|------|
| 32 | 55 | 58 | 59 | A | |
| 9 | 84 | 81 | 82 | B | |
| 50 | 50 | 10 | 11 | C | |
| 10 | 90 | 94 | 95 | D | |
| 85 | 15 | 17 | 18 | F | |
| 94 | 6 | 10 | 11 | G | |

```

from pandas.plotting import parallel_coordinates
df = pd.read_csv('~/Dropbox/Book1.csv')
print(df)
plt.figure()
parallel_coordinates(df, 'Type', colormap=plt.get_cmap("Set2"), alpha=0.5, linewidth=2)

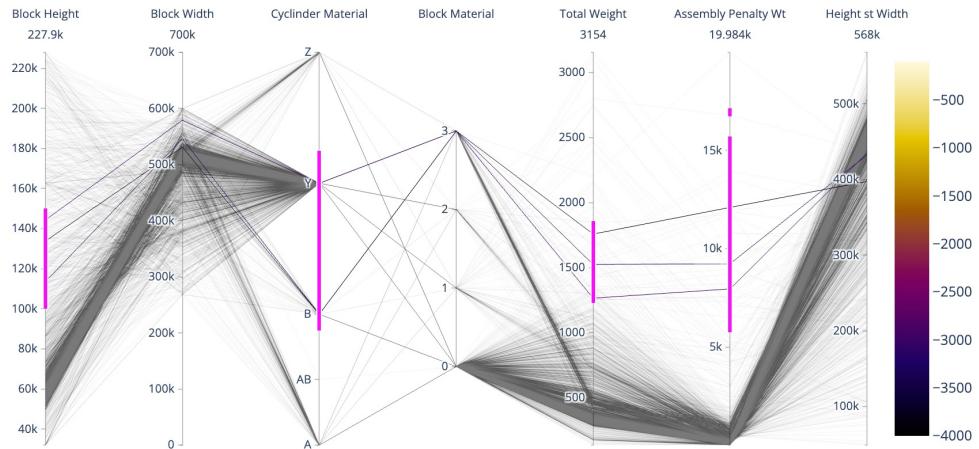
```



36

Parallel coordinates – cont

- import plotly.graph_objects as go



An interactive^{32k} system – could change the order of coordinates³⁴, and/or perform selection or a subset of points