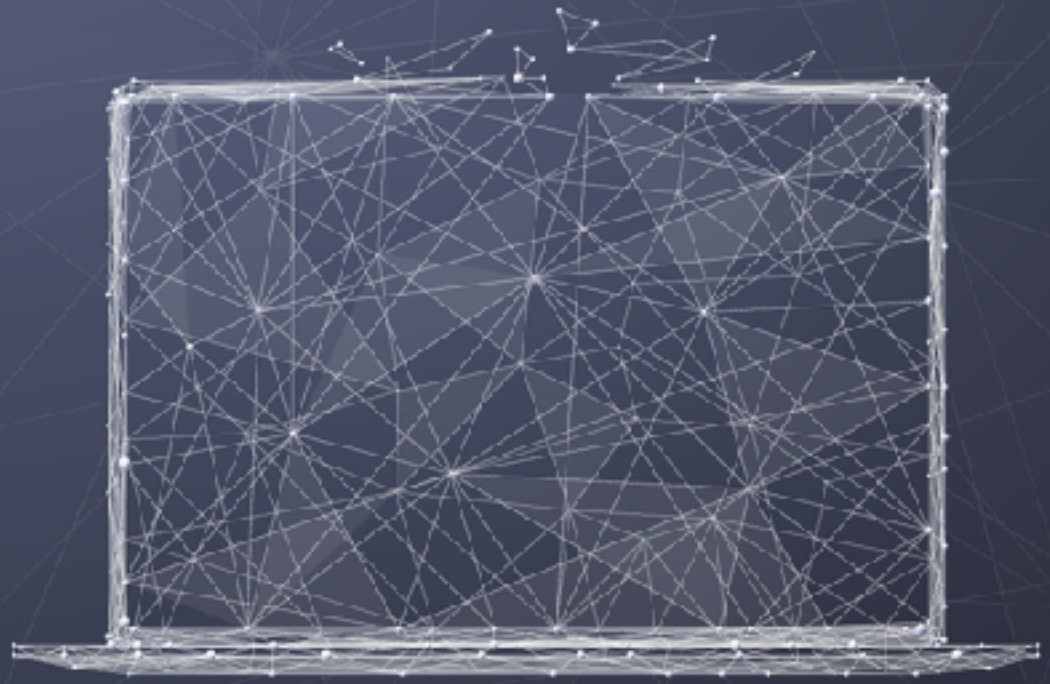
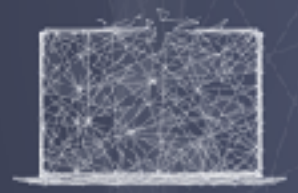
Data Science
Data Engineering I
Data processing for analysis

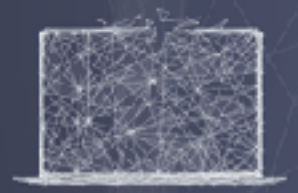


PURDUE UNIVERSITY | College of Science



Processing data for analysis

- Sampling
- Attribute transformations
- Feature construction and selection

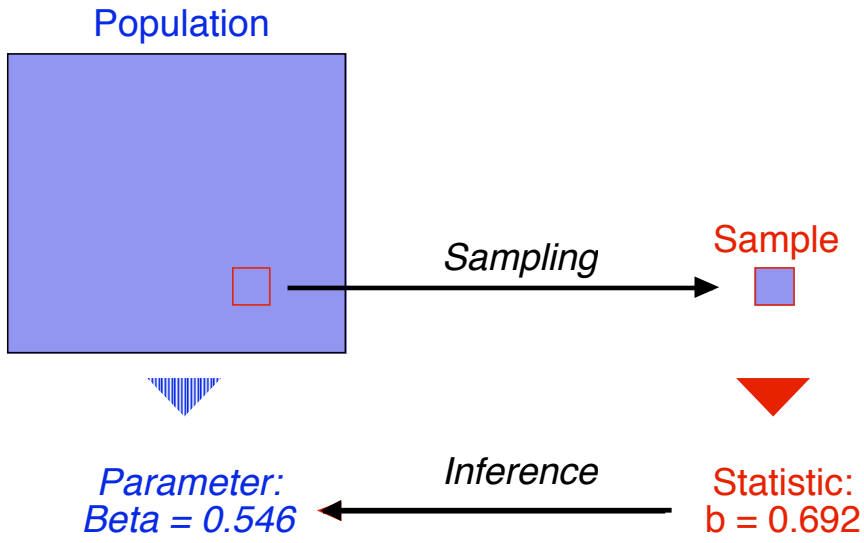


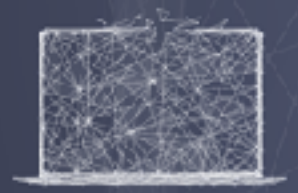
Sampling

- Sampling is main technique employed for data selection
  - It is often used for both preliminary investigations of the data and final data analysis
- Reasons to sample
  - Obtaining the entire set of data of interest is too expensive or time consuming
  - Processing the entire set of data of interest is too expensive or time consuming
  - Note: Even if you use an entire dataset for analysis, you should be aware of the sampling method that was used to gather the dataset

Analysis with samples

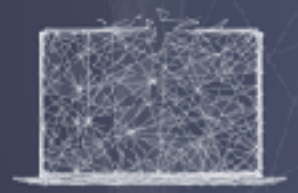
- The key principle for effective sampling is the following:
- Using a sample will work almost as well as using the entire data set, if the sample is representative of the overall data
- A sample is representative if it has approximately the same property (of interest) as the target population





Populations and samples

- Elementary units:
  - Entities (e.g., persons, objects, events) that meet a set of specified criteria
  - Example: All people who've purchased something from Walmart in the past month
- Population:
  - Aggregate set of elementary units (i.e, all items of interest)
- Sampling:
  - Sub-group of the population
  - Serves as a reference group for estimating characteristics about the population and drawing conclusions



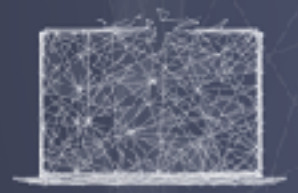
SAMPLING METHODS

Probability Sampling

1. Simple random sampling
2. Systematic sampling
3. Stratified sampling
4. Cluster sampling
5. Multi-stage sampling

Non-probability Sampling

1. Convenience sampling
2. Quota sampling
3. Snowball sampling
4. Judgement sampling



Types of probability sampling

- Simple random sampling
  - There is an equal probability of selecting any particular item
- Sampling without replacement
  - As each item is selected, it is removed from the population
- Sampling with replacement
  - Items are not removed from the population as they are selected for the sample; the same item can be picked up more than once
- Stratified sampling
  - Split the data into several partitions; then draw random samples from each partition

Pandas: sample without replacement

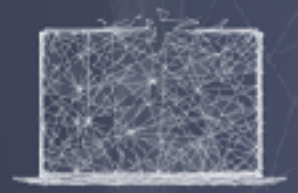
```
# get a random sample of ten rows
data.sample(10)
```

	Index	Year	Age	Name	Movie
29	30	1957	28	Joanne Woodward	The Three Faces of Eve
24	25	1952	54	Shirley Booth	Come Back,Little Sheba
61	62	1989	26	Jodie Foster	The Accused
46	47	1974	37	Glenda Jackson	A Touch of Class
37	38	1965	25	Julie Christie	Darling
28	29	1956	41	Ingrid Bergman	Anastasia
65	66	1993	33	Emma Thompson	Howards End
49	50	1977	36	Faye Dunaway	Network
4	5	1932	32	Helen Hayes	The Sin of Madelon Claudet
38	39	1966	35	Elizabeth Taylor	Who's Afraid of Virginia Woolf?

Pandas: sample with replacement

```
# get a 10% random sample, with replacement
data.sample(frac=0.1,replace=True)
```

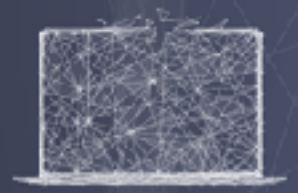
	Index	Year	Age	Name	Movie
38	39	1966	35	Elizabeth Taylor	Who's Afraid of Virginia Woolf?
17	18	1945	40	Joan Crawford	Mildred Pierce
12	13	1940	29	Ginger Rogers	Kitty Foyle
11	12	1939	26	Vivien Leigh	Gone with the Wind
9	10	1937	28	Luise Rainer	The Good Earth
1	2	1929	37	Mary Pickford	Coquette
34	35	1962	31	Anne Bancroft	The Miracle Worker
12	13	1940	29	Ginger Rogers	Kitty Foyle
58	59	1986	61	Geraldine Page	The Trip to Bountiful



Pandas: sampling w/repeatability

```
# get a 7.5% random sample, without replacement, with specified random seed
data.sample(frac=0.075, random_state=999)
```

	Index	Year	Age	Name	Movie
85	86	2013	22	Jennifer Lawrence	Silver Linings Playbook
18	19	1946	30	Olivia de Havilland	To Each His Own
45	46	1973	27	Liza Minnelli	Cabaret
10	11	1938	30	Bette Davis	Jezebel
35	36	1963	31	Patricia Neal	Hud
58	59	1986	61	Geraldine Page	The Trip to Bountiful
34	35	1962	31	Anne Bancroft	The Miracle Worker

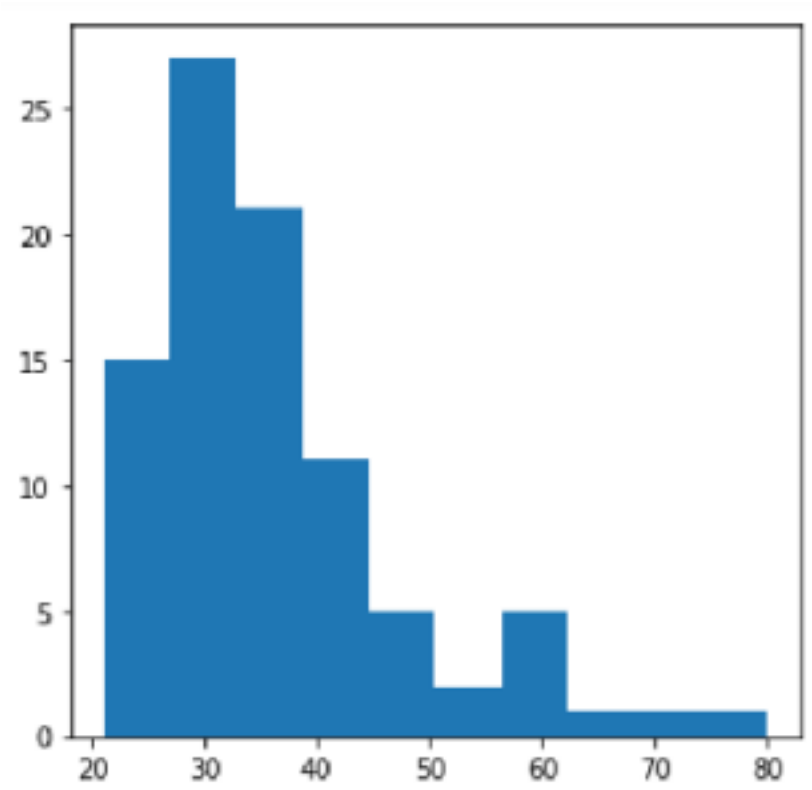


Attribute transformations

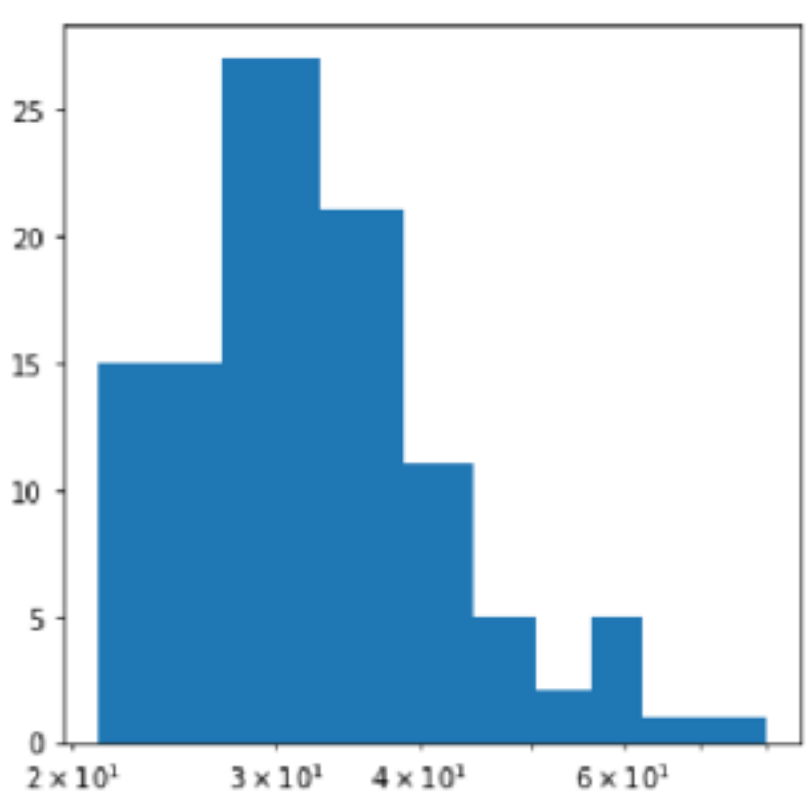
- Reasons to transform variables before analysis
  - Different scale/range may be more convenient for interpretation of patterns
  - Visualization can be more effective if skew is reduced
  - Many machine learning methods work better if the variables have roughly the same scale and/or are normally distributed

Transform during visualization

```
fig = plt.figure()
ax = plt.axes()
ax.hist(data.Age)
```

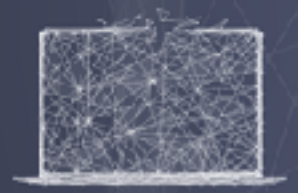



```
fig = plt.figure()
ax = plt.axes()
ax.hist(data.Age)
plt.xscale('log')
```



Reducing skewness

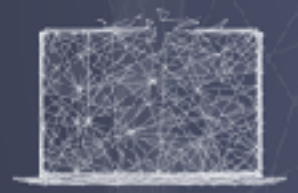
- A distribution that is nearly symmetric is often easier to handle and interpret than a skewed distribution
  - To reduce right skewness, take roots or logarithms or reciprocals
  - To reduce left skewness, take squares or cubes or higher powers



Transform variables in data frame

```
# pandas transform function is like map
data['logAge'] = data.Age.transform(func='log')
data.head()
```

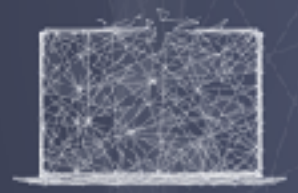
	Index	Year	Age	Name	Movie	logAge
0	1	1928	22	Janet Gaynor	Seventh Heaven,Street Angel and Sunrise: A Son...	3.091042
1	2	1929	37	Mary Pickford	Coquette	3.610918
2	3	1930	28	Norma Shearer	The Divorcee	3.332205
3	4	1931	63	Marie Dressler	Min and Bill	4.143135
4	5	1932	32	Helen Hayes	The Sin of Madelon Claudet	3.465736



Standardize variables in data frame

```
# standardize Age values to have mean=0 and std=1
avgAge = data.Age.mean()
stdAge = data.Age.std()
data['stdAge'] = data.Age.map(lambda x: (x - avgAge)/stdAge)
data.head()
```

	Index	Year	Age	Name	Movie	stdAge
0	1	1928	22	Janet Gaynor	Seventh Heaven,Street Angel and Sunrise: A Son...	-1.202496
1	2	1929	37	Mary Pickford	Coquette	0.074618
2	3	1930	28	Norma Shearer	The Divorcee	-0.691651
3	4	1931	63	Marie Dressler	Min and Bill	2.288282
4	5	1932	32	Helen Hayes	The Sin of Madelon Claudet	-0.351087

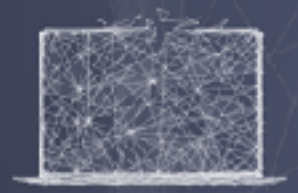


Scale variables in data frame

```
# scale Age values to have range [0,1]
ageMin = data.Age.min()
ageRange = data.Age.max() - ageMin
data['maxNormAge'] = data.Age.transform(lambda x: (x - ageMin)/ageRange)
data.maxNormAge.describe()
```

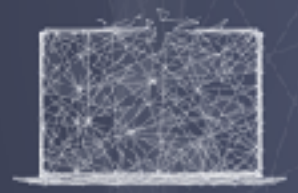
```
count    89.000000
mean      0.256332
std       0.199072
min       0.000000
25%       0.118644
50%       0.203390
75%       0.338983
max       1.000000
Name: maxNormAge, dtype: float64
```

PURDUE
UNIVERSITY



Other python transformation libraries

- There are also attribute transformation methods in scikit-learn, e.g.:
  - `sklearn.preprocessing.MinMaxScaler`
  - `sklearn.preprocessing.StandardScaler`
- These functions work on numpy arrays/matrices.
  - To use, you will need to convert your Pandas data frame to a numpy representation using `DataFrame.values`
  - Then you will need to convert the results back into a data frame, eg. `pd.DataFrame(scaled_features, index=df.index, columns=df.columns)`



Distance measures

- If objects have the same fixed set of numeric attributes, then the data objects can be thought of as points in a multi-dimensional space, where each dimension represents a distinct attribute
- Many machine learning methods use proximity between objects as a measure of similarity/dissimilarity
- Euclidean distance is the most commonly used distance measure for continuous data

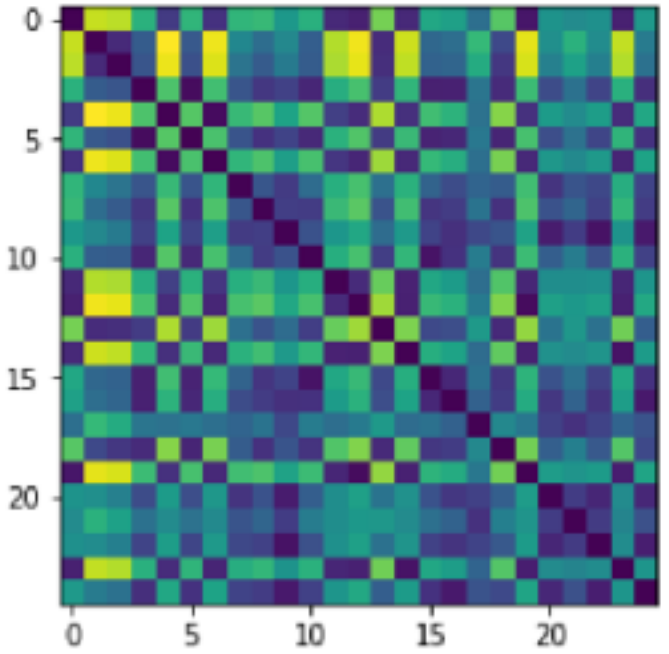
$$d_E(x, y) = \sqrt{\sum_{i=1}^{p} (x_i - y_i)^2}$$

  Here p is the number of attributes/columns

- Hamming distance or matching coefficient is often used for bit vectors and discrete data

Calculating pairwise distances

```
import numpy as np
import math
data2 = data.sample(25)
dists = np.zeros((len(data2),len(data2)))
for i in range(len(data2)):
    ivalues = data2.iloc[[i]].values[0]
    for j in range(len(data2)):
        if i != j:
            jvalues = data2.iloc[[j]].values[0]
            tmpdist = 0
            for k in range(4):
                tmpdist = tmpdist + (ivalues[k] - jvalues[k])**2
            dists[i,j] = math.sqrt(tmpdist)
plt.imshow(dists)
```

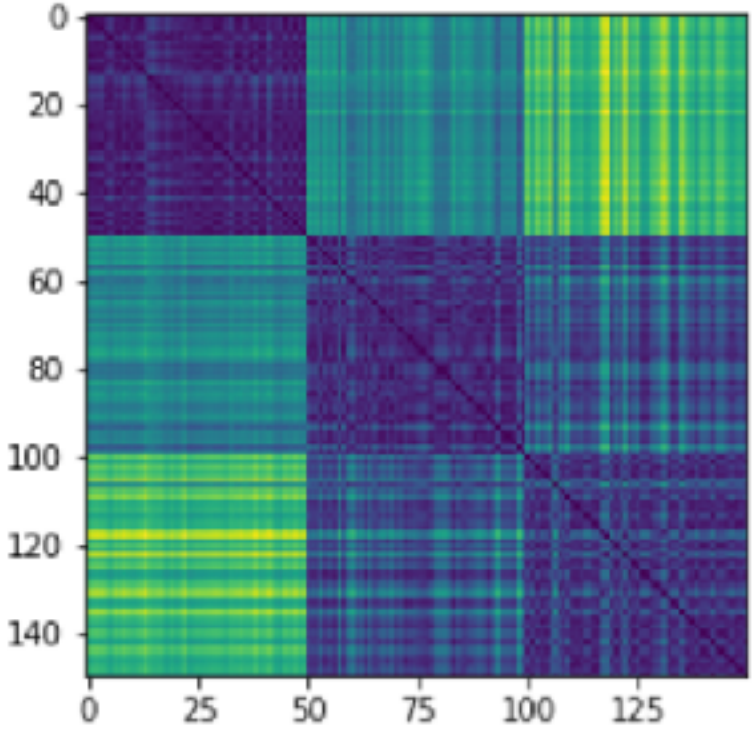


PURDUE UNIVERSITY

Calculating pairwise distances

```
from sklearn.metrics.pairwise import euclidean_distances

dists = euclidean_distances(data.iloc[:,0:4].values)
plt.imshow(dists)
```



PURDUE UNIVERSITY