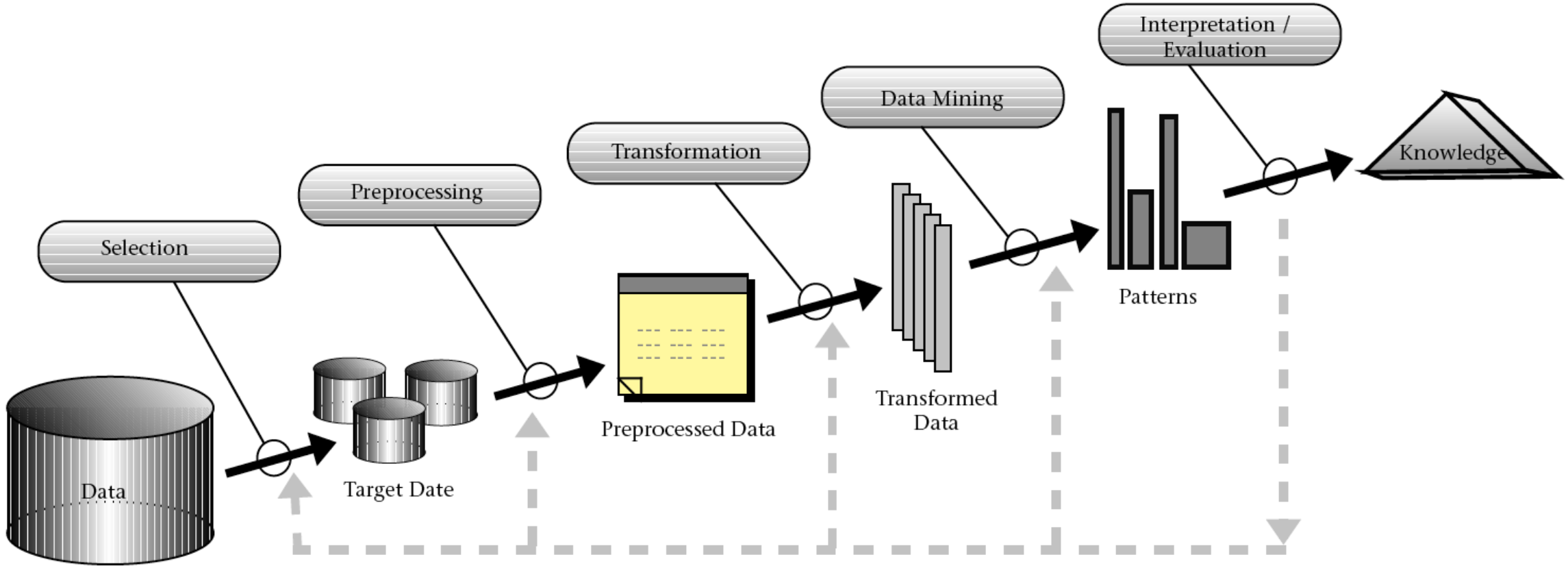
人工智能程序设计

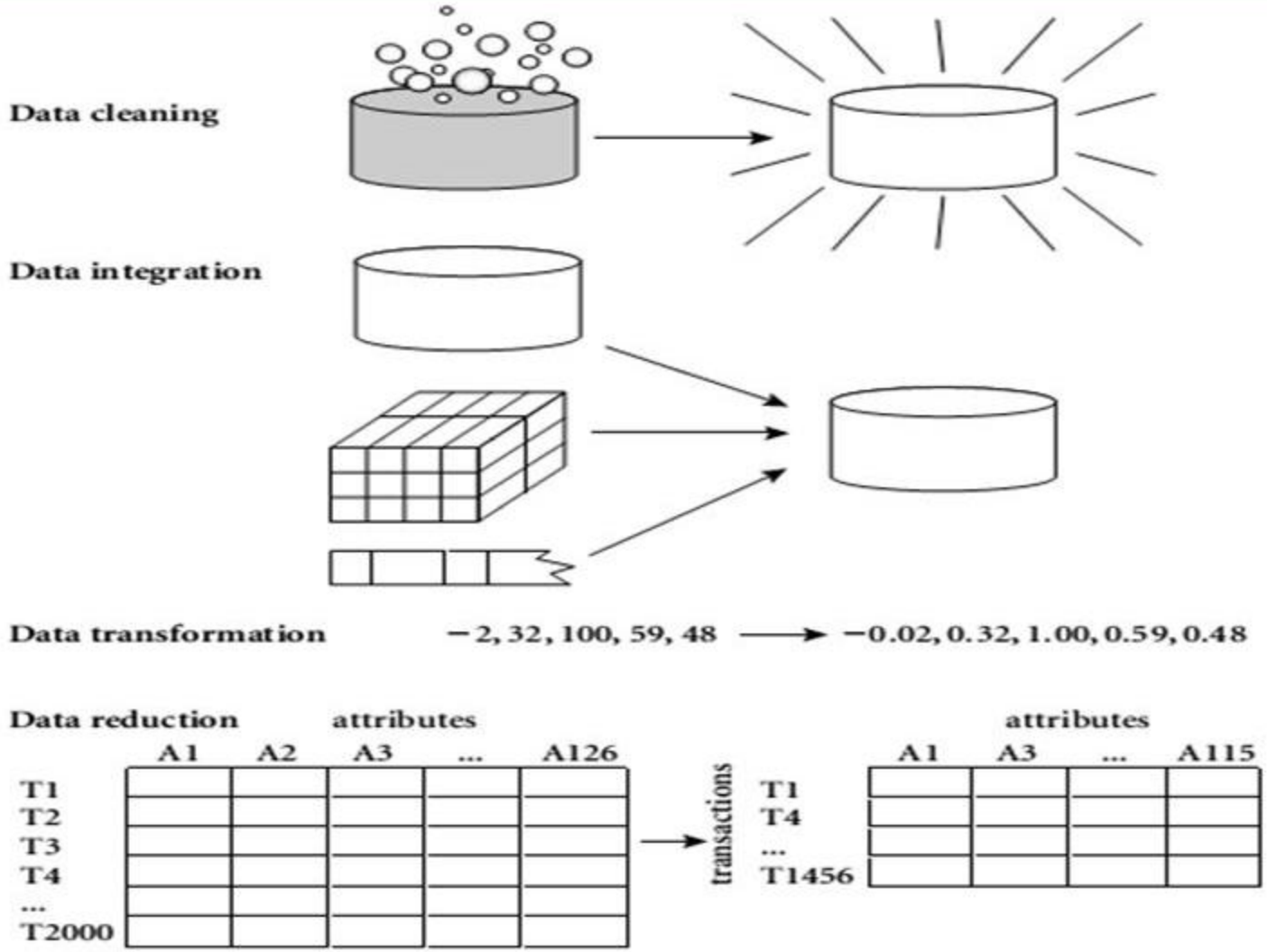
M3 人工智能基础方法

2 数据预处理

张 莉


Interpretation / Evaluation
Data Mining
Transformation
Preprocessing
Selection
Knowledge
Patterns
Transformed Data
Preprocessed Data
Target Date
Data

Forms of Data Preprocessing



数据预处理

1. 数据清洗
2. 数据集成
3. 数据变换
4. 数据规约之数值规约

人工智能程序设计

1

数据清洗·DATA CLEANING

缺失值处理

如何处理?

- 删除
- 填充

缺失值填充	固定值
	均值，中位数/众数
	上下数据
	插值函数
	最可能的值

缺失值处理

```
scores_df = pd.read_excel('scores.xlsx', index_col = 'name')
```

判断缺失值 df.isnull()

删除缺失行 df.dropna()

填充缺失行 df.fillna()

如何用均值填充?

```
scores_df.fillna(method='ffill', inplace = True)
```

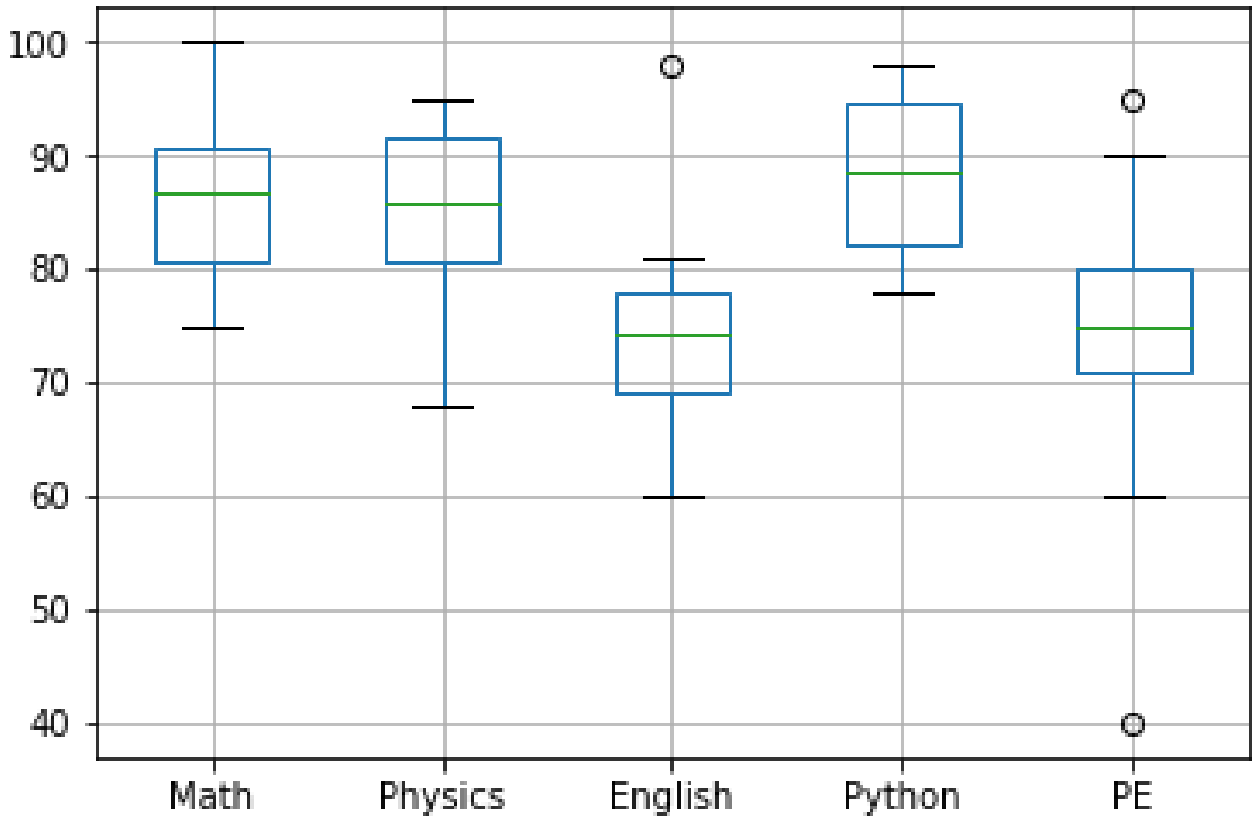
异常值处理

如何观察异常值?

- 简单统计
- 绘图
- 基于密度，最近邻和聚类等方法

如何处理?

- 删除
- 同缺失值处理
- 局部均值(分箱)
- 不处理



人工智能程序设计

2 数据集成·DATA INTEGRATION

数据集成

考虑哪些问题?

- 实体识别
- 冗余属性/记录识别
- 数据值冲突的检测与处理

id id
student_name stu_name

id id
student_name stu_name
T1: price days T2: prices
12,17,23,11 12,17,23,11

height: 170
Height: 1.7

```
>>> df1 = pd.DataFrame({'lkey': ['foo', 'bar', 'baz', 'foo'], 'value': [1, 2, 3, 5]})
>>> df2 = pd.DataFrame({'rkey': ['foo', 'bar', 'baz', 'xht'], 'value': [5, 6, 7, 8]})
>>> x = df1.merge(df2, left_on='lkey', right_on='rkey')
>>> x.drop('rkey', axis = 1)
   lkey  value_x  value_y
0  foo        1        5
1  foo        5        5
2  bar        2        6
3  baz        3        7
>>> df1.merge(df2, left_on='lkey', right_on='rkey', how = 'outer')
```

人工智能程序设计

3 数据变换·DATA TRANSFORMATION

数据变换

把数据变换成适合的形式

常见方式	规范化
	连续属性离散化
	属性构造

数据规范化

解决哪些影响？

- 量纲不同
- 数值范围差异大

规范化常用方法

- 最小-最大规范化
- z-score规范化
  (零-均值规范化)
- 小数定标规范化

```
>>> boston = datasets.load_boston()
>>> boston.data     # (503, 13)
array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.9690e+02,
        4.9800e+00], …., ]])
>>> boston.target
array([24. , 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, ..., ]])
>>>  boston_df = pd.DataFrame(boston.data[:, 4:7])
>>>  boston_df.columns = boston.feature_names[4:7]
>>> boston_df
        NOX     RM   AGE
0     0.538  6.575   65.2
1     0.469  6.421   78.9
2     0.469  7.185   61.1
3     0.458  6.998   45.8
4     0.458  7.147   54.2
…
504  0.573  6.794   89.3
505  0.573  6.030   80.8
```

最小-最大规范化

$$x' = \frac{x - min}{\max - min}$$

常见做法落在[0,1]区间

```
(df-df.min())/(df.max()-df.min())
```

	NOX	RM	AGE
0	0.314815	0.577505	0.641607
1	0.172840	0.547998	0.782698
2	0.172840	0.694386	0.599382
3	0.150206	0.658555	0.441813
4	0.150206	0.687105	0.528321
5	0.150206	0.549722	0.574665

问题：

- 若将来的数字超过min和max，会越界，需重新定义
- 若某个数很大则规范化后值相近且均接近0

最小-最大规范化

```
from sklearn import preprocessing

min_max_scaler = preprocessing.minmax_scale(df)    # [0,1]
max_abs_scaler = preprocessing.maxabs_scale(df)    # [-1,1]
```

z-score规范化

$$x' = \frac{x - \bar{x}}{\sigma}$$

```
(df-df.mean())/df.std()
```

	NOX	RM	AGE
0	-0.144075	0.413263	-0.119895
1	-0.739530	0.194082	0.366803
2	-0.739530	1.281446	-0.265549
3	-0.834458	1.015298	-0.809088
4	-0.834458	1.227362	-0.510674
5	-0.834458	0.206892	-0.350810

特征：

- 使用最多
- 处理后数据的均值为0，标准差为1

z-score规范化

```
scaler = preprocessing.scale(df)
```


```
array([[-0.14421743,  0.41367189, -0.12001342],
       [-0.74026221,  0.19427445,  0.36716642],
       [-0.74026221,  1.28271368, -0.26581176],
       ...,
       [ 0.15812412,  0.98496002,  0.79744934],
       [ 0.15812412,  0.72567214,  0.73699637],
       [ 0.15812412, -0.36276709,  0.43473151]])
```

小数定标规范化

$$x' = \frac{x}{10^{j}}$$

```
df/10**np.ceil(np.log10(df.abs().max()))
```

特征：

- 移动小数点位置，移动位数取决于属性绝对值的最大值
- 常见落在[-1, 1]之间

	NOX	RM	AGE
0	0.538	0.6575	0.652
1	0.469	0.6421	0.789
2	0.469	0.7185	0.611
3	0.458	0.6998	0.458
4	0.458	0.7147	0.542
5	0.458	0.6430	0.587

连续属性离散化

方法：

- 分箱（binning）：
  等宽法，等频法
- 聚类

```
pd.cut(df.AGE, 5, labels = range(5))
```

0	65.2
1	78.9
2	61.1
3	45.8
4	54.2
5	58.7

0	3
1	3
2	2
3	2
4	2
5	2

- 等宽法和等频法讨论

人工智能程序设计

4 数据规约之数值规约·VALUE REDUCTION

数据规约

目的：

- 对属性和数值进行规约获得一个原数据集的小的多的规约表示，但仍接近原数据的完整性，在规约后数据集上挖掘可产生近乎相同的分析结果

数据规约	属性规约：向前选择，向后删除，决策树，PCA
	数值规约：有参方法（回归法，对数线性模型），无参法（直方图，聚类，抽样）

数值规约—直方图

表现：

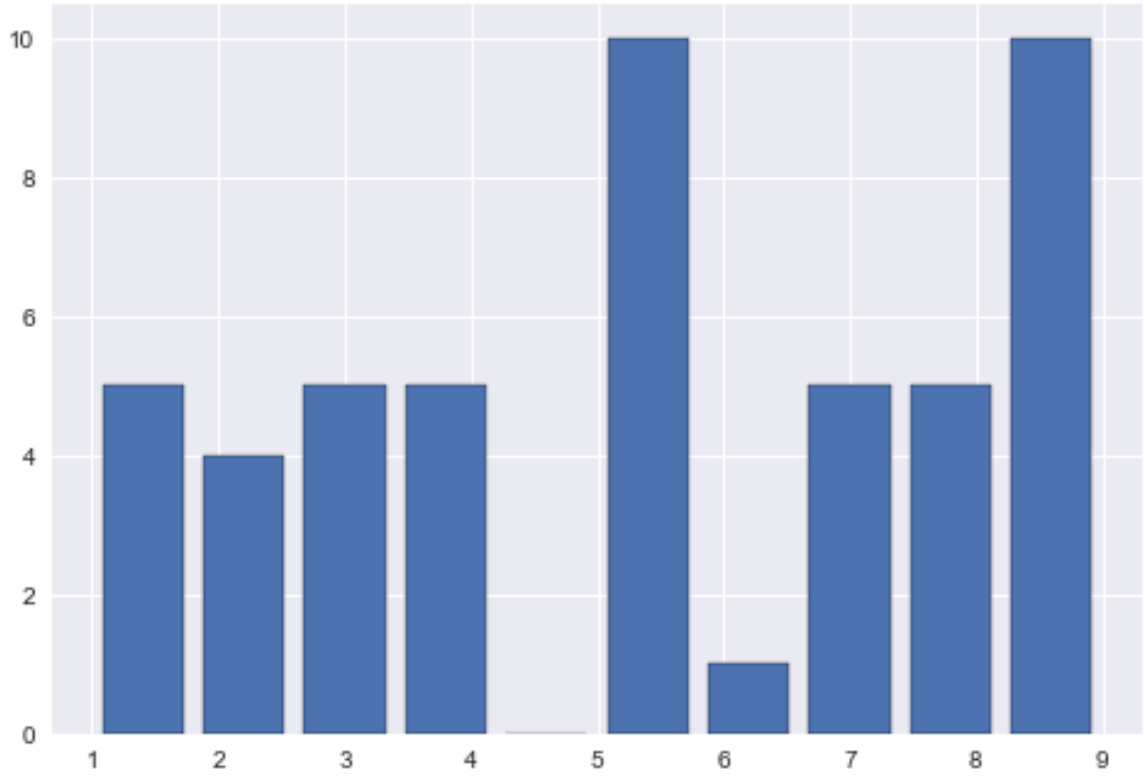
- 用分箱表示数据分布
- 每个桶（称为单桶）代表一个属性-频率对
- 数值范围差异大

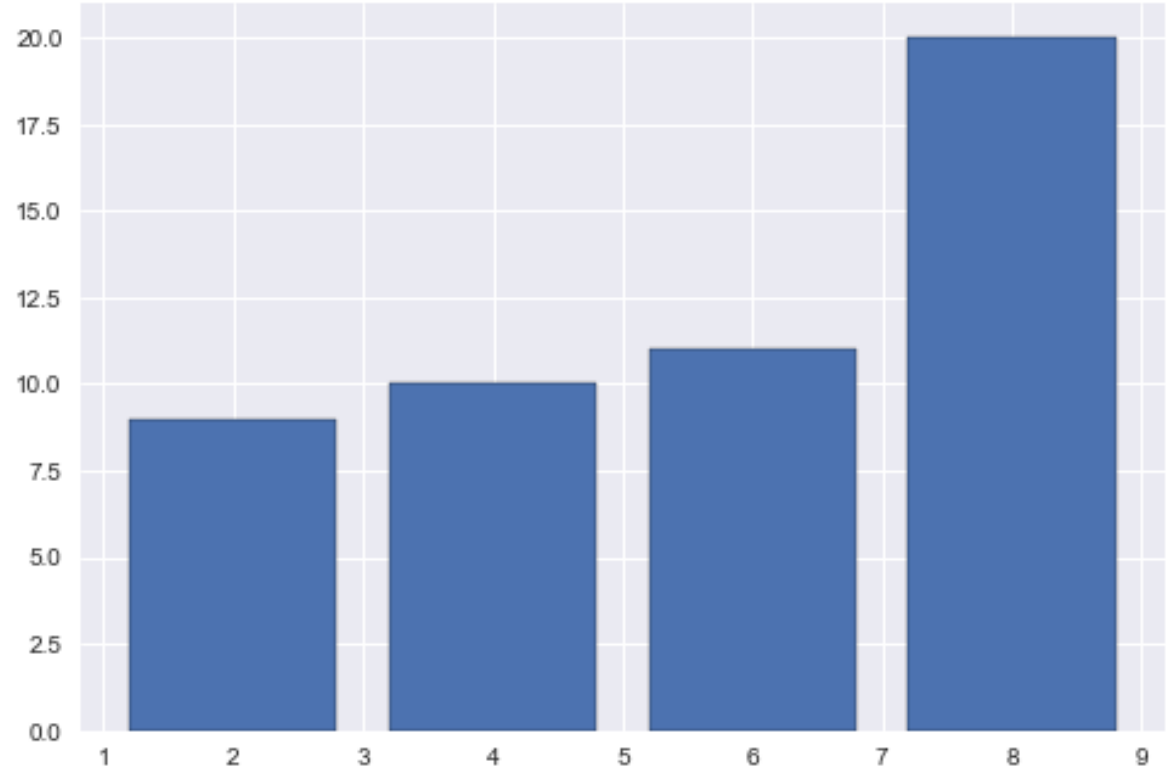
```
array([4, 8, 9, 8, 7, 2, 8, 7, 5, 3, 1, 4, 5, 8, 7, 9, 5, 9, 9, 5, 9, 1, 9, 7, 1, 2, 9, 5,
5, 5, 9, 4, 3, 5, 5, 4, 7, 4, 9, 8, 2, 6, 3, 5, 3, 2, 9, 1, 3, 1])
```

```
h = np.random.randint(1,10,50)
```

数值规约—直方图

plt.hist(data, bins=...)





array([4, 8, 9, 8, 7, 2, 8, 7, 5, 3, 1, 4, 5, 8, 7, 9, 5, 9, 9, 5, 9, 1, 9, 7, 1, 2, 9, 5, 5, 5, 9, 4, 3, 5, 5, 4, 7, 4, 9, 8, 2, 6, 3, 5, 3, 2, 9, 1, 3, 1])

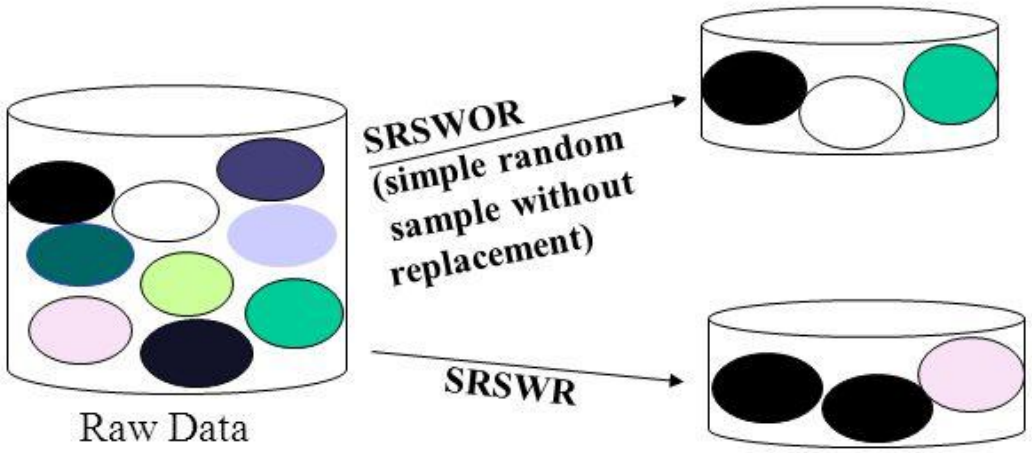
数值规约—抽样

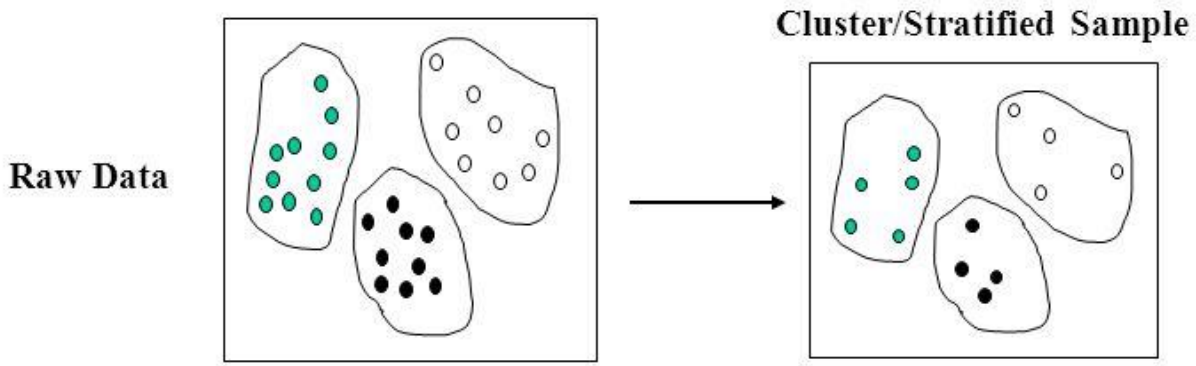
抽样	随机抽样：不放回 随机抽样：放回
	聚类抽样
	分层抽样

特征列举：

- 不放回随机抽样：从原始数据集D的N个样本中抽取n个样本，每次抽到不同的数据
- 放回随机抽样：从原始数据集D的N个样本中抽取n个样本，抽取后记录它后放回，有可能抽到同样的数据
- 分层抽样：数据集D为划分成互不相交的部分即层，对每一层进行简单随机抽样获得最终结果

Sampling Techniques





From: slideplayer.com

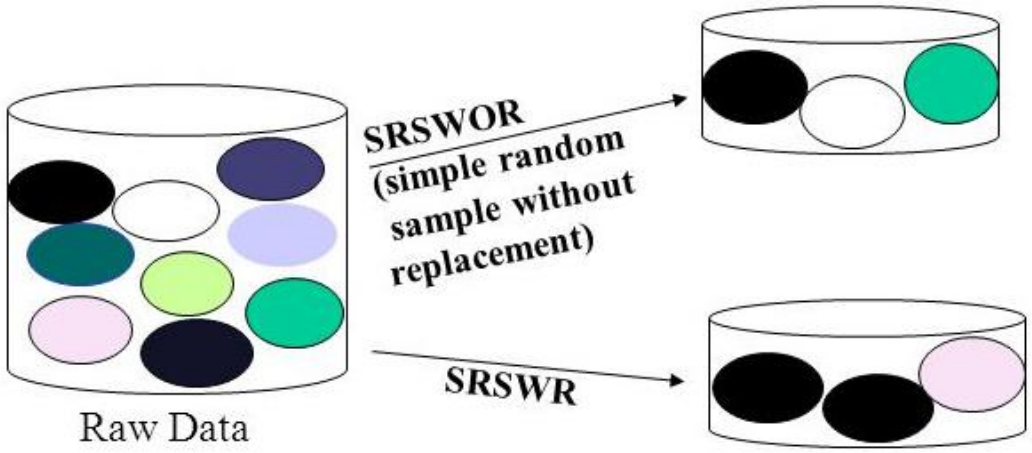
随机抽样

不放回：

```
iris_df.sample(n = 20)
iris_df.sample(frac = 0.3)
```

有放回：

```
iris_df.sample(frac = 0.3, replace = True)
```



分层抽样

```
grp = iris_df.groupby('target')
print(len(grp.groups))

iris_df[iris_df.target==0].sample(frac = 0.3)

iris_df.groupby('target').apply(lambda x: x.sample(frac = 0.3))

sampling_dict = {
    0: 0.3,
    1: 0.5,
    2: 0.3,
}
iris_df.groupby('target').apply(lambda x: x.sample(frac = sampling_dict[x.name]))
```

M3.2 小结

01 数据清洗

02 数据集成

03 数据变换

04 数据规约之数值规约