

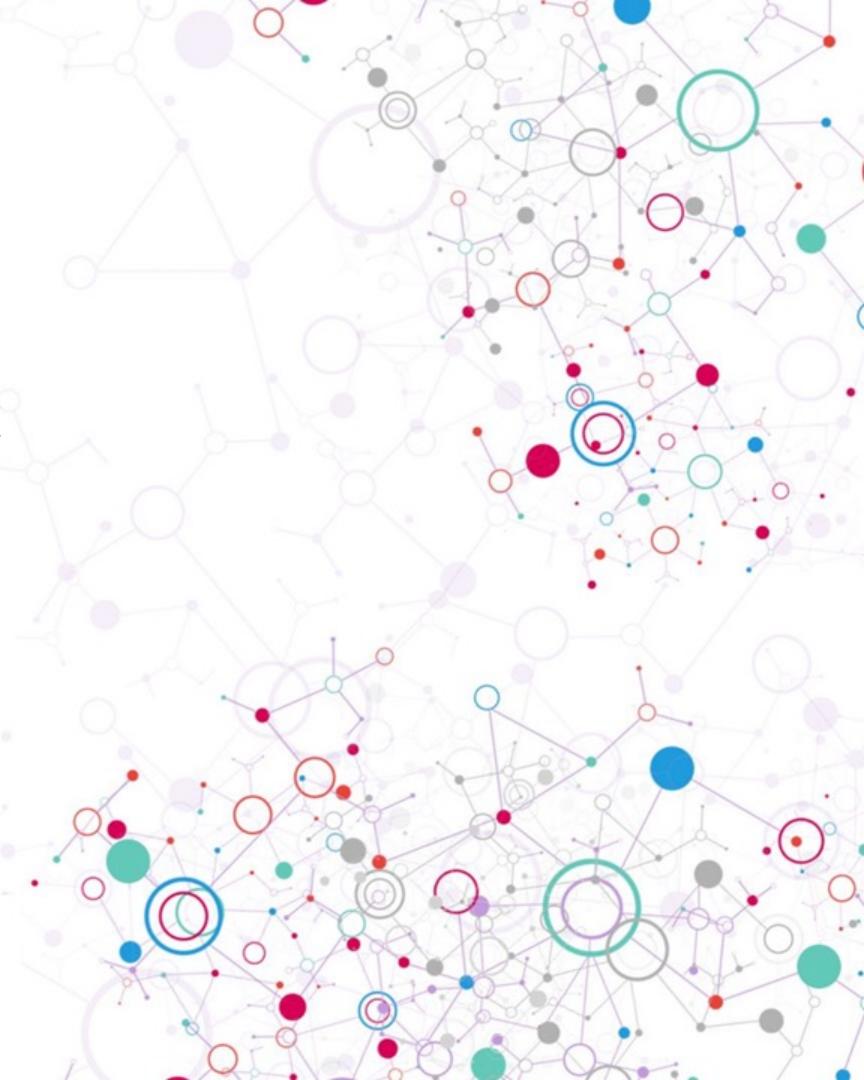
数据思维与实践

王伟

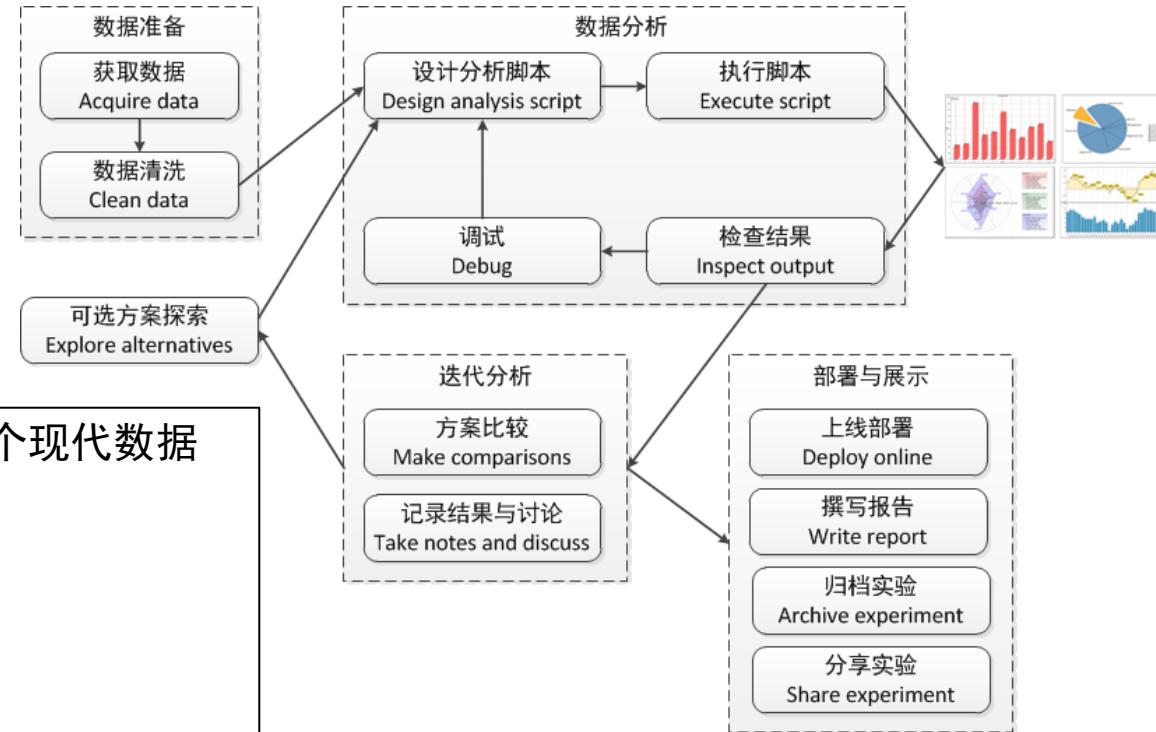
华东师范大学

数据科学与工程学院

全民数字素养与技能培训基地

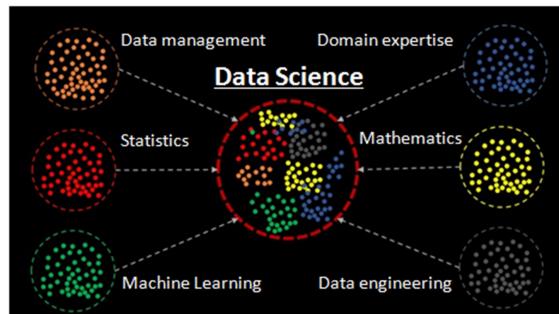


开篇实例：数据科学工作流 (Data science workflow)



数据思维与实践

第04讲 数据探索与预处理



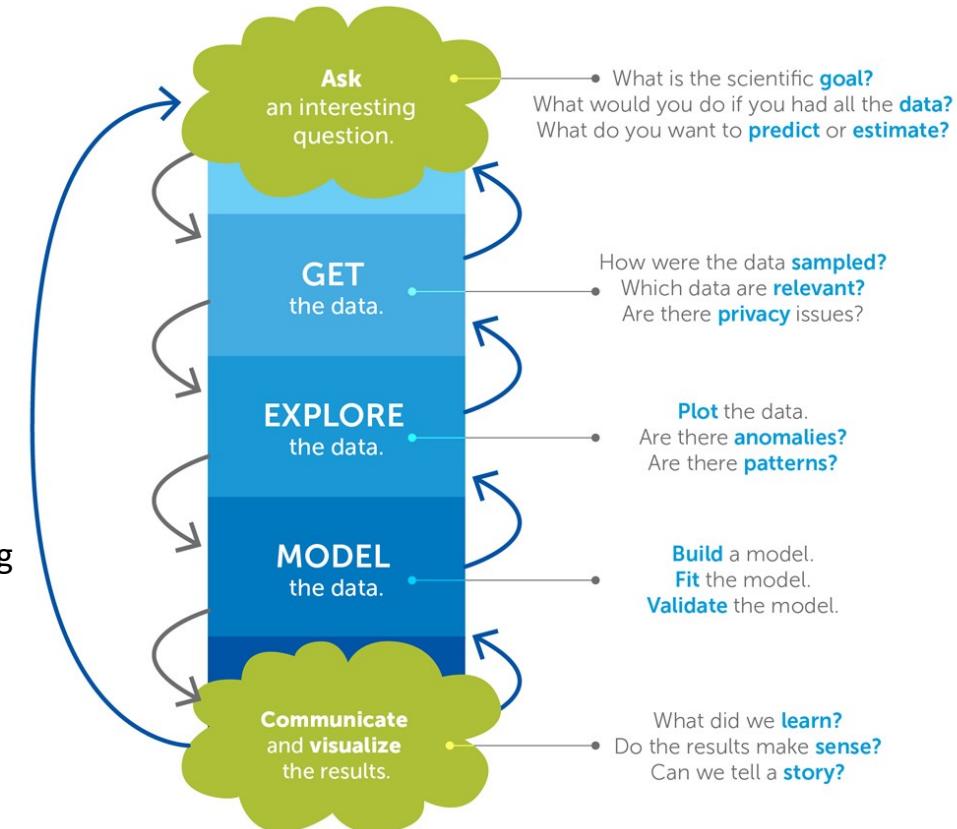
- 数据科学过程
- 数据预处理与探索性分析
- Python 数据预处理与可视化
- 开源数字王国的数据看板

什么是数据科学过程？

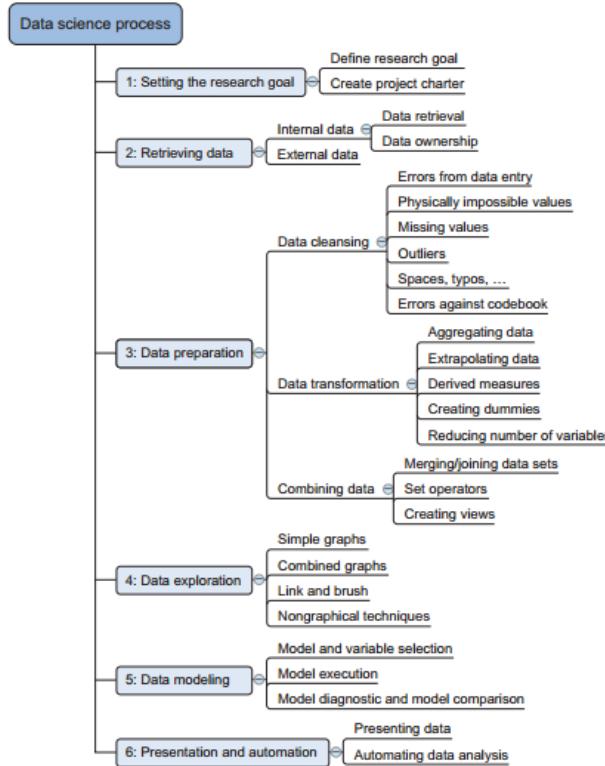
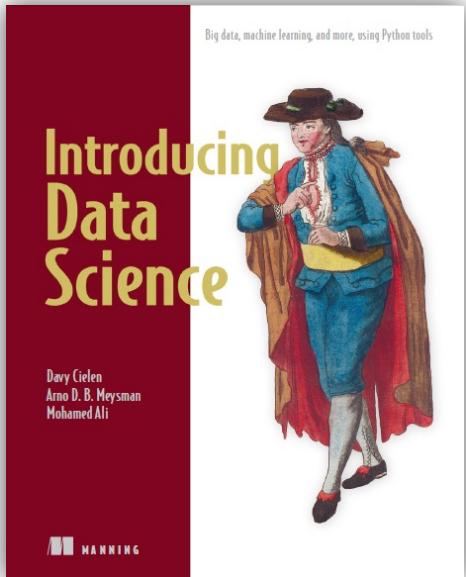
- **微软**: Data Science Process (DSP) 是一种敏捷的迭代式数据科学方法，可有效交付预测分析解决方案和智能应用程序。
 - **Joe Blitzstein** (Harvard): a data science process framework including:
 - Stage 1: Ask A Question
 - Stage 2: Get the Data
 - Stage 3: Explore the Data
 - Stage 4: Model the Data
 - Stage 5: Communicate the Data
- | 1. Business Understanding | 2. Data Acquisition and Understanding | 3. Modeling | 4. Deployment |
|---------------------------|--|--------------------|----------------------------------|
| Technical needs | Load data into storage environments | Engineer features | Publish model as a Web service |
| Identify your scenario | Import data into Azure Machine Learning Studio | Select features | Consume a model programmatically |
| | Prepare data | Learn with counts | Consume a model in Excel |
| | Explore data | Train the model | |
| | Sample data | Evaluate the model | |
| | | Tune the model | |

Data Science Process

1. 提出一个有趣的问题
 - Skills: science, domain expertise, curiosity
 - Tools: your brain, talking to experts, experience
2. 获取数据
 - Skills: web scraping, data cleaning, querying databases, CS stuff
 - Tools: python, pandas
3. 探索数据
 - Skills: Get to know data, develop hypotheses, patterns? anomalies?
 - Tools: matplotlib, numpy, scipy, pandas
4. 数据建模
 - Skills: regression, machine learning, validation, big data
 - Tools: scikits learn, pandas, mrjob, mapreduce
5. 交流与可视化结果
 - Skills: presentation, speaking, visuals, writing
 - Tools: matplotlib, powerpoint/keynote



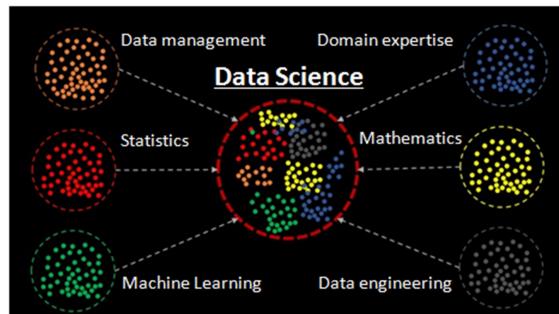
Python 数据科学过程



1. 目标设定
2. 数据获取
3. 数据准备
4. 数据探索
5. 数据建模
6. 数据展示与自动化

数据思维与实践

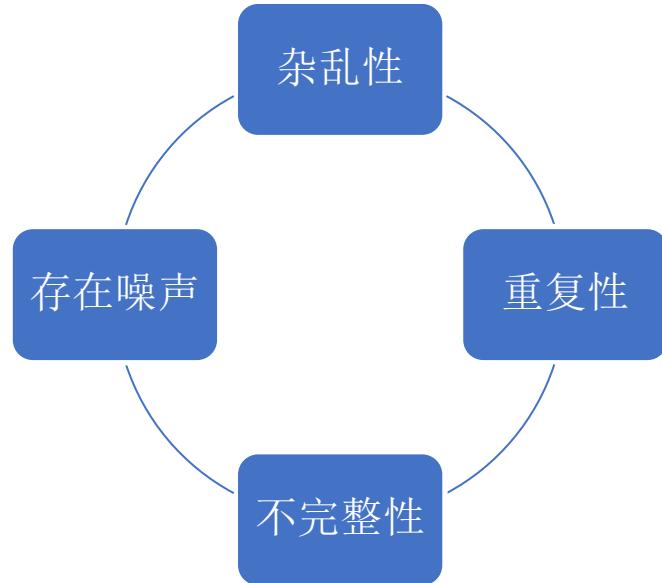
第04讲 数据探索与预处理



- 数据科学过程
- **数据预处理与探索性分析**
- Python 数据预处理与可视化
- 开源数字王国的数据看板

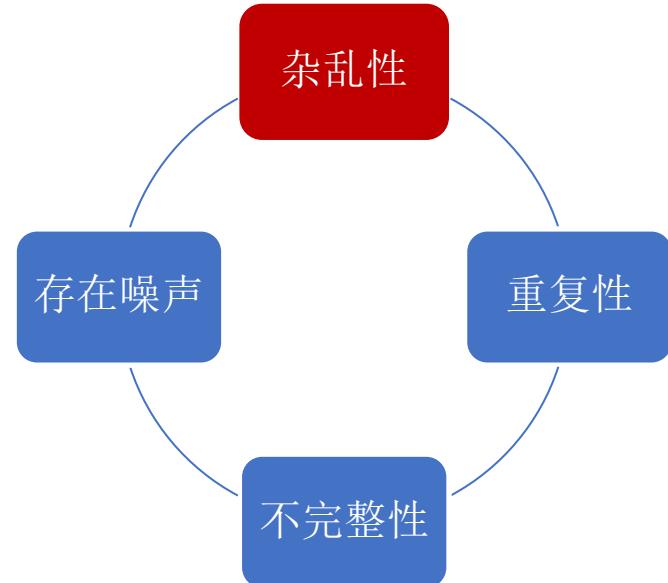
数据预处理的原因

- **数据预处理**是重要的第一步，是最耗费时间的一环，很枯燥和烦闷，但是绝对不可或缺。数据挖掘的价值有百分之八十都取决于探索式数据分析和数据清洗的效果。
- 原始数据中存在着各种问题：
 - 杂乱性
 - 重复性
 - 不完整性
 - 存在噪声



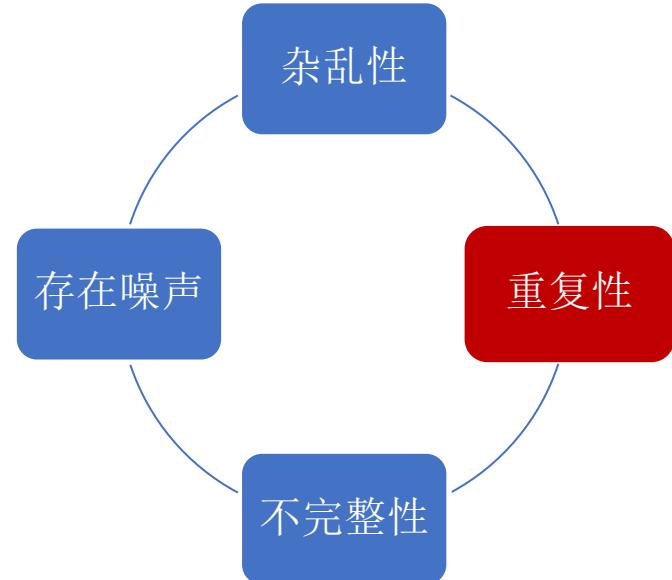
杂乱性

- **杂乱性**是指系统中的数据缺乏统一的标准和定义，具体表现形式包括：
 - 在不同的数据来源中的同义异名情况，例如为了表示客户，一些数据来源用 `cust_id` 来区分，另一些则用 `cust_number`；
 - 不同数据来源采用的度量标准可能不同，比如对于性别，一些采用 M 或 F 来区分，另一些则采用男或女来区分；
 - 对同一属性定义的类型不同，以工资为例，一些数据来源可能定义为 Int 型，另一些则将工资定义为 Double 型。



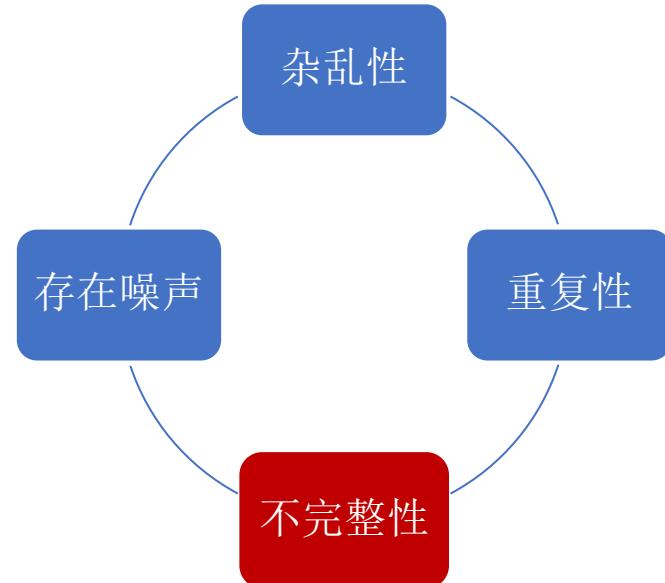
重复性

- **重复性**是指同一事物在数据库中存在两条或多条完全相同的记录。
- 这种情况非常常见，如实际使用过程中出现的意义相同或者表示同一信息的多个属性。



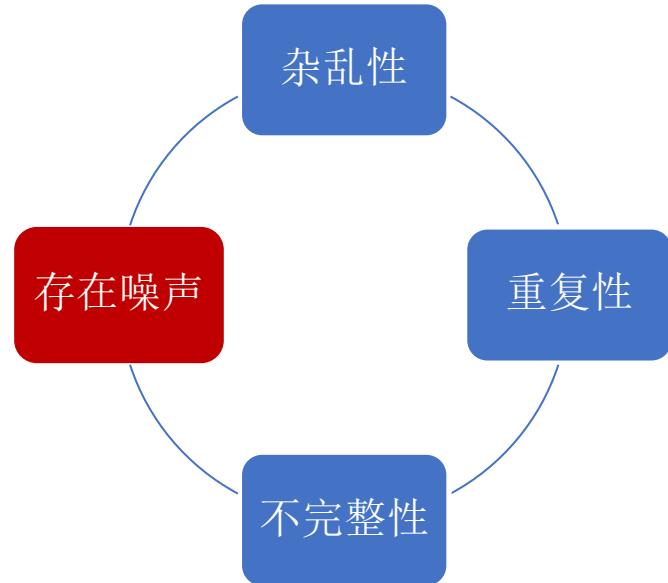
不完整性

- 不完整性是指系统设计的不合理或者使用过程中的某些因素所造成的属性值缺失或者不确定；或者是某个元组中缺少了某样或者某几样属性，甚至是多个元组直接缺失。
- 造成这种情况的原因，或许是在数据输入时，某些数据可能被误认为不重要而删除掉了；或是某些数据由于存在不一致性，结果被删除了。



存在噪声

- **存在噪声**是指测量变量中的随机错误或偏离期望的鼓励点值。噪声数据的来源众多，起因也各异。



数据预处理的主要任务

- 按照所处理的内容不同，可以分为以下几类：
 - **数据清洗**：数据清理即填写空缺值、平滑噪声数据、识别删除孤立点，以及解决数据中的不一致性问题；
 - **数据集成**：数据集成即通过操作集成多个来源不同的数据库、数据立方或文件；
 - **数据变换**：数据变换即对原始数据进行规范化和聚集操作；
 - **数据规约**：数据规约即通过操作得到数据集的压缩表示，所得到的压缩表示将会小得多，但可以在其上得到与原始数据相同或相近的数据挖掘结果。

数据清洗

- **数据清洗**的主要任务就是对原始数据进行处理，将“**脏**”数据转化为“**干净的**”数据。其主要任务包括：
 - 填补空缺值
 - 平滑噪声数据
 - 纠正不一致数据
 - 消除冗余数据

数据集成

- **数据集成**就是将多个数据源中的数据结合起来存放在一个一致的数据存储中。在数据集成的过程中，通常需要考虑多信息源的匹配、数据冗余、数据值冲突等问题。
- 将不同源的数据集成到一起，需要完成各信息源的匹配，即从多信息源中识别现实世界的实体，并进行匹配。
- 有的时候需要借助元数据（即数据的数据）的帮助，从而避免在数据集成中发生错误。

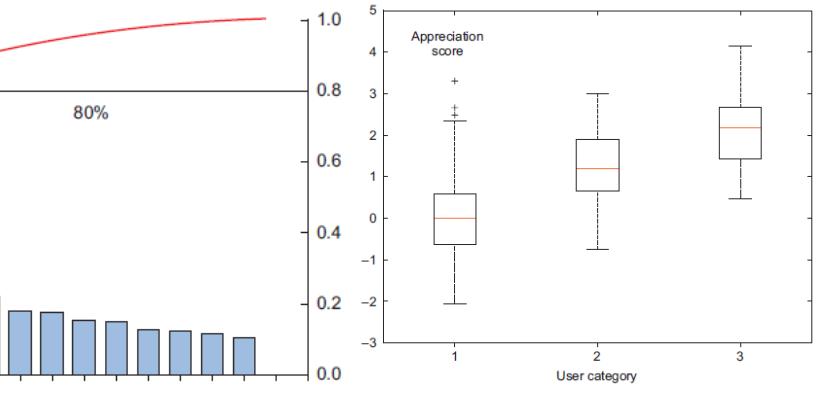
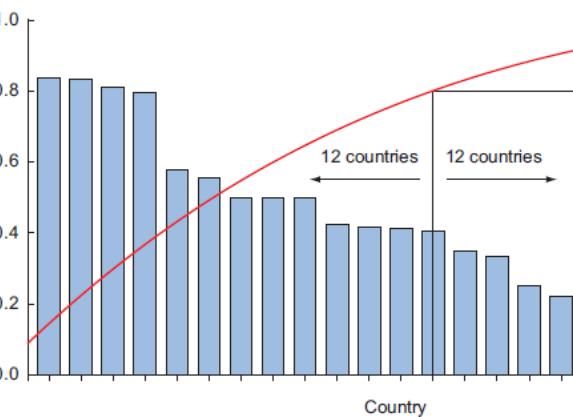
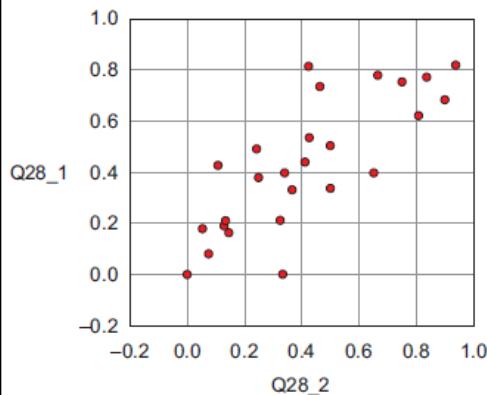
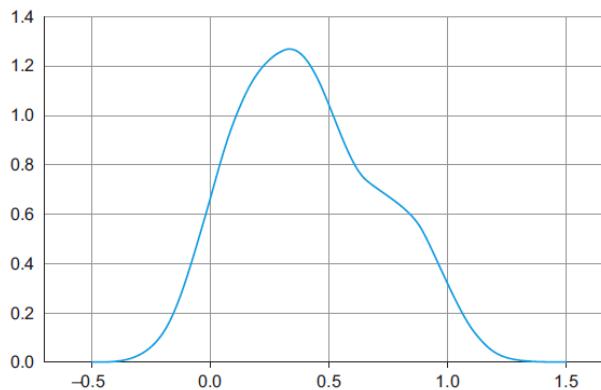
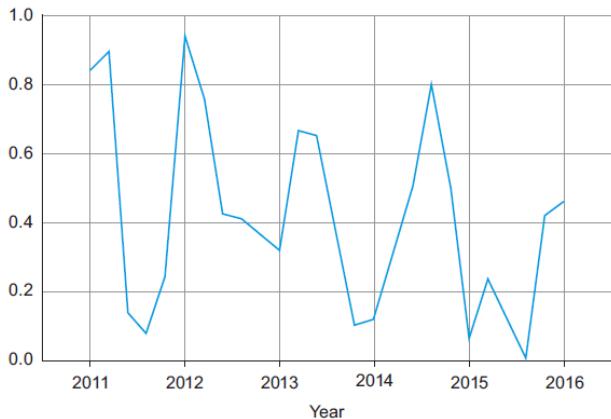
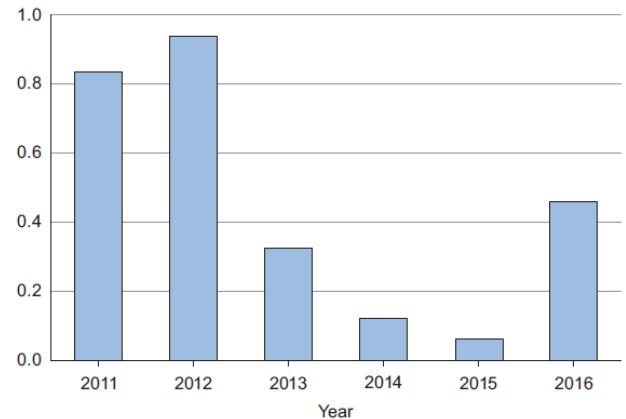
数据变换

- **数据变换**是采用数学变换方法将多维数据压缩成较少维数的数据，消除它们在时间、空间、属性及精度等特征表现方面的差异。这种方法虽然对原始数据都有一定的损害，但其结果往往具有更大的实用性。
- 常见数据变换方法：
 - 数据平滑
 - 数据聚集
 - 数据概括
 - 数据规范化
 - 属性构造

数据归约

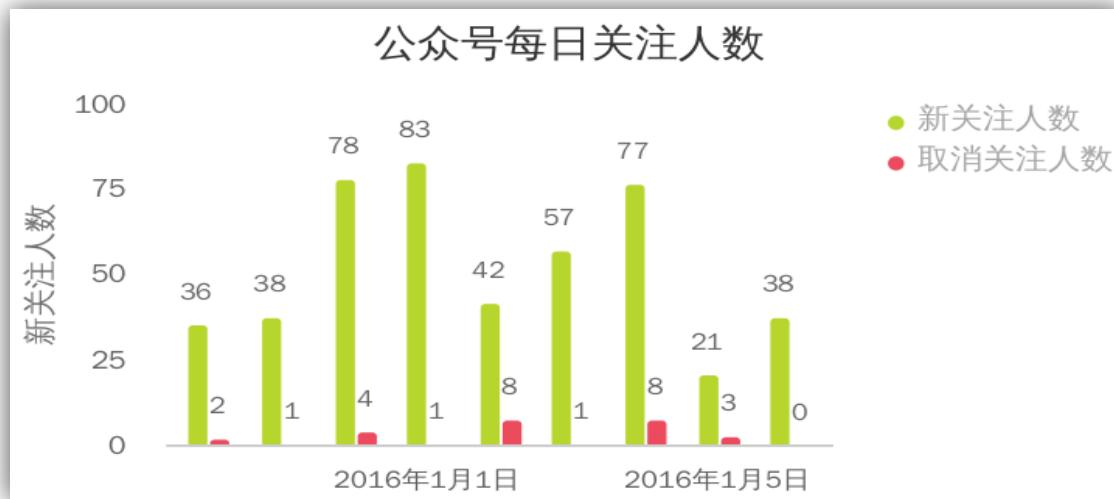
- **数据归约**是因为被分析的对象数据集往往非常大，分析与挖掘会特别耗时甚至不能进行。通过数据归约处理，可以减少对象数据集大小。
- 数据归约技术能够从原有的庞大数据集中获得一个精简的数据集合，并使这一精简的数据集保持原有数据集的完整性，以提高数据挖掘的效率。
 - 所得归约数据集要小；
 - 归约后的数据集仍接近于保持原数据的完整性；
 - 在归约数据集上所得分析结果应与原始数据集相同或基本相同；
 - 归约处理时间少于挖掘所节约的时间。

可视化探索性分析



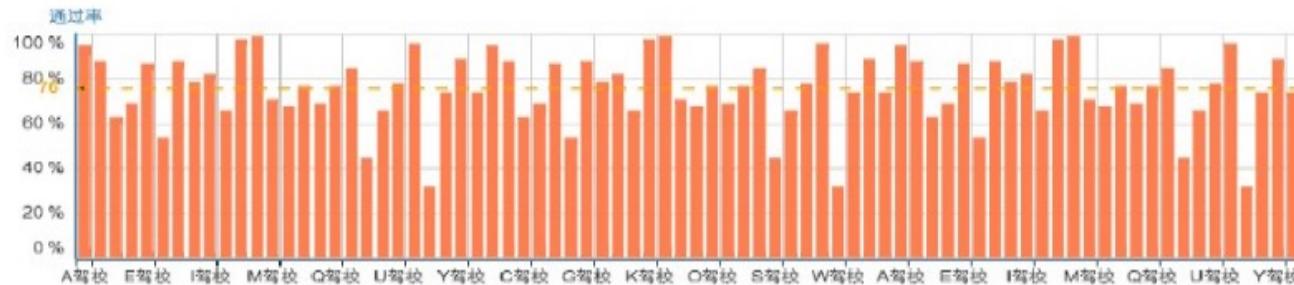
柱状图

- 柱状图是一种以长方形的长度为变量的表达图形的统计报告图，它由一系列高度不等的纵向条纹表示数据分布的情况。



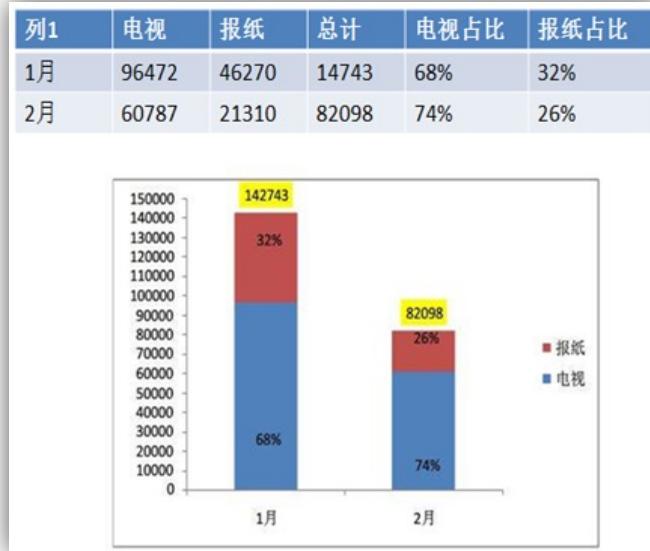
柱状图适用场景和优缺点

- **适用场景**: 适用场合是二维数据集（每个数据点包括两个值x和y），但只有一个维度需要比较，用于显示一段时间内的数据变化或显示各项之间的比较情况。
 - **优点**: 利用人眼对高度差异的敏感性，将数据大小反应在柱子的高度上，可以很明显的对比数据的差异
 - **缺点**: 柱状图的局限在于只适用中小规模的数据集。



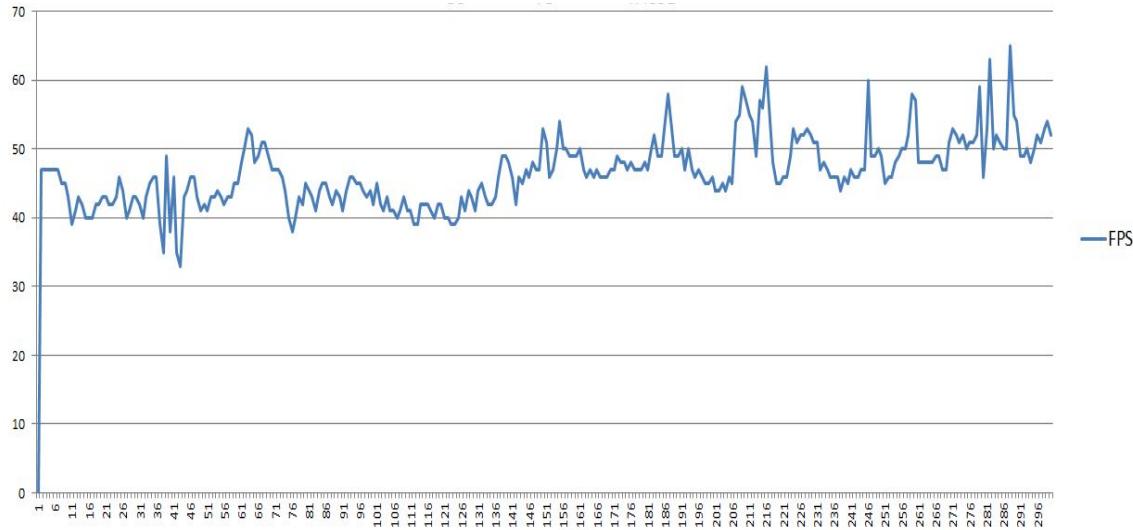
堆积柱状图

- **堆积柱状图**不仅可以直观的看出每个系列的值，还能反映出系列的总和，尤其是当需要看某一单位的综合以及各系列值的比重时最适合。



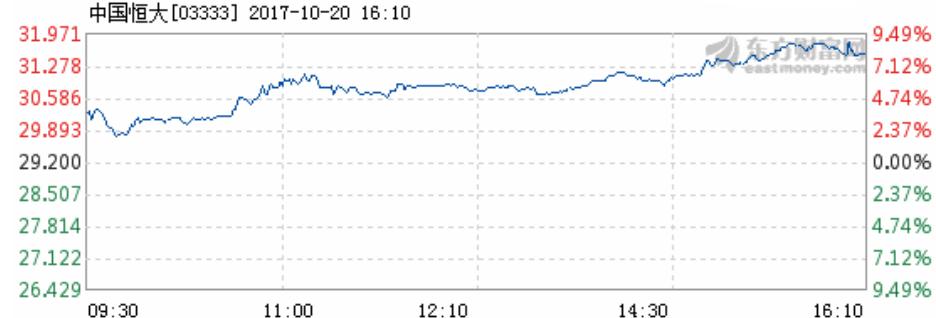
折线图

- **折线图**将排列在工作表列或行中的数据进行绘制。折线图可以显示随时间而变化的连续数据，因此非常适用于显示在相等时间间隔下数据的趋势。

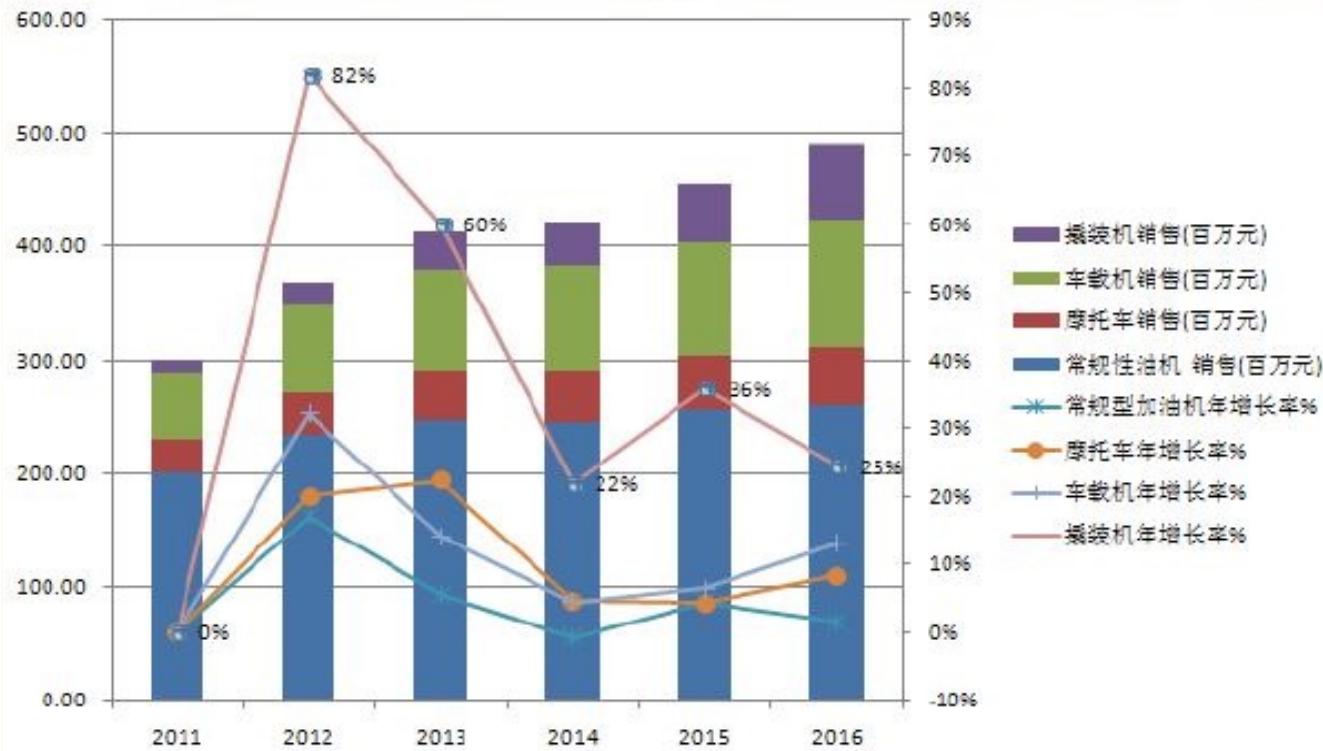


折线图

- **适用场景：**折线图将排列在工作表列或行中的数据进行绘制，适用二维的较大数据集，特别是分析趋势比单个数据点更重要的分析过程。
- 一般用来分析某时间段内事物的变化趋势，纵轴 Y 为可量化的变量，横轴 X 为时间维度

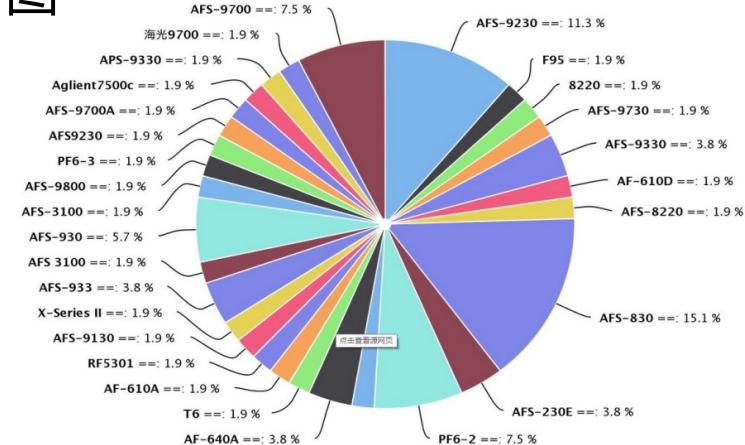


柱状图与折线图结合



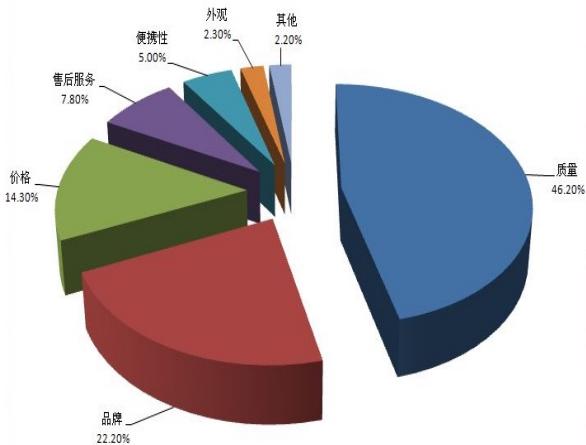
饼图

- 饼图以二维或三维格式显示每一数值相对于总数值的大小。
 - 饼图具有较大的局限性，因为人对面积的大小不敏感
 - 一般情况下，可以使用柱状图代替饼图
 - 当反映部分占总体比重时，可以用饼图

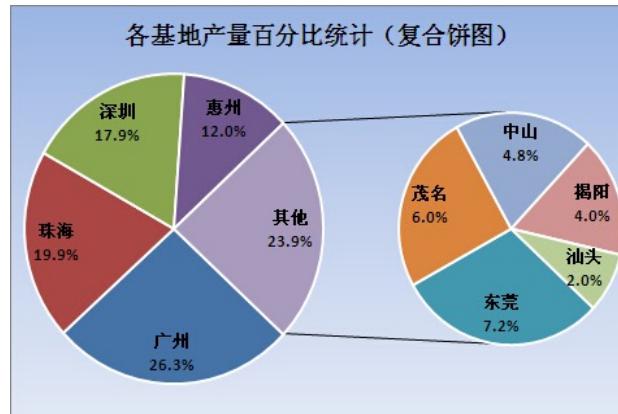


分离型与复合型饼图

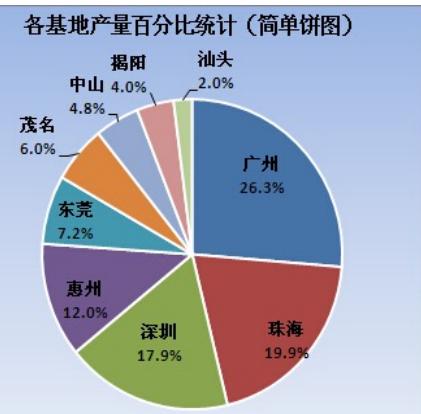
- 除了传统的饼图以外，还有**分离型**饼图和**复合型**饼图。



分离饼图

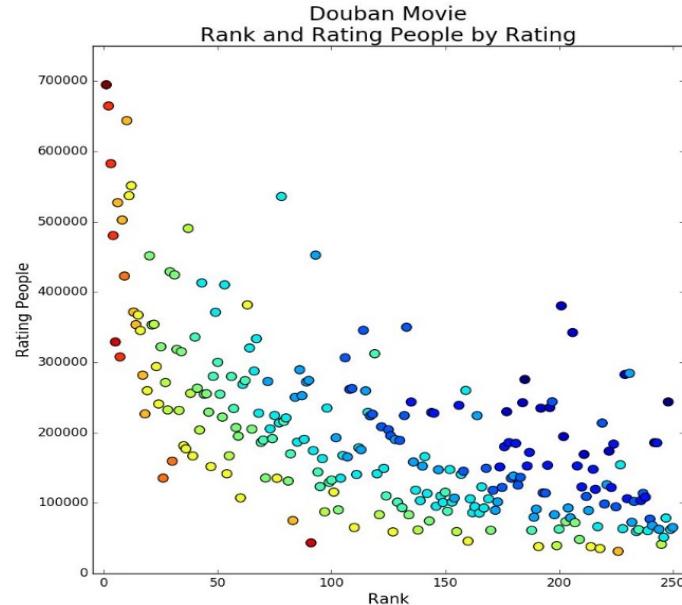


复合饼图



散点图

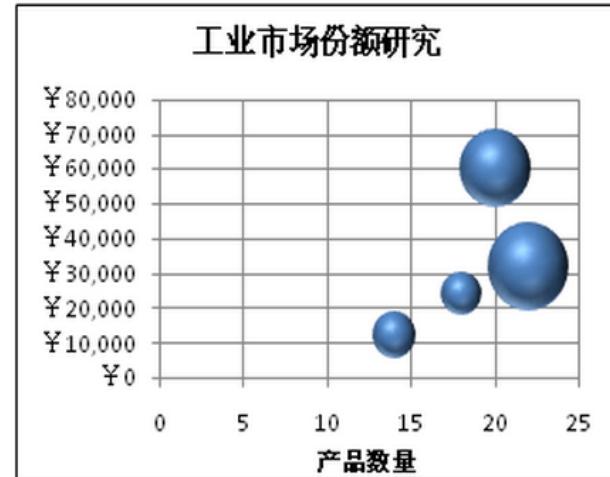
- **散点图**适用于三维数据集，但是其中只有两个维度需要比较。为了识别第三个维度，可以为每个点加上文字标识或不同的颜色。
 - **适用场景：**散点图适用于三维数据集，但是其中只有两个维度需要比较；
 - 一般情况，为了识别第三个维度，可以为每个点加上文字标示，或者不同的颜色；
 - 散点图可以分析坐标点的分布情况，判断两个变量之间的关联或分布趋势。



气泡图

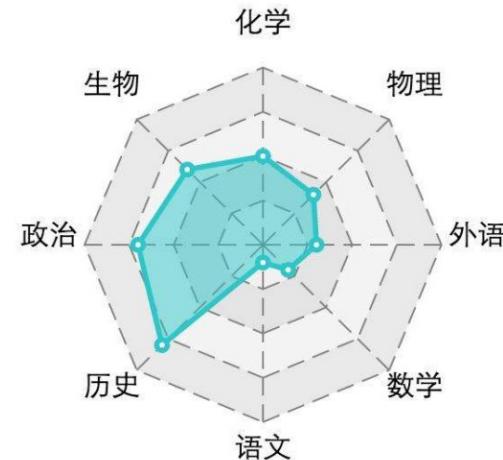
- 气泡图是散点图的变体，在气泡图中，三个维度都是可变化的量；
- 气泡图将散点图的数据点变为气泡，通过气泡的面积大小反应第三维度变量值，面积越大表示强度越大（因为用户不善于判断面积大小，所以气泡图只适用于不要求精确判断第三维的场合）；
- 如果数据有三个维度，并且每个维度都包含一组数值，则可以使用气泡图来代替散点图。

产品数量	销售额	市场份额%
14	¥12,200	12%
20	¥60,000	33%
18	¥24,400	10%
22	¥32,000	42%



雷达图

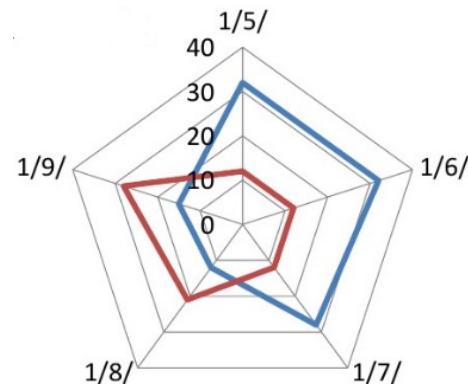
- **雷达图**也称为蜘蛛图、星状图，是一种以二维形式展示多维数据的图形，雷达图从中心点出发辐射出多条坐标轴，数据在每一维度的数值都占用了一条坐标轴，并和相邻坐标轴上的数据点连接起来，形成一个不规则多边形。



雷达图

- **适用场景：**雷达图适用于多维数据，而且每个维度可以排序：
 - 不同数值单位可以不同，但需要按照同样比例进行分布；
 - 雷达图可以对两组变量进行多种项目的对比，常用于多想指标的全面分析，尤其适用于对多属性体系结构描述的对象作出全局性、整体性评价；
 - 雷达图的重要特点是**直观**，从雷达图可以直观地看出评价对象的状况，因而可以直接用雷达图进行定性评价。

	销售	管理	技术	客服	研发
预算分配	4300	10000	28000	35000	50000
实际开销	5000	14000	28000	31000	42000

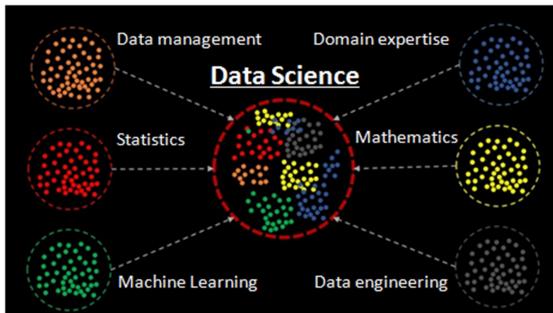


总结

图表	维度	注意
柱状图	二维	只能比较其中一维
折线图	二维	适用于较大的数据集
饼图	二维	只适用反映部分与整体的关系
散点图	二维或三维	有两个维度需要比较
气泡图	三维或四维	其中只有两个维度能精准辨识
雷达图	四维以上	需要添加文字说明

数据思维与实践

第04讲 数据探索与预处理



- 数据科学过程
- 数据预处理与探索性分析
- **Python 数据预处理与可视化**
- 开源数字王国的数据看板

Matplotlib



Matplotlib 是 Python 语言及其数值计算库 NumPy 的绘图库。它提供了一个面向对象的 API，可以将绘图嵌入到使用通用 GUI 工具包的程序中。

Matplotlib 的安装与简单制图

```
python -m pip install matplotlib
```

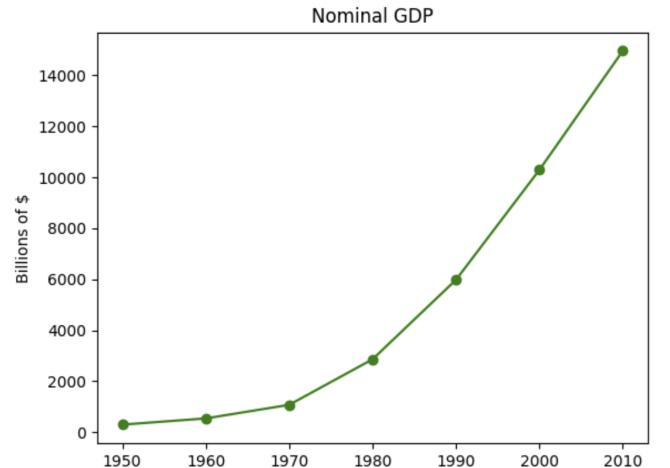
```
from matplotlib import pyplot as plt

years = [1950, 1960, 1970, 1980, 1990, 2000, 2010]
gdp = [300.2, 543.3, 1075.9, 2862.5, 5979.6, 10289.7, 14958.3]

# create a line chart, years on x-axis, gdp on y-axis
plt.plot(years, gdp, color='green', marker='o', linestyle='solid')

# add a title
plt.title("Nominal GDP")

# add a label to the y-axis
plt.ylabel("Billions of $")
plt.show()
```



条形图

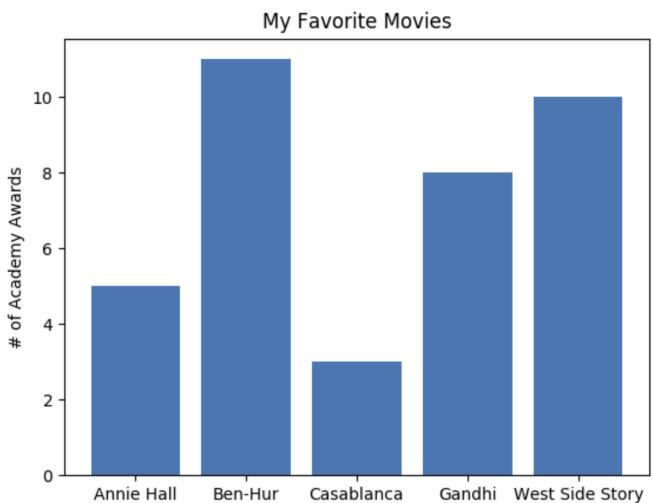
```
movies = ["Annie Hall", "Ben-Hur", "Casablanca", "Gandhi", "West Side Story"]
num_oscars = [5, 11, 3, 8, 10]

# plot bars with left x-coordinates [0, 1, 2, 3, 4], heights [num_oscars]
plt.bar(range(len(movies)), num_oscars)

plt.title("My Favorite Movies")      # add a title
plt.ylabel("# of Academy Awards")    # label the y-axis

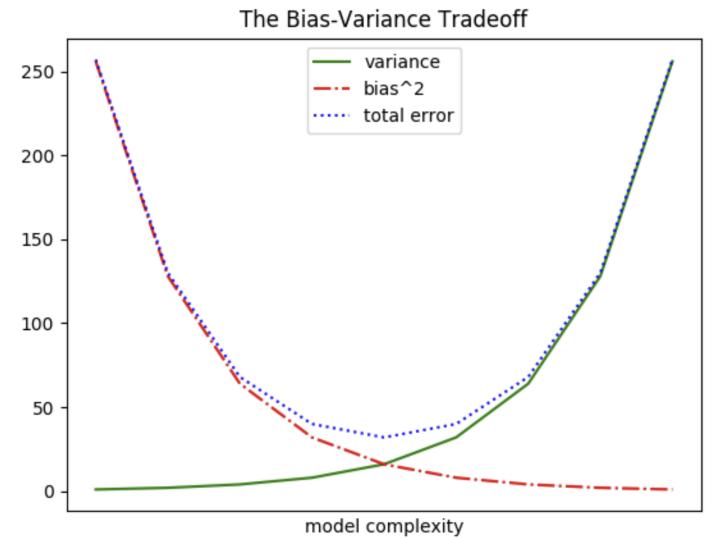
# label x-axis with movie names at bar centers
plt.xticks(range(len(movies)), movies)

plt.show()
```



线图

```
variance      = [1, 2, 4, 8, 16, 32, 64, 128, 256]
bias_squared = [256, 128, 64, 32, 16, 8, 4, 2, 1]
total_error  = [x + y for x, y in zip(variance, bias_squared)]
xs = [i for i, _ in enumerate(variance)]  
  
# We can make multiple calls to plt.plot  
# to show multiple series on the same chart
plt.plot(xs, variance,      'g-',    label='variance')      # green solid line
plt.plot(xs, bias_squared,  'r-.',   label='bias^2')        # red dot-dashed line
plt.plot(xs, total_error,   'b:',    label='total error')   # blue dotted line  
  
# Because we've assigned labels to each series,  
# we can get a legend for free (loc=9 means "top center")
plt.legend(loc=9)
plt.xlabel("model complexity")
plt.xticks([])
plt.title("The Bias-Variance Tradeoff")
plt.show()
```



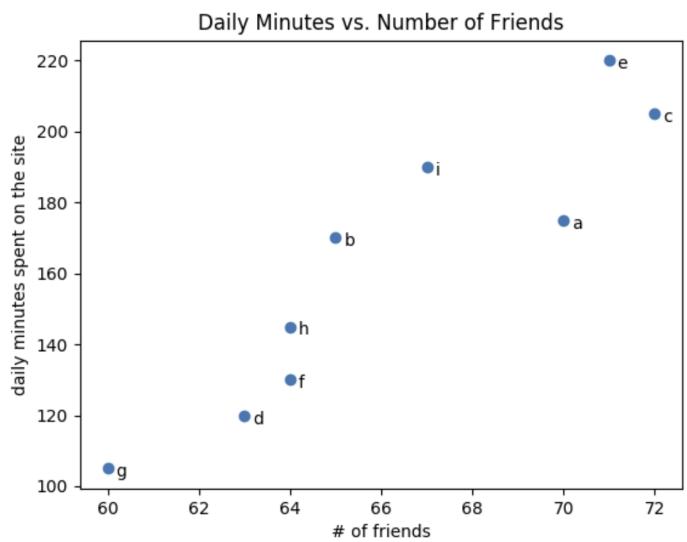
散点图

```
friends = [ 70, 65, 72, 63, 71, 64, 60, 64, 67]
minutes = [175, 170, 205, 120, 220, 130, 105, 145, 190]
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']

plt.scatter(friends, minutes)

# label each point
for label, friend_count, minute_count in zip(labels, friends, minutes):
    plt.annotate(label,
                 xy=(friend_count, minute_count), # Put the label with its point
                 xytext=(5, -5),                  # but slightly offset
                 textcoords='offset points')

plt.title("Daily Minutes vs. Number of Friends")
plt.xlabel("# of friends")
plt.ylabel("daily minutes spent on the site")
plt.show()
```



探索一维数据

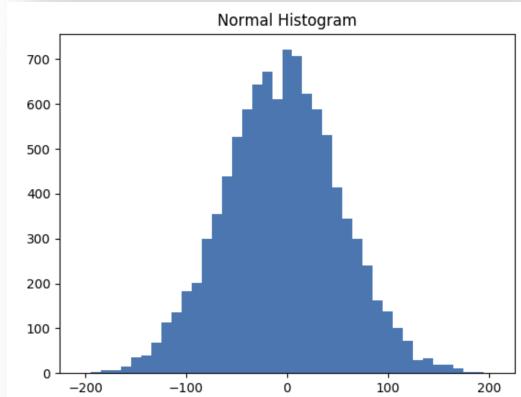
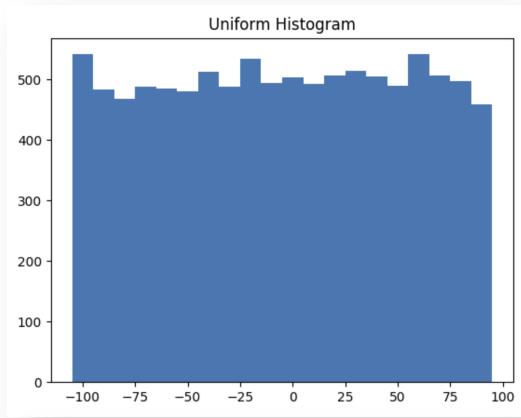
```
from typing import List, Dict
from collections import Counter
import math

import matplotlib.pyplot as plt

def bucketize(point: float, bucket_size: float) -> float:
    """Floor the point to the next lower multiple of bucket_size"""
    return bucket_size * math.floor(point / bucket_size)

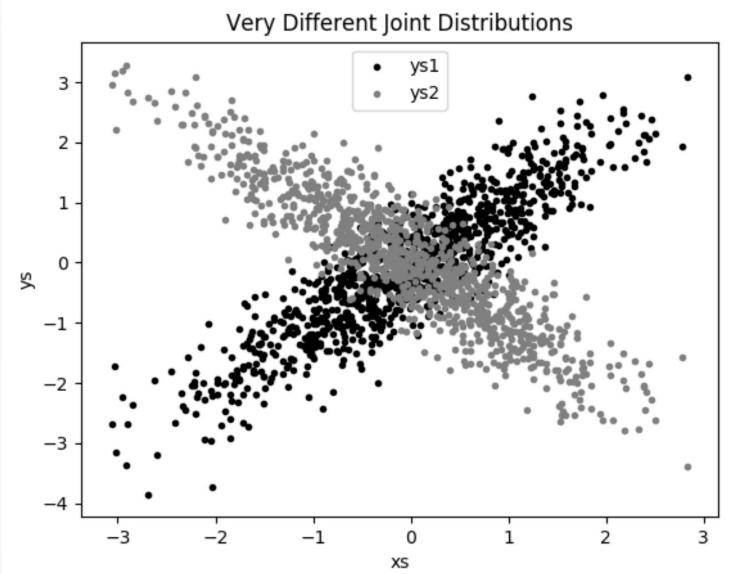
def make_histogram(points: List[float], bucket_size: float) -> Dict[float, int]:
    """Buckets the points and counts how many in each bucket"""
    return Counter(bucketize(point, bucket_size) for point in points)

def plot_histogram(points: List[float], bucket_size: float, title: str = ""):
    histogram = make_histogram(points, bucket_size)
    plt.bar(histogram.keys(), histogram.values(), width=bucket_size)
    plt.title(title)
```



两个维度

```
def random_normal() -> float:  
    """Returns a random draw from a standard normal distribution"""  
    return inverse_normal_cdf(random.random())  
  
xs = [random_normal() for _ in range(1000)]  
ys1 = [x + random_normal() / 2 for x in xs]  
ys2 = [-x + random_normal() / 2 for x in xs]  
  
plt.scatter(xs, ys1, marker='.', color='black', label='ys1')  
plt.scatter(xs, ys2, marker='.', color='gray', label='ys2')  
plt.xlabel('xs')  
plt.ylabel('ys')  
plt.legend(loc=9)  
plt.title("Very Different Joint Distributions")  
plt.show()
```



多维数据

```
from scratch.linear_algebra import Matrix, Vector, make_matrix

def correlation_matrix(data: List[Vector]) -> Matrix:
    """
    Returns the len(data) x len(data) matrix whose (i, j)-th entry
    is the correlation between data[i] and data[j]
    """
    def correlation_ij(i: int, j: int) -> float:
        return correlation(data[i], data[j])

    return make_matrix(len(data), len(data), correlation_ij)

# corr_data is a list of four 100-d vectors
num_vectors = len(corr_data)
fig, ax = plt.subplots(num_vectors, num_vectors)

for i in range(num_vectors):
    for j in range(num_vectors):

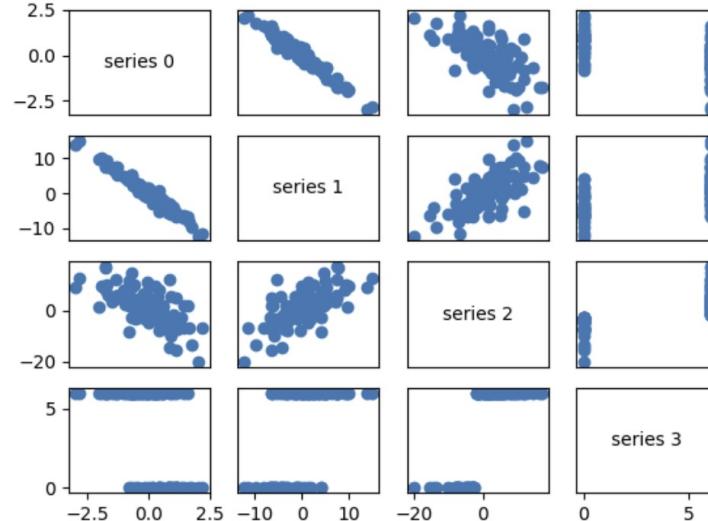
        # Scatter column_j on the x-axis vs. column_i on the y-axis
        if i != j: ax[i][j].scatter(corr_data[j], corr_data[i])

        # unless i == j, in which case show the series name
        else: ax[i][j].annotate("series " + str(i), (0.5, 0.5),
                               xycoords='axes fraction',
                               ha="center", va="center")

        # Then hide axis labels except left and bottom charts
        if i < num_vectors - 1: ax[i][j].xaxis.set_visible(False)
        if j > 0: ax[i][j].yaxis.set_visible(False)

    # Fix the bottom-right and top-left axis labels, which are wrong because
    # their charts only have text in them
    ax[-1][-1].set_xlim(ax[0][-1].get_xlim())
    ax[0][0].set_ylim(ax[0][1].get_ylim())

plt.show()
```



NamedTuple 和数据类

```
import datetime

stock_price = {'closing_price': 102.06,
               'date': datetime.date(2014, 8, 29),
               'symbol': 'AAPL'}
```

```
if stock_price.clo
    closing_price
```

```
from collections import namedtuple

StockPrice = namedtuple('StockPrice', ['symbol', 'date', 'closing_price'])
price = StockPrice('MSFT', datetime.date(2018, 12, 14), 106.03)

assert price.symbol == 'MSFT'
assert price.closing_price == 106.03

from typing import NamedTuple

class StockPrice(NamedTuple):
    symbol: str
    date: datetime.date
    closing_price: float

    def is_high_tech(self) -> bool:
        """It's a class, so we can add methods too"""
        return self.symbol in ['MSFT', 'GOOG', 'FB', 'AMZN', 'AAPL']

price = StockPrice('MSFT', datetime.date(2018, 12, 14), 106.03)

assert price.symbol == 'MSFT'
assert price.closing_price == 106.03
assert price.is_high_tech()
```

清洗、修改、处理与调整

```
import csv

data: List[StockPrice] = []

with open("comma_delimited_stock_prices.csv") as f:
    reader = csv.reader(f)
    for row in reader:
        maybe_stock = try_parse_row(row)
        if maybe_stock is None:
            print(f"skipping invalid row: {row}")
        else:
            data.append(maybe_stock)
```

Person	Height (inches)	Height (centimeters)	Weight (pounds)
A	63	160	150
B	67	170.2	160
C	70	177.8	171

```
from typing import Tuple

from scratch.linear_algebra import vector_mean
from scratch.statistics import standard_deviation

def scale(data: List[Vector]) -> Tuple[Vector, Vector]:
    """returns the mean and standard deviation for each position"""
    dim = len(data[0])

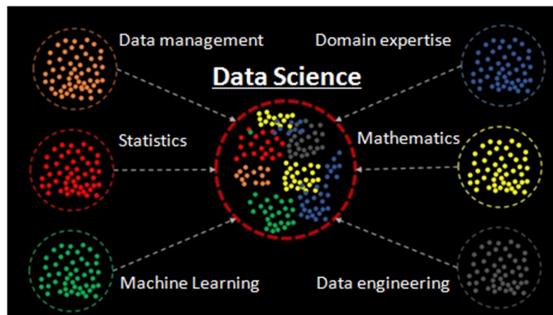
    means = vector_mean(data)
    stdevs = [standard_deviation([vector[i] for vector in data])
              for i in range(dim)]

    return means, stdevs

vectors = [[-3, -1, 1], [-1, 0, 1], [1, 1, 1]]
means, stdevs = scale(vectors)
assert means == [-1, 0, 1]
assert stdevs == [2, 1, 0]
```

数据思维与实践

第03讲 数据收集与管理



- 大数据时代
- 数据的全生命周期
- Python 数据收集方法
- **开源数字王国的数据看板**



OpenDigger

回顾：OpenDigger 开源项目

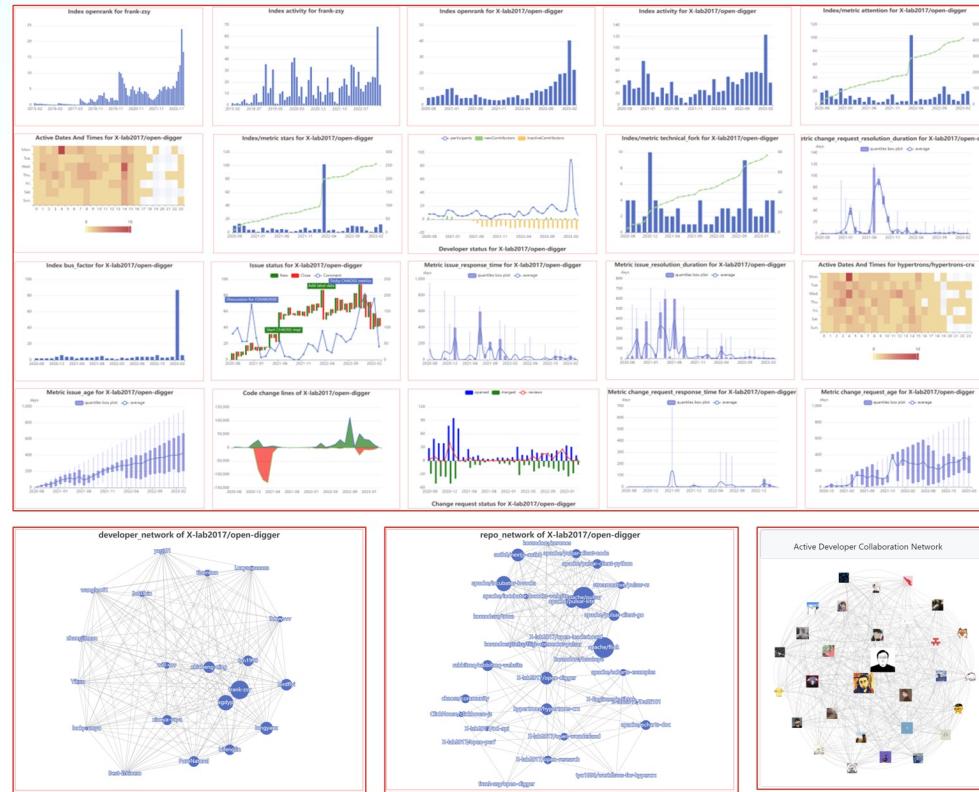
指标

For repos

- [标准院/X-lab] **activity**
- [标准院/X-lab] **openrank** 💎
- [标准院/X-lab] **attention**
- [标准院/X-lab] **stars**
- [标准院/X-lab] **issue_comments**
- [标准院/X-lab] **participants**
- [LF/CHAOSS] **technical_fork**
- [LF/CHAOSS] **issues_new**
- [LF/CHAOSS] **issues_closed**
- [LF/CHAOSS] **code_change_lines_add**
- [LF/CHAOSS] **code_change_lines_rem**
- [LF/CHAOSS] **code_change_lines_sum**
- [LF/CHAOSS] **change_requests**
- [LF/CHAOSS] **change_requests_accept**
- [LF/CHAOSS] **change_requests_reviews**
- [LF/CHAOSS] **bus_factor**

For users

- [标准院/X-lab] **activity**
- [标准院/X-lab] **openrank**



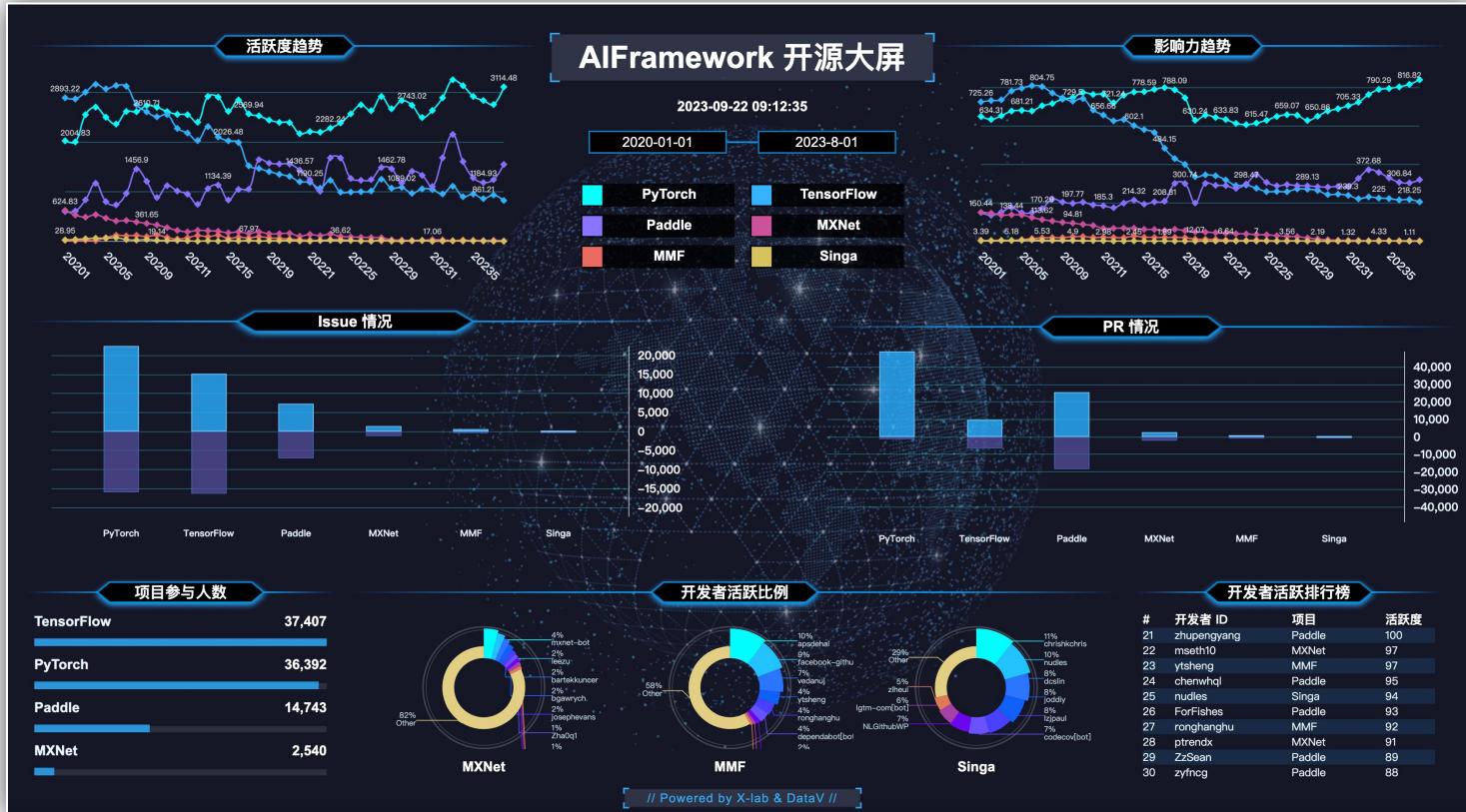
单项目开源数据大屏



多项目开源数据大屏



开源人工智能框架演化分析



THANK



YOU