交通大数据

数据预处理与探索性数据分析

- 郭延永
- guoyanyong@seu.edu.cn

数据预处理与探索性数据分析

❑ 学习目标及内容

1. 数据预处理

- ✓ 数据质量分析
- ✓ 缺失值处理
- ✓ 异常值处理
- ✓ 数据标准化处理

2. 时空数据分析基础

- ✓ 时空坐标系转化
- ✓ 时空单元划分
- ✓ 时空特征提取
- ✓ 案例解析——滴滴网约车轨迹数据分析

3. 探索性数据分析

- ✓ 数据分布特征
- ✓ 统计量分析
- ✓ 数据周期性分析
- ✓ 数据相关性分析

数据预处理与探索性数据分析

针对多源、非结构化和高维交通大数据等海量数据，不能直接进行建模处理，必须分为“探索”和“证实”两个阶段进行分析。探索阶段，就是本章的核心——探索性数据分析

❑ 数据预处理

数据获取手段

- **人工采集**数据是人工使用仪器对研究对象进行观测和记录。
- **自动采集**是由传感器自动测量，并将测量信息自动转换成数字信号，传送至终端数据存储设备。

☞ **采集到的数据存在哪些问题？可能会造成那些后果？数据预处理的重要性？**

✍ 本节将介绍交通大**数据预处理的四个核心阶段：**

数据质量分析——缺失值处理——异常值处理——数据标准化处理

数据预处理与探索性数据分析

本章的核心——探索性数据分析

❑ 数据预处理——数据质量分析

数据质量分析是预处理的重要一环，也是确保数据分析结论有效性和准确性的基础。数据质量分析的主要任务是检查是否存在**“脏数据”**。

脏数据：指不符合要求、不能直接进行相应分析的数据，包括**缺失值、异常值、格式不一致的值、重复数据及含有特殊符号**（如#、￥、*）的数据等。

1. 缺失值检查

缺失值主要包括数据记录的整体缺失，以及记录中某些字段信息的缺失。这两者都会造成计算中断或者分析结果的不准确。

（1）缺失值产生原因有：

①交通数据中部分信息**无法获取**，或者获取信息的代价太大。

②部分信息由于某些特定原因被**遗漏**。

③在某些情况下，缺失值并不意味着数据有错误，而是由于某些属性值对一些对象来说是不存在的。

数据预处理与探索性数据分析

本章的核心——探索性数据分析

❑ 数据预处理——数据质量分析

（2）缺失值的影响有：

①缺失值本身可能丢失了大量的有用信息；

②缺失值使数据分析时所表现出来的不确定性更加显著，其隐藏的规律更难把握；

③包含空值的数据可能会使数据分析过程陷入混乱，导致输出存在误差或者输出没有具有数字意义的值。

2. 异常值检查

异常值是指录入错误或者不合常理的数据，异常值通常明显偏离大部分观测值的取值范围。？

（1）异常值产生原因有

人为原因包括**数据输入错误、测量误差、实验误差、抽样错误**等因素，会对后续分析带来较大的影响。

自然原因造成的异常值，也称为**离群点**，其在一定程度上反映了数据集的分布特征。

数据预处理与探索性数据分析

本章的核心——探索性数据分析

❑ 数据预处理——数据质量分析

2. 异常值检查

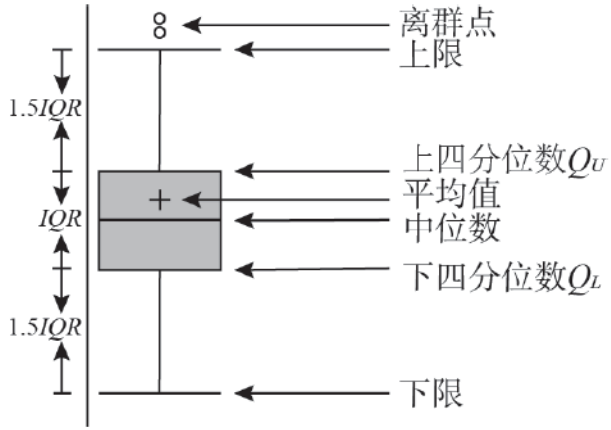
（2）异常值检查方法

① **基础统计量分析**。先对变量做一个“描述性统计”分析，得到一些基础统计量，从而查看哪些数据是不合理的。最常用的统计量是最大值和最小值。

② **3σ原则**。如果数据服从正态分布，在3σ原则下，异常值被定义为一组测定值中与平均值的偏差超过3倍标准差的值。

③ **箱型图分析**。

箱型图识别异常值的标准是：小于Q_L - 1.5IQR 或大于Q_U + 1.5IQR 的值，如图所示。



④**一致性检查**。是指检查数据间是否存在互相矛盾、冲突、不相关等不一致的情况。

数据预处理与探索性数据分析

本章的核心——探索性数据分析

❑ 数据预处理——缺失值处理

处理缺失值的方法包括**删除记录**和**数据插补**。

1. 删除记录

假设某条样本记录的大部分有效数据都缺失了，且插补操作可能改变该条记录原本的含义，所以常常将之直接删除[5-7]。

2. 缺失值插补

对于分析中必不可少的缺失值，需要进行插补填充，常见的缺失值插补方法如表所示。

插补方法	方法描述	插补函数
均值/中位数/众数	根据属性值的类型、用该属性值的平均数/中位数/众数进行插补	fillna(mean())、fillna(median())
固定值插补	将缺失的属性值用一个常量替换	fillna(a)(a为任意常数值)
最近邻插补	在记录中找到与缺失样本最接近的样本的该属性值插补	fillna(method='pad')/ fillna(method='bfill')
插值法	是利用已知点建立合适的插值函数$f(x)$,未知值由对应点x_i求出的函数值$f(x_i)$近似替代	-

数据预处理与探索性数据分析

本章的核心——探索性数据分析

❑ 数据预处理——缺失值处理

2. 缺失值插补

在这里着重介绍一下两种多项式插值法的原理与实现。

（1）拉格朗日插值法。

拉格朗日插值成功地用构造插值基函数的方法解决了求n 次多项式插值函数问题。它的基本思想是将待求的n 次多项式插值函数改写成另一种表示方式，再利用插值条件确定其中的待定函数，从而求出插值多项式。对于平面上已知的n 个点(x_1, y_1),(x_2, y_2)…(x_n, y_n)，可以找到一个n - 1 次多项式 $y = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$ ，使此多项式曲线过这n 个点，通过代入未知点的x_{n+1}，可求得插补值$\hat{y}_{n+1}$。

数据预处理与探索性数据分析

本章的核心——探索性数据分析

❑ 数据预处理——缺失值处理

2. 缺失值插补

具体步骤：

①求已知的过n 个点的n - 1 次多项式：

将n 个点的坐标(x_1, y_1),(x_2, y_2)…(x_n, y_n)代入多项式函数，得：

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$$

$$y_1 = a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_{n-1} x_1^{n-1}$$

$$y_2 = a_0 + a_1 x_2 + a_2 x_2^2 + \cdots + a_{n-1} x_2^{n-1}$$

$$\cdots$$

$$y_n = a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_{n-1} x_n^{n-1}$$

数据预处理与探索性数据分析

本章的核心——探索性数据分析

❑ 数据预处理——缺失值处理

2. 缺失值插补

（1）拉格朗日插值法。

具体步骤：

解出拉格朗日插值多项式为：

$$
\begin{aligned}
L(x)=&y_1\frac{(x-x_2)(x-x_3)\cdots(x-x_n)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_n)}\\
&+y_2\frac{(x-x_1)(x-x_3)\cdots(x-x_n)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_n)}\\
&\cdots\\
&+y_n\frac{(x-x_1)(x-x_2)\cdots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\cdots(x_n-x_{n-1})}\\
=&\sum_{i=0}^{n}y_i\prod_{j=0,j\neq i}^{n}\frac{x-x_j}{x_i-x_j}
\end{aligned}
$$

数据预处理与探索性数据分析

本章的核心——探索性数据分析

❑ 数据预处理——缺失值处理

2. 缺失值插补

②将缺失的函数值对应的点x_{n+1}代入插值多项式得到缺失值的近似值$\hat{y}_{n+1} = L(x_{n+1})$。

拉格朗日插值公式结构紧凑，理论分析方便，考虑了全局信息，但当插值节点增减时，插值多项式就会随之变化，在实际计算中不方便。为了克服这一缺点，可以使用牛顿插值法。

数据预处理与探索性数据分析

本章的核心——探索性数据分析

❑ 数据预处理——缺失值处理

2. 缺失值插补

（2）牛顿插值法。

牛顿插值法也是多项式插值，但引入了差商的概念，相比于拉格朗日插值法优势在于它在重新引入数据时可以仅仅更新F矩阵，具有承袭性和易于变动节点的特点。两者从本质上给出的结果是一样的（相同次数、相同系数的多项式），不过表示形式不同。

①求已知的n个点对$(x_1, y_1),(x_2, y_2)\cdots(x_n, y_n)$的所有阶差商公式［5，7］：

$$f[x_1,x]=\frac{f[x]-f[x_1]}{x-x_1}=\frac{f(x)-f(x_1)}{x-x_1}$$

$$f[x_2,x_1,x]=\frac{f[x_1,x]-f[x_2,x_1]}{x-x_2}$$

$$f[x_3,x_2,x_1,x]=\frac{f[x_2,x_1,x]-f[x_3,x_2,x_1]}{x-x_3}$$

$$\cdots\cdots$$

$$f[x_n,x_{n-1},\cdots,x_1,x]=\frac{f[x_{n-1},\cdots,x_1,x]-f[x_n,x_{n-1},\cdots,x_1]}{x-x_n}$$

数据预处理与探索性数据分析

本章的核心——探索性数据分析

❑ 数据预处理——缺失值处理

2. 缺失值插补

（2）牛顿插值法。

联立以上差商公式建立如下：

$$\begin{aligned} f(x)=&f(x_1)+(x-x_1)f[x_2-x_1]+(x-x_1)(x-x_2)f[x_3,x_2,x_1]+\cdots\\ &+(x-x_1)(x-x_2)\cdots(x-x_{n-1})f[x_n,x_{n-1},\cdots,x_2,x_1]+\\ &+(x-x_1)(x-x_2)\cdots(x-x_n)f[x_n,x_{n-1},\cdots,x_1,x]\\ =&P(x)+R(x) \end{aligned}$$

其中：

$$\begin{aligned} P(x)=&f(x_1)+(x-x_1)f[x_2-x_1]+(x-x_1)(x-x_2)f[x_3,x_2,x_1]+\cdots\\ &+(x-x_1)(x-x_2)\cdots(x-x_{n-1})f[x_n,x_{n-1},\cdots,x_2,x_1]\\ R(x)=&(x-x_1)(x-x_2)\cdots(x-x_n)f[x_n,x_{n-1},\cdots,x_1,x] \end{aligned}$$

$P(x)$是牛顿插值逼近函数，$R(x)$是误差函数。

②将缺失的函数值对应的点x_{n+1}代入插值多项式得到缺失值的近似值$\hat{y}_{n+1}=f(x_{n+1})$。

数据预处理与探索性数据分析

本章的核心——探索性数据分析

❑ 数据预处理——缺失值处理

2. 缺失值插补

（3）插值法在网约车轨迹数据的应用。

案例（某订单部分轨迹数据）数据如表，是网约车轨迹数据DATASET-A中某订单的部分经纬度信息。传感器记录的时间间隔一般为3秒左右，但有时会因为信号不佳等原因产生缺失值（时间间隔大于等于6秒则被定义为缺失值）。

	lat	lon	timestamp
0	30.72724	104.0751	1477969147
1	30.72702	104.0751	1477969150
2	30.72672	104.075	1477969154
3	30.7263	104.075	1477969156
4	30.72582	104.075	1477969159
5	30.72544	104.075	1477969162
6	30.72487	104.0749	1477969168
7	30.72456	104.0748	1477969171
8	30.72434	104.0746	1477969174
9	30.72406	104.0743	1477969177
10	30.72379	104.0742	1477969180
11	30.72351	104.0742	1477969183
12	30.72315	104.0745	1477969186
13	30.723	104.0748	1477969189
14	30.72298	104.0751	1477969193

（3）插值法在网约车轨迹数据的应用。在此场景下，可以通过多项式插值的方法进行填充。通过以下代码处理，使用拉格朗日插值法。

```
1. import pandas as pd
2. import numpy as np
3. # 拉格朗日插值
4. from scipy.interpolate import lagrange #scipy.interpolate是内置工具包
5. def ploy (s,n,k=5):
6. y=s[list(range(n-k,n))+list(range(n+1,n+1+k))] #取出插值位置前后k个数据
7. y=y[y.notnull()] #剔除空值
8. return lagrange(y.index,list(y))(n)
9. traj = pd.read_csv(′ DATASET-A.csv′ , header=None, usecols=[2,3,4]).iloc[:15]
10. traj.columns = [‘timestamp’,’lon’,’lat’]
11. traj[‘time_interval’] = traj[‘timestamp’]-traj[‘timestamp’].shift(1)
12. index = traj[traj[‘time_interval’] >=6].index.to_list()
13. for i in index:
14. timestamp = traj[‘timestamp’].loc[i-1] + 3
15. insertRow = pd.DataFrame([[np.nan, np.nan, timestamp]],
columns=[‘lon’,’lat’,’timestamp’])
16. traj = pd.concat([traj[:i], insertRow, traj[i:]], ignore_index=True)
17. traj[‘lon’][i]=ploy(traj[‘lon’],i)
18. traj[‘lat’][i]=ploy(traj[‘lat’],i)
19. traj = traj.drop([‘time_interval’], axis=1)
```

2. 缺失值插补

（3）插值法在网约车轨迹数据的应用。通过以下代码处理，使用拉格朗日插值法，可以得到插值结果如表所示。

	lat	lon	timestamp
0	30.72724	104.0751	1477969147
1	30.72702	104.0751	1477969150
2	30.72672	104.075	1477969154
3	30.7263	104.075	1477969156
4	30.72582	104.075	1477969159
5	30.72544	104.075	1477969162
6	30.72516	104.0749	1477969165
7	30.72487	104.0749	1477969168
8	30.72456	104.0748	1477969171
9	30.72434	104.0746	1477969174
10	30.72406	104.0743	1477969177
11	30.72379	104.0742	1477969180
12	30.72351	104.0742	1477969183
13	30.72315	104.0745	1477969186
14	30.723	104.0748	1477969189
15	30.72298	104.0751	1477969193

数据预处理与探索性数据分析

本章的核心——探索性数据分析

❑ 数据预处理——异常值处理

在数据预处理时，可以根据“数据质量分析”章节内容对数据中的异常值进行检测和识别。但对于异常值的处理方法，则需视其产生原因及其对数据分析任务和模型产生的影响而定，也需结合实际情况考虑，没有固定统一的方法。常见的异常值处理方法见表。

异常值处理方法	方法描述
删除含有异常值的记录	直接将含有异常值的记录删除
视为缺失值	将异常值视为缺失值，利用缺失值处理的方法进行处理
不处理	直接在具有异常值的数据集上进行数据挖掘和分析

数据预处理与探索性数据分析

本章的核心——探索性数据分析

❑ 数据预处理——数据标准化

数据标准化（规范化或归一化）处理，是数据挖掘的一项非常重要的基础工作。因为不同类型的数据往往具有不同的量纲，数值间的差别可能很大。

在交通大数据分析问题中，常用的数据标准化方法有三种：**最小-最大标准化**、**零均值标准化**和**小数定标标准化**。

1. 最小-最大标准化

最小-最大标准化也称为离差标准化，是对原始数据的线性变换，将数值映射到[0,1]之间。转换公式如下：

$$x^* = \frac{x - \min}{\max - \min}$$

其中，max为样本数据的最大值，min为样本数据的最小值，max-min为极差。离差标准化保留了原来数据中存在的关系，是消除量纲和数据取值范围影响的最简单方法。这种处理方法的缺点是：如果数据集中且某个数值很大，则标准化后各值会接近于0，并且标准化后的数据之间将会相差不大。同时，如果遇到超过目前属性取值范围[min,max]的时候，会引起系统出错，需要重新确定min和max。

数据预处理与探索性数据分析

数据预处理——数据标准化

2. 零均值标准化

零均值标准化也称标准差标准化，是当前用得最多的数据标准化方法。经过零均值标准化处理的数据均值为0，标准差为1。变换公式如下：

$$x^* = \frac{x - \bar{x}}{\sigma}$$

其中，$\bar{x}$为原始数据的均值，σ 为原始数据的标准差。

3. 小数定标标准化

通过移动属性值的小数位数，将属性值映射到[-1,1]之间，移动的小数位数取决于属性值绝对值的最大值。变换公式如下：

$$x^* = \frac{x}{10^{\lfloor \log|x| \rfloor}}$$

最小-最大标准化可以通过sklearn中的preprocessing.MinMaxScaler()函数实现，零均值标准化可以通过preprocessing.StandardScaler()函数实现。大多数模型训练之前会进行数据标准化，具体代码实现方法可以参考后续章节内容。

数据预处理与探索性数据分析

本章的核心——探索性数据分析

❑ 时空数据分析基础

➢ **时间型数据**具有典型的序列性，根据其应用场景可以分为两种：

一类是**时间戳**（timestamp），记录的是某个特定时刻，比如GPS定位时间、IC卡刷卡时间等；

另一种是**时间间隔**（interval），与统计性特征相关，比如观测到的小时交通流量。

➢ **空间型数据**是指带有空间坐标的数据，用点（轨迹点）、线（路段）、面（城市）等基本空间数据结构来表示。

时空数据处理的基本思想是在充分理解时间数据、空间数据、其他特征数据及其关联性的基础上，通过整合、清洗和转换不同来源的时空数据，以满足特定时空特征分析与数据挖掘算法的需求。

数据预处理与探索性数据分析

❑ 时空数据分析基础

网约车轨迹数据是一类最典型的时空数据，如表

rxcrGx4c4vw-nvvpwpzIq955wDbfoxxm	mrkrDueb.tx3nwoylfyxqhjbwD6hvtst	1477969147	104.0751	30.72724
rxcrGx4c4vw-nvvpwpzIq955wDbfoxxm	mrkrDueb.tx3nwoylfyxqhjbwD6hvtst	1477969150	104.0751	30.72702
rxcrGx4c4vw-nvvpwpzIq955wDbfoxxm	mrkrDueb.tx3nwoylfyxqhjbwD6hvtst	1477969154	104.075	30.72672
rxcrGx4c4vw-nvvpwpzIq955wDbfoxxm	mrkrDueb.tx3nwoylfyxqhjbwD6hvtst	1477969156	104.075	30.7263
rxcrGx4c4vw-nvvpwpzIq955wDbfoxxm	mrkrDueb.tx3nwoylfyxqhjbwD6hvtst	1477969159	104.075	30.72582
rxcrGx4c4vw-nvvpwpzIq955wDbfoxxm	mrkrDueb.tx3nwoylfyxqhjbwD6hvtst	1477969162	104.075	30.72544
rxcrGx4c4vw-nvvpwpzIq955wDbfoxxm	mrkrDueb.tx3nwoylfyxqhjbwD6hvtst	1477969168	104.0749	30.72487
rxcrGx4c4vw-nvvpwpzIq955wDbfoxxm	mrkrDueb.tx3nwoylfyxqhjbwD6hvtst	1477969171	104.0748	30.72456
rxcrGx4c4vw-nvvpwpzIq955wDbfoxxm	mrkrDueb.tx3nwoylfyxqhjbwD6hvtst	1477969174	104.0746	30.72434
rxcrGx4c4vw-nvvpwpzIq955wDbfoxxm	mrkrDueb.tx3nwoylfyxqhjbwD6hvtst	1477969177	104.0743	30.72406
rxcrGx4c4vw-nvvpwpzIq955wDbfoxxm	mrkrDueb.tx3nwoylfyxqhjbwD6hvtst	1477969180	104.0742	30.72379

数据预处理与探索性数据分析

❑ 时空数据分析基础——空间坐标系转换

可以通过下述代码将经纬度（GCJ-02）转化为UTM坐标系下的横纵坐标。

```
1.  from utm import *
2.  from tqdm import tqdm, tqdm_pandas
3.  import pandas as pd
4.  import numpy as np
5.  import time
6.  time1 = '20161101 08:00:00'
7.  time2 = '20161101 09:00:00'
8.  stamp1 = time.mktime(time.strptime(time1, "%Y%m%d %H:%M:%S"))
9.  stamp2 = time.mktime(time.strptime(time2, "%Y%m%d %H:%M:%S"))
10. df = pd.read_csv('DATASET-A.csv', header=None)
11. df.columns=['driver_id', 'order_id', 'timestamp', 'lon', 'lat']
12. # 转换为utc+8时区
13. df.timestamp = df.timestamp + 8 * 3600
14. df = df[(df['timestamp'] >= stamp1) & (df['timestamp'] < stamp2)].reset_index(drop=True)
15. # 定义坐标转换
```

数据预处理与探索性数据分析

❑ 时空数据分析基础——空间坐标系转换

可以通过下述代码将经纬度（GCJ-02）转化为UTM坐标系下的横纵坐标。

```
16. from osgeo import osr
17. wgs84 = osr.SpatialReference()
18. wgs84.ImportFromEPSG(4326)
    #wgs-84坐标系
19. inp = osr.SpatialReference()
20. inp.ImportFromEPSG(3857)
    #Pseudo-Mercator坐标系
21. transformation = osr.CoordinateTransformation(wgs84, inp)
22. #转换坐标
23. xy = df[['lon', 'lat']].apply(lambda x: transformation.TransformPoint(x[0], x[1])[:2], axis=1)
24. # xy为一个list,每一个元素为一个tuple
25. # 转换为dataframe中的两列
26. df['x'] = [x[0] for x in xy]
27. df['y'] = [x[1] for x in xy]
```

数据预处理与探索性数据分析

❑ 时空数据分析基础——时空单元划分

时空单元划分是时空数据处理的重要方法，一方面可以高效地挖掘数据中隐含的时空特征，另一方面便于不同类别的时空数据进行融合。

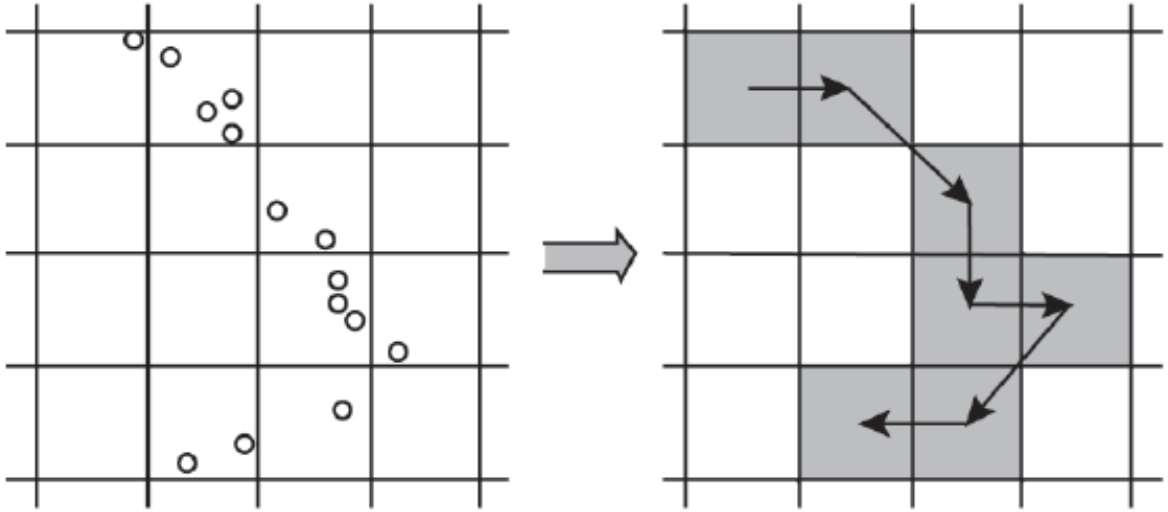
时间数据只有一维属性且具有严格的序列性，因此可以按照起始时间和间隔将时间范围划分为若干个大小相等的时间窗。

空间数据一般具有多维属性，单元之间的空间关系也更加复杂，往往需要用到网格划分的方法。如图所示，网格划分是指将区域划分为棋盘状的栅格，将空间数据点归类到各个网格中，是一种非常高效的空间数据处理方法。

数据预处理与探索性数据分析

❑ 时空数据分析基础——时空单元划分

如图所示，网格划分是指将区域划分为棋盘状的栅格，将空间数据点归类到各个网格中，是一种非常高效的空间数据处理方法。



时间	时间网络ID
00:00:00 00:00:01 00:00:02	0
……	……
00:30:01 00:30:02 00:30:03	3
……	……
08:00:01 08:00:02 08:00:03	48
……	……

数据预处理与探索性数据分析

❑ 时空数据分析基础——时空单元划分

网格划分时需要依据区域大小、数据密度与精度及分析对象的层次（中观、微观），选取合适的粒度。这里通过下述代码，计算轨迹点在时空网格中的索引，实现轨迹数据的栅格化处理。

```
#时间窗划分
time_interval=600                                                    #时间窗长度
df[ 'time_id' ] = df[ ' timestamp ' ].apply(lambda x: (x-stamp1)//time_interval)    #生成时间窗索引
#空间网格划分
left = df['x'].min()                                                 #计算左边界
up = df['y'].max()                                                   #计算上边界
interval=70                                                          #网格单元大小
df['rowid'] = df['y'].apply(lambda x: (up - x) // interval).astype('int')    #计算横向索引
df['colid'] = df['x'].apply(lambda x: (x - left) // interval).astype('int')  #计算纵向索引
```

数据预处理与探索性数据分析

❑ 时空数据分析基础——时空特征提取

1. 个体特征计算

```
1.  df = df.sort_values(by=['driver_id', 'order_id', 'timestamp']).reset_index(drop=True)
2.  # 将订单id,下移一行,用于判断相邻记录是否属于同一订单
3.  df['orderFlag'] = df['order_id'].shift(1)
4.  df['identi'] = (df['orderFlag']==df['order_id'])
5.  # 将坐标、时间戳下移一行,从而匹配相邻轨迹点
6.  df['x1'] = df['x'].shift(1)
7.  df['y1'] = df[ 'y' ].shift(1)
8.  df['timestamp1'] = df['timestamp'].shift(1)
9.  df = df[df['identi']==True]
    #将不属于同一订单的轨迹点对删去
10. dist = np.sqrt(np.square((df['x'].values-df['x1'].values)) + np.square((df['y'].values-df['y1'].values)))
    #计算相邻轨迹点之间的距离
11. time = df['timestamp'].values - df['timestamp1'].values
    #计算相邻轨迹点相差时间
12. df['speed'] = dist / time
    #计算速度
13. df = df.drop(columns=['x1', 'y1', 'orderFlag', 'timestamp1', 'identi'])
    #删去无用列
```

数据预处理与探索性数据分析

❑ 时空数据分析基础——时空特征提取

1. 个体特征计算

基于计算得到的瞬时速度，还能够继续提取更深层次的特征，如加速度，它代表了车辆的瞬时运行状态变化情况。

```
1.  df['speed1'] = df.speed.shift(1)                                   #将速度下移一行
2.  df['timestamp1'] = df.timestamp.shift(1)                           #将时间下移一行
3.  df['identi'] = df.order_id.shift(1)                                #将订单号下移一行
4.  df = df[df.order_id==df.identi]                                    #去除两个订单分界点数据
5.  df['acc'] = (df.speed1.values - df.speed.values) / (df.timestamp1.values - df.timestamp.values)
                                                                       #计算加速度
6.  df = df.drop(columns=[ 'speed1' , 'timestamp1' , 'identi' ])       #删除临时字段
```

数据预处理与探索性数据分析

❑ 时空数据分析基础——时空特征提取

2. 个体特征集计

在交通大数据分析中，常用的时空数据分析特征包括区域平均速度、流量、最小行程速度、自由流速度等，这里将详细介绍这几类特征的计算。

（1）网格平均速度V_{ins}。

对于每个时段，计算每辆车在同一网格下的平均速度，然后对此再次求平均得到网格平均速度。这么做是因为低速车辆在网格中留下的轨迹点往往更多，直接求平均速度会导致结果偏低。

```
    #基于时空网格与轨迹id进行分组
1.  orderGrouped = df.groupby(['rowid', 'colid','time_id', 'order_id'])
2.  #网格平均车速
3.  grouped_speed = orderGrouped.speed.mean().reset_index()
4.  grouped_speed = grouped_speed.groupby(['rowid' , 'colid' , 'time_id'])
5.  grid_speed = grouped_speed.speed.mean()
6.  grid_speed = grid_speed.clip(grid_speed.quantile(0.05), grid_speed.quantile(0.95))#去除异常值
```

数据预处理与探索性数据分析

❑ 时空数据分析基础——时空特征提取

2. 个体特征集计

（2）网格平均加速度acc_{mean}

对网格内单一时间段内的车辆瞬时加速度求取平均值，即得到网格平均加速度。

```
1.  #网格平均加速度
2.  gridGrouped = df.groupby([' rowid', 'colid', 'time_id'])
3.  grid_acc = gridGrouped.acc.mean()
```

（3）网格浮动车流量q_{grid}

根据每个网格中的轨迹点数量确定，同一辆浮动车的轨迹点仅计数一次。

```
1.  # 网格流量
2.  grouped_volume = orderGrouped.speed.last().reset_index()
3.  grouped_volume = grouped_volume.groupby(['rowid', 'colid', 'time_id'])
4.  grid_volume = grouped_volume[' speed'].size()
5.  grid_volume = grid_volume.clip(grid_volume.quantile(0.05), grid_volume.quantile(0.95))
```

数据预处理与探索性数据分析

❑ 时空数据分析基础——时空特征提取

2. 个体特征集计

（4）网格速度标准差std_v

速度标准差代表了网格交通流的稳定状态，当道路处于拥堵状态，车辆往往需要进行多次加减速操作，由此造成网格的瞬时速度标准差在不同路况下具有不同的特征。

```
# 网格车速标准差
grid_v_std = gridGrouped.speed.std()
```

（5）网格平均停车次数$stop_{grid}$

$$stop_{grid} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} \mathrm{II}(v_{ij}=0)}{m}$$

```
#网格平均停车次数
stopNum = gridGrouped.speed.agg(lambda x: (x==0).sum())
grid_stop = pd.concat((stopNum, grid_volume), axis=1)
grid_stop['stopNum'] = stopNum.values / grid_volume.values
grid_stop = grid_stop['stopNum']
grid_stop = grid_stop.clip(0,grid_stop.quantile(0.95))
```

数据预处理与探索性数据分析

❑ 时空数据分析基础——时空特征提取

2. 个体特征集计

（6）数据整理

将上面得到的各类特征进行整理，得到结构整齐的时空数据表。

```
1. feature = pd.concat([grid_speed, grid_acc, grid_volume, grid_v_std, grid_stop], axis=1).reset_index()
2. feature.columns = ['rowid', 'colid', 'time_id','aveSpeed','gridAcc','volume','speed_std','stopNum']
```

❑ 时空数据分析基础——网格化处理后的滴滴网约车轨迹数据

经过前边3节内容的时空数据分析后，原始轨迹数据，已经被网格化，并提取出了很多与网约车相关的二级特征，例如平均速度、平均加速度、浮动车流量等。这些特征都将为我们后续章节的建模和分析提供有力的支持。

数据预处理与探索性数据分析

❑ 时空数据分析基础——网格化处理后的滴滴网约车轨迹数据

采用上述时空数据处理方法，将成都市2016年11月1日至2016年11月30日的网约车轨迹数据分别进行处理，得到DATASET-B（除了label字段），下表（网约车轨迹数据10min 网格化特征）展示了部分数据。

rowid	colid	time_id	aveSpeed	gridAcc	volume	speed_std	stopNum	date
0	0	3	12.1093	1.755359	1	8.498806	0	20161101
0	0	52	4.992756	−0.08244	1	3.262287	0	20161101
0	0	53	4.186041	−0.16211	3	2.388629	0.3333	20161101
0	0	55	3.983647	−0.31791	2	2.353482	3.4	20161101
0	0	61	3.983647	0.039795	1	1.790372	3.4	20161101
0	0	62	3.983647	0	1	0.189662	3.4	20161101
…	…	…	…	…	…	…	…	…

数据预处理与探索性数据分析

❑ 探索性数据分析——数据分布特征分析

分布分析可以解释数据的分布特征和分布类型。在介绍分布分析之前，我们先了解两类重要的数据概念，即定量数据和定性数据：

（1）定量数据，可以认为是以数量形式存在着的属性，我们可以对其进行测量，并用数字表征。在定量数据中，又包括连续变量和离散变量，其中离散变量有时又可近似连续（如交通流量，从0到10000，虽是离散整数值，但处理时近似连续变量）；

（2）定性数据，通常是一组表示事物性质或者类别的文字表述型数据。除分类数据外，像顺序数据（如交通调查中，我们用1~5表示交通拥堵程度）也属于定性数据。

对于定量数据，如果我们想了解其分布形式是对称的还是非对称的，或发现某些极大或极小的可疑值，可通过绘制频率分布表、频率分布直方图、茎叶图等进行直观的分析；而对于定性数据，可使用饼图和条形图来显示其分布情况。

下一章将会对数据可视化内容进行详细讲解。

数据预处理与探索性数据分析

❑ 探索性数据分析——统计量分析

1. 集中趋势度量

（1）均值。均值是所有数据的平均值，假设有n 个原始数据，x_i 代表原始数据中第i个数据的值，则原始数据集均值的计算公式为：

$$\text{mean}(x)=\bar{x}=\frac{\sum x_i}{n}$$

有时，为了反映在均值中不同成分的重要程度，可以为数据集中的每一个x_i赋予权重w_i，这就得到了加权均值的计算公式：

$$\text{mean}(x)=\bar{x}=\frac{\sum w_i x_i}{\sum w_i}=\frac{w_1x_1+w_2x_2+\cdots+w_nx_n}{w_1+w_2+\cdots+w_n}$$

类似地，频率分布表的平均数可以使用下式计算：

$$\text{mean}(x)=\bar{x}=\sum f_i x_i=f_1x_1+f_2x_2+\cdots f_kx_k$$

式中，x_1，x_2，…，x_k 分别为k 个组段的组中值；f_1，f_2，…，f_k 分别为k 个组段的频率。这里的f_i起到了权重的作用。

数据预处理与探索性数据分析

❑ 探索性数据分析——统计量分析

1. 集中趋势度量

（2）中位数。中位数是将一组观察值按从小到大的顺序排序后，位于中间的那个数。即在全部数据中，小于和大于中位数数据个数相等。将某一数据集$\{x_1, x_2, \ldots, x_n\}$按照从小到大排序$\{x_{(1)}, x_{(2)}, \ldots, x_{(n)}\}$，中位数取值为：

$$M=\begin{cases} x_{\left(\frac{n+1}{2}\right)}, n\text{为奇数} \\ \frac{1}{2}\left(x_{\left(\frac{n}{2}\right)}+x_{\left(\frac{n+1}{2}\right)}\right), n\text{为偶数} \end{cases}$$

（3）众数。众数是指数据集中出现最频繁的值。众数经常用来度量定性变量的中心位置，更适合用于定性变量。此外，众数不具有唯一性，且众数一般用于离散型变量而非连续型变量。

数据预处理与探索性数据分析

❑ 探索性数据分析——统计量分析

2. 离中趋势度量

（1）极差。极差是样本最大值和最小值的差值，其对数据集的极端值非常敏感，并且忽略了位于最大值与最小值之间的数据的分布情况。

（2）标准差。标准差可以度量数据偏离均值的程度，假设包含n 个数据的数据集 $\{x_1, x_2, \ldots, x_n\}$，其标准差计算公式为：

$$s = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n}}$$

其中，x_i 代表数据集中的第i 个数据值，$\bar{x}$ 代表平均值。

（3）变异系数。变异系数可以度量标准差相对于均值的离中趋势，计算公式为：

$$CV = \frac{s}{\bar{x}} \times 100\%$$

数据预处理与探索性数据分析

❑ 探索性数据分析——统计量分析

2. 离中趋势度量

（4）四分位数间距。四分位数包括上四分位数和下四分位数。

四分位数间距，是上四分位数QU 与下四分位数QL 之差，其间包括了全部观察值的一半。其值越大，说明数据的变异程度越大；反之，说明变异程度越小。

根据DATASET-B中的车速数据，可以计算2016年11月1日第一个time_id各个空间单元平均速度的统计指标。

```
1. import pandas as pd
2. data = pd.read_csv('DATASET-B.csv')
3. data_speed = data[(data['date']==20161101) & (data['time_id']==0)]['aveSpeed']
4. statistics = data_speed.describe() #保存基本统计量
5. statistics.loc['range']=statistics.loc['max']-statistics.loc['min'] #极差
6. statistics.loc['cv']=statistics.loc['std']/statistics.loc['mean'] #变异系数
7. statistics.loc['dis']=statistics.loc['75%']-statistics.loc['25%'] #四分位数间距
```

数据预处理与探索性数据分析

❑ 探索性数据分析——统计量分析

2. 离中趋势度量

（4）四分位数间距。车速统计分析结果如表所示。

统计量	意义	数值
count	样本量	301
mean	平均值	10.94271
std	标准差	4.334557
min	最小值	3.983647
25%	下四分位数	7.550379
50%	中位数	11.43991
75%	上四分位数	14.48759
max	最大值	17.445
range	极差	13.46136
cv	变异系数	0.396114
dis	四分位间距	6.937216

数据预处理与探索性数据分析

❑ 探索性数据分析——对比分析

对比分析主要有以下两种形式[4]：

1. 绝对数比较

绝对数比较是利用绝对数进行对比，从而寻找差异的一种方法。

2. 相对数比较

相对数比较是由两个有联系的指标对比计算的，用以反映客观现象之间数量联系程度的综合指标，其数值表现为相对数。由于研究目的和对比基础不同，相对数可以分为以下几种：

（1）结构相对数：将同一总体内的部分数值与全部数值对比求得比重，用以说明事物的性质、结构或质量。

（2）比较相对数：将同一时期两个性质相同的指标数值进行对比，说明同类现象在不同空间条件的数量对比关系。如不同地区商品价格对比，不同行业、不同企业间某项指标对比等。

数据预处理与探索性数据分析

❑ 探索性数据分析——对比分析

对比分析主要有以下两种形式[4]：

（3）强度相对数：将两个性质不同但有一定联系的总量指标进行对比，以说明现象的强度、密度等。如公交线路密度用“km/km2”表示，公共自行车站点密度用“个/km2”来表示。

（4）计划完成程度相对数：某一时期实际完成数与计划数的对比，用以说明计划完成程度。

（5）动态相对数：将同一现象在不同时期的指标数值进行对比，用以说明发展方向和变化的速度。如发展速度、增长速度等。

数据预处理与探索性数据分析

❑ 探索性数据分析——周期性分析

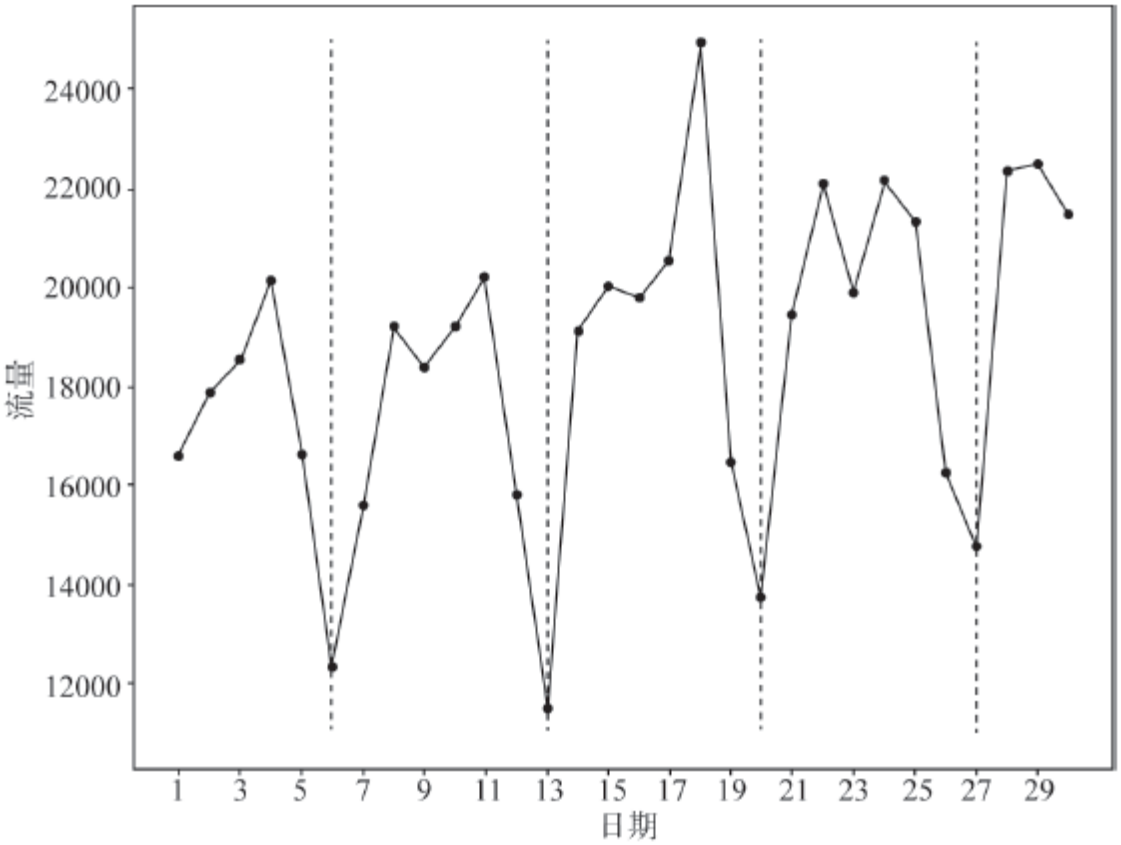
为了研究网约车流量在特定时段的周期性分布情况，可以基于DATASET-B，对每天第50个时段统计研究区域流量总和。为了研究这一变化是否与星期变化有直接关系，还计算了day字段，指的是周中日期（1~7分别代表周一至周日）。代码如下：

```
1.  import pandas as pd
2.  data = pd.read_csv('DATASET-B.csv')
3.  data_vol = data[data['time_id']==50].groupby(['date'])['volume'].sum()
4.  df = pd.DataFrame(data_vol)
5.  df = df.reset_index(drop=False)
6.  from datetime import datetime
7.  df['day'] = df['date'].apply(lambda x: datetime.strptime(str(x), "%Y%m%d").weekday()+1)
8.  # weekday()函数返回值:周一为0,周日为6
```

数据预处理与探索性数据分析

❑ 探索性数据分析——周期性分析

为了直观地展示这一变化，我们绘制了日变化折线图，如图所示，11月各日网约车流量变化情况（time_id=50）。用竖直线标定了每周日，从图中可以看出较为明显的周度周期性趋势。读者可以根据本书第4章的内容尝试自己绘制出这一图形。



数据预处理与探索性数据分析

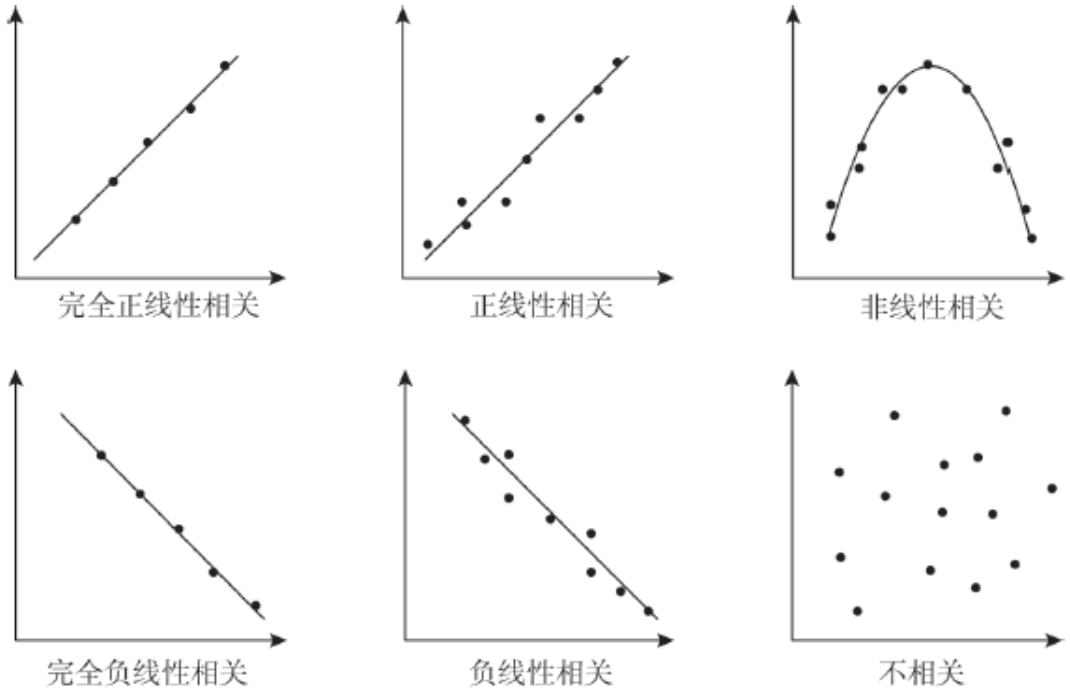
❑ 探索性数据分析——相关性分析

1. 直接绘制散点图

散点的分布特征常常有6种：

完全正线性相关（Pearson相关系数=1）、完全负线性相关（Pearson相关系数=-1）、非线性相关（-0.3<Pearson相关系数<0.3）、正线性相关（0.3<Pearson相关系数<1）、负线性相关（-1<Pearson相关系数<-0.3）、完全不相关（Pearson相关系数=0）。

具体相关关系如图：



数据预处理与探索性数据分析

❑ 探索性数据分析——相关性分析

2. 绘制散点图矩阵

需要同时考察多个变量间的相关关系时，一一绘制两两变量间的简单散点图是比较麻烦的。此时可利用散点图矩阵同时绘制各变量间的散点图，从而快速发现多个变量间的相关性，这在进行多元线性回归时是比较重要的。

3. 计算相关系数

为了更加准确地描述变量之间的线性相关程度，可以通过计算相关系数来进行相关性分析。在二元变量的相关分析过程中比较常用的有Pearson相关系数、Spearman秩相关系数和判定系数。

（1）Pearson相关系数。一般用于分析两个连续性变量之间的关系，其公式如下：

$$r=\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2\sum_{i=1}^{n}(y_i-\bar{y})^2}}$$

数据预处理与探索性数据分析

❑ 探索性数据分析——相关性分析

（1）Pearson相关系数。

$$r=\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2\sum_{i=1}^{n}(y_i-\bar{y})^2}}$$

相关系数r 的取值范围为：$-1\le r\le 1$。

$$\begin{cases} r>0\text{为正相关},r<0\text{为负相关} \\ |r|=0\text{表示不存在相关关系} \\ |r|=1\text{表示完全线性相关} \end{cases}$$

$0<|r|<1$表示不同程度线性相关：

$$\begin{cases} |r|\leqslant 0.3\text{为不存在线性相关} \\ 0.3<|r|\leqslant 0.5\text{为低度线性相关} \\ 0.5<|r|\leqslant 0.8\text{为显著线性相关} \\ |r|>0.8\text{为高度线性相关} \end{cases}$$

数据预处理与探索性数据分析

❑ 探索性数据分析——相关性分析

3. 计算相关系数

（2）Spearman秩相关系数。Pearson线性相关系数要求连续变量取值服从正态分布。不服从正态分布的变量、分类变量或等级变量之间的关联性可采用Spearman秩相关系数，也称等级相关系数描述：

$$r_s = 1 - \frac{6\sum_{i=1}^{n}(R_i - Q_i)^2}{n(n^2-1)}$$

变量x 秩次的计算过程如表所示：

x_i从小到大排序	从小到大排序时的位置	秩次R_i
0.5	1	1
0.8	2	2
1.0	3	3
1.2	4	(4+5)/2=4.5
1.2	5	(4+5)/2=4.5
2.3	6	6
2.8	7	7

数据预处理与探索性数据分析

❑ 探索性数据分析——相关性分析

3. 计算相关系数

（2）Spearman秩相关系数。

在实际应用计算中，上述两种相关系数都要对其进行假设检验，使用t 检验方法检验其显著性水平以确定其相关程度。研究表明，在正态分布假设下，Spearman相关系数与Pearson相关系数在效率上等价的，而对于连续测量数据更适合用Pearson相关系数来进行分析。

（3）判定系数。

判定系数是相关系数的平方，用r2 表示；用来衡量回归方程对y 的解释程度。判定系数取值范围：0 ≤ r^2 ≤ 1。r^2 越接近于1，表明x 与y 之间的相关性越强；r^2 越接近于0，表明两个变量之间几乎没有支线相关关系。

❑ 本章小结

- 本章重点讲解数据预处理以及探索性数据分析的理论和技术方法。首先介绍了数据预处理的重要性以及如何进行有效的数据质量分析，介绍交通大数据预处理的四个核心阶段。接着以网约车轨迹数据为例，介绍时空大数据处理与特征挖掘方法。

- 随后，以预处理后的滴滴网格数据为例，逐一介绍了常用数据分析方法的原理、作用与步骤，包括对比分析、统计量分析、周期性分析、相关性分析等。总而言之，数据分析是在发现数据基本结构的基础上，从不同层次挖掘数据中的分布特征、变化特征，分析影响因子的相关性和贡献度。

- 到目前为止，本书介绍了大数据的存储、清洗以及大数据预处理和探索性分析的相关方法，以上工作是对大数据进行建模和深入研究的基础性工作。在后面的章节将具体介绍该如何对数据进行建模、选用何种机器学习算法进行建模。。

❑ 本章参考文献

[1] 闫红伟.交通大数据在智能高速公路中的应用探讨[J].中国交通信息化,2015(3)：94-95.

[2] 刘汝焯, 戴佳筑, 何玉洁.大数据应用分析技术与方法[M].北京:清华大学出版社，2018.

[3] 陆治荣.探索性数据分析及其在流程业的应用［M］. 北京：中国石化出版社，2013.

[4] 张良均.Python数据分析与挖掘实战［M］.北京：机械工业出版社, 2016.

[5] 韦斯·麦金尼.利用Python进行数据分析［M］.3版.北京：机械工业出版社，2018.

[6] 吴翌琳，房祥忠.大数据探索性分析［M］.北京：中国人民大学出版社，2016.

[7] Nell F. Python数据分析实战［M］.北京:人民邮电出版社，2016.