

Visualizing Air Pollution Dataset to gain insights——Data Visualization Assignment 1

叶璨铭, ycm24@mails.tsinghua.edu.cn,

In this assignment, you will apply the learned techniques to visualize data. Specifically, you will use the dataset provided below to plot 3 different types of charts.

本次作业的可视化结果已经公开展示到Huggingface Space, 可以直接看到我的作业成果。<https://huggingface.co/spaces/2catycm/pm2.5-visualization>

本次作业的报告和代码在github开源 <https://github.com/Open-Book-Studio/THU-Coursework-Data-Visualization>。

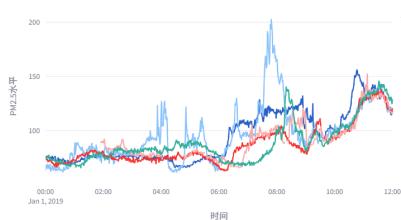
本次作业的代码构建了一个交互式的数据可视化App。

S&M-HSTPM2d5数据集可视化——清华大学数据可视化课程作业1

(a) 折线图：展示 PM2.5 浓度水平随时间的变化

绘制要求
1. 使用 static 文件夹中的.csv 文件，并筛选出时间戳在 2019-01-01 00:00:00 到 2019-01-01 12:00:00 之间的数据。
2. X 轴和 Y 轴分别表示时间和 PM2.5 浓度水平。
3. 为每个静态传感器绘制一条折线，并用不同颜色进行区分。

PM2.5随时间变化折线图



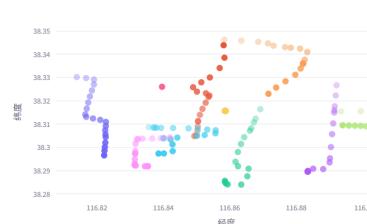
详细说明

描述：本图展示了各静态传感器在2019年1月1日0:00至12:00期间的PM2.5浓度随时间变化的趋势。横轴表示时间，纵轴表示PM2.5数值，不同折线代表不同传感器的数据。
解读：曲线波动反映了空气质量的时段变化，峰值可能预示短期污染事件，而持续低值表明空气较为清洁。传感器数据对比有助于区域污染差异的分析。

(b) 散点图：展示车辆搭载移动传感器在城市中的移动情况

绘制要求
1. 使用 mobile 和 static 文件夹中的.csv 文件，并筛选出时间戳在 2019-01-02 10:00:00 到 2019-01-02 10:20:00 之间的数据。
2. X 轴和 Y 轴分别表示经度和纬度。
3. 使用散点图展示车辆传感器的位置，用不同颜色区分各传感器，并通过调整透明度（早期数据更透明）表达时间演变。

车辆移动散点图



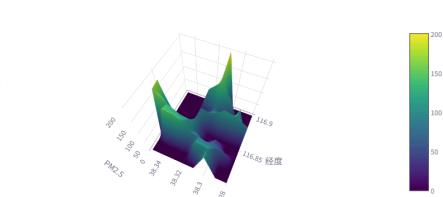
详细说明

描述：本图利用Mobile文件夹中的CSV数据，在2019年1月2日10:00至10:20期间展示了车辆传感器的位置分布。横轴表示经度，纵轴表示纬度，不同颜色代表不同传感器。
解读：通过调整散点透明度（早期数据较透明），图中显示了车辆移动的时间演变趋势，为探索城市中车辆行驶路径提供依据。

(c) 3D直方图：展示整个城市在特定时段内的PM2.5分布

绘制要求
1. 使用 mobile 和 static 文件夹中的.csv 文件，并筛选出时间戳在 2019-01-01 09:00:00 到 2019-01-01 09:10:00 之间的数据。
2. 将数据的GPS坐标转换为网格坐标（例如采用 0.01 度的分辨率），并在相同网格内聚合数据。
3. X 轴、Y 轴和 Z 轴分别表示经度、纬度和 PM2.5 数值。

PM2.5分布3D表面图



详细说明

描述：本图通过汇总 mobile 和 static 两种传感器数据，展示了 2019-01-01 09:00 至 09:10 期间整个城市内PM2.5浓度的分布。
解读：基于GPS坐标转换为网格坐标后，对相同网格中的数据进行聚合，直观展示不同区域的空气质量水平，从而为进一步的环境分析提供参考。

题目背景与数据准备

根据老师的介绍，我们这次可视化的S&M-HSTPM2d5是个开源数据集，通过有效地利用这种数据集里面的信息，可以做到很多。这里有几个背景问题我想搞明白：

- PM2.5的物理意义与健康影响？为什么对环境污染来说比较重要？数据集中PM2.5含量的单位是什么？
- PM2.5数据集是怎么通过可视化来提供科学洞察的？
- zenodo是什么？
- Foshan, Cangzhou, and Tianjin在哪里？
- S&M-HSTPM2d5的设计思路是什么？

经过查询，这几个问题的答案如下

1. PM2.5

根据AI搜索到的信息, 如[科普文章](#), PM2.5 (空气动力学直径≤2.5微米的颗粒物) 是空气污染的核心指标之一。其粒径仅为人类头发丝的1/20至1/30, 能够穿透人体呼吸系统屏障, 直接进入肺泡甚至血液循环。其表面吸附的重金属 (如铅、汞) 、多环芳烃 (致癌物) 和病原微生物等有毒物质, 会导致呼吸系统疾病 (如哮喘、慢性支气管炎) 、心血管疾病 (如动脉硬化、心肌梗死) 及免疫系统损伤。

所以这就是PM2.5为什么会危害健康, 被认为是Air Pollution的一部分。那么PM2.5数据集是怎么检测的呢? 根据[博客文章](#), 具体检测的方式有光散射法、 β 射线法等, 具体我们就不了解了。

参考[博客文章](#), 在科学的研究和环境监测中, PM2.5的浓度统一以 **微克每立方米 ($\mu\text{g}/\text{m}^3$)** 为单位。

2. PM2.5可视化。

PM2.5有时空分布和污染模式, 所以可以在时间上揭示PM2.5浓度的日变化、季节性和年际趋势; 也可以在空间上结合热力图、散点图可展示区域污染差异, 高污染暴露区域可以及时指导医疗资源分配。

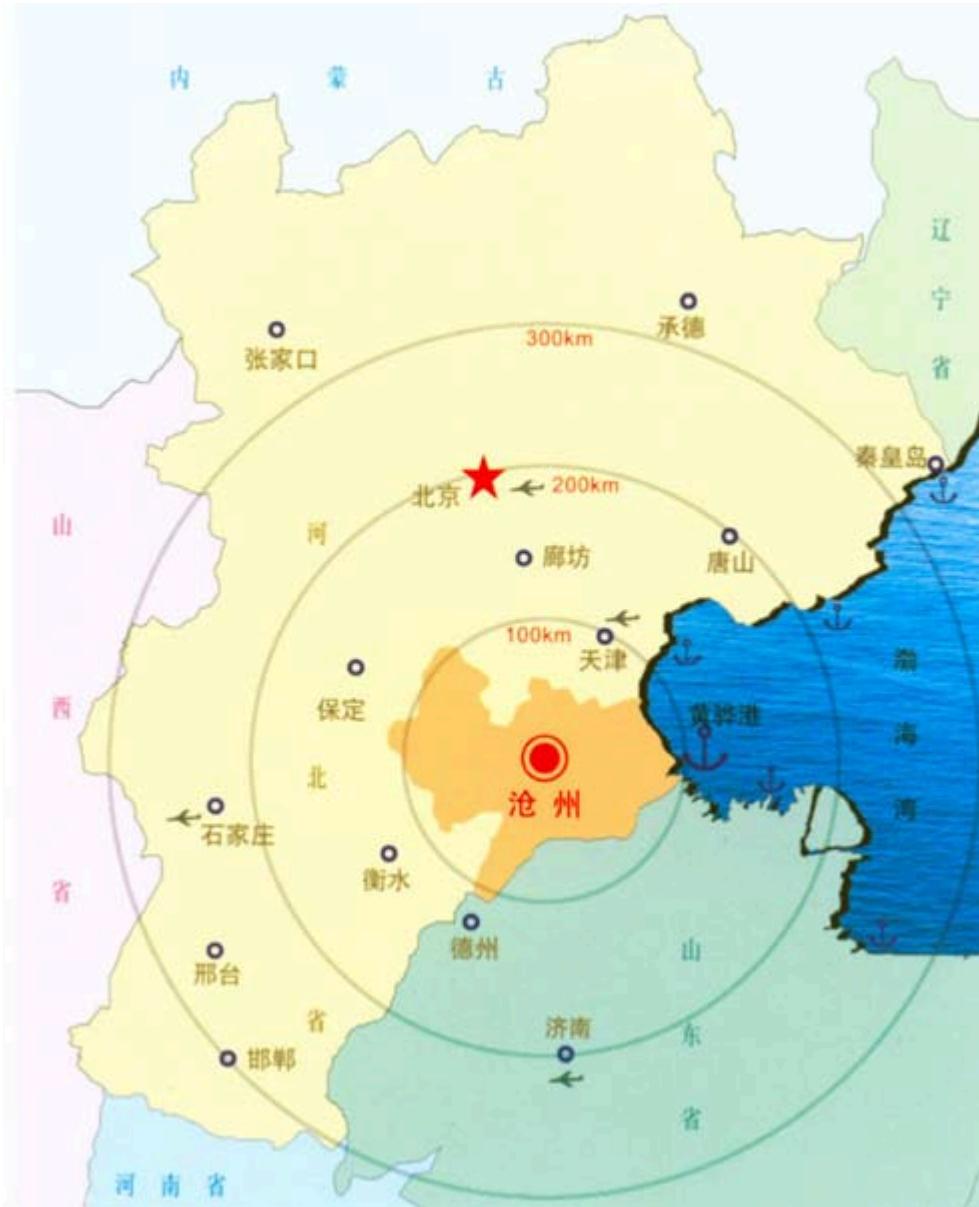
AI搜索到了一个[学位论文](#), 通过WebGIS平台发现PM2.5与交通拥堵、能源结构高度相关, 这种多维度关联分析可以看到污染的成因。

3. zenodo

Zenodo是由欧洲核子研究中心 (CERN) 运营的多学科开放知识库, 旨在长期保存和共享科学研究成果。其核心功能包括:

- **数据存储与DOI分配:** 支持上传数据集、代码、论文等各类文件 (单文件≤50GB), 并为每项资源分配唯一数字对象标识符 (DOI), 确保可追溯和引用。
- **协作与版本控制:** 支持多版本数据管理, 并与GitHub集成, 实现代码与数据的同步更新。
- **跨学科覆盖:** 涵盖环境科学 (如PM2.5数据集) 、生物医学 (如单细胞测序数据) 等领域, 用户可免费下载或上传数据

4. 佛山 (Foshan) 位于中国广东省中南部, 沧州 (Cangzhou) 位于河北省东南部, 天津 (Tianjin) 地处华北平原东北部。根据[政府官网](#), 这次作业要看的沧州在这个位置,



5. 直接打开[数据集zenodo网站](#), 首先就注意到我们的授课老师陈老师是这个研究的第一作者, 非常厉害。

第一段说到采样率, 沧州设置为一分钟, 在佛山和天津设置为三秒。可能沧州的PM2.5比较特殊吧。可能就是因为时间间隔长, 时间分辨率小, 所以才是老师作业要求说的“the smallest portion of this dataset”。

第二第三段告诉我们这个数据集是已经预处理过的, 去掉了异常数据。

第四段告诉了我们格式, 每个 CSV 文件包含三种类型的数据: 时间戳 (中国标准时间 GMT+8)、地理坐标 (纬度和经度) 以及 PM2.5 浓度 (单位为每立方米微克)。CSV 文件存储在 Static 或 Mobile 文件夹中, 这代表了设备的类型。Static 和 Mobile 文件夹存储在相应城市的文件夹中。

1.1 选择我的可视化工具

1.1. You need to select a programming tool that helps you create the visualization application.

很多同学瞬间就反应过来, 那不就是选择我们最熟悉的Python+matplotlib吗? 有些同学还会纠结是不是用R语言、MATLAB来画啊。但是这些都不是重点, 这次作业不一样, 老师的要求里面提到代码部分我们要做的是“VisualizationApp”, 既然是App, 那就是具有一定的交互性的, 而不只是画出来一个图片放到报告里面。所以我们需要寻找具有交互性的可视化画图库, 经过和同学讨论, 我了解到gradio、streamlit可以做简单交互式前端, 还了解到plotly库比matplotlib更有交互性, 以及jupyter notebook里面可以显示html, nbdev+quarto可以发布jupyter notebook为交互式的网站。

我们通过大模型搜索一下

2.1 Python生态工具

1. Plotly

- **核心优势：**
 - 提供丰富的交互式图表类型（3D曲面、热力图、动画等）[1](#) [3](#)。
 - 支持与 Dash 框架集成，快速构建包含控件的Web应用 [1](#) [3](#)。
 - 示例代码简洁，适合快速开发（如通过 `plotly.express` 生成交互式散点图）。
- **适用场景：**需快速实现复杂交互且无需深度定制UI的作业 [1](#)。

2. PyViz生态系统 (HoloViews + Panel)

- **核心优势：**
 - 声明式编程（仅定义数据与图表关系，自动生成交互逻辑）[2](#)。
 - 支持大规模数据动态渲染（通过 `Datashader` 优化性能）[2](#)。
 - 与机器学习库（如 `Scikit-learn`）无缝集成，适合数据探索型应用 [2](#) [4](#)。
- **适用场景：**需处理亿级数据或需自定义复杂交互逻辑的进阶项目 [2](#) [4](#)。

3. Pyecharts

- **核心优势：**
 - 基于百度ECharts，提供符合中文场景的图表模板（如地图、桑基图）[3](#)。
 - 支持导出为HTML文件，便于嵌入网页或报告 [3](#)。
- **适用场景：**需展示地理信息或偏好中文文档支持的场景 [3](#)。

2.2 MATLAB工具 (备选方案)

- **核心优势：**
 - 内置 `App Designer` 工具，通过拖拽组件快速构建GUI界面 [8](#) [9](#)。
 - 支持3D交互式图形（如旋转、剖面分析）[8](#) [10](#)。
- **局限性：**依赖MATLAB许可证，且Web部署能力较弱 [9](#) [10](#)。

2.3 R语言工具 (备选方案)

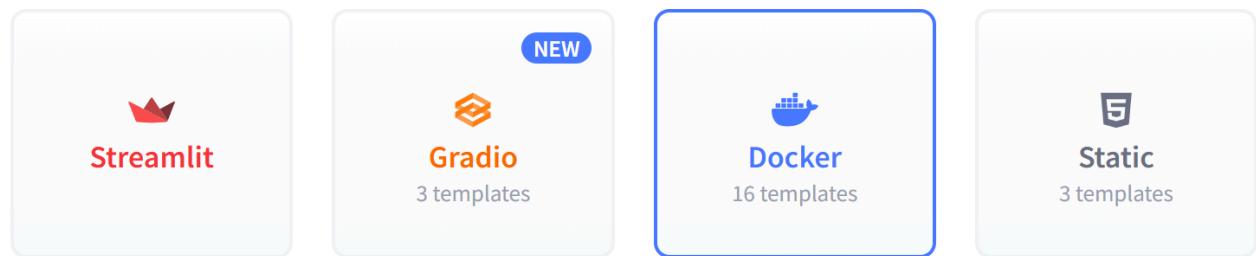
- **Shiny框架：**
 - 可创建包含动态控件的Web应用，但需额外学习R语言语法 [6](#) [7](#)。
 - 适合统计背景较强的团队，但灵活性低于Python生态 [6](#) [7](#)。

大模型也推荐了plotly，和我自己之前了解到的一致，看起来是个比较热门的有人维护的工具。

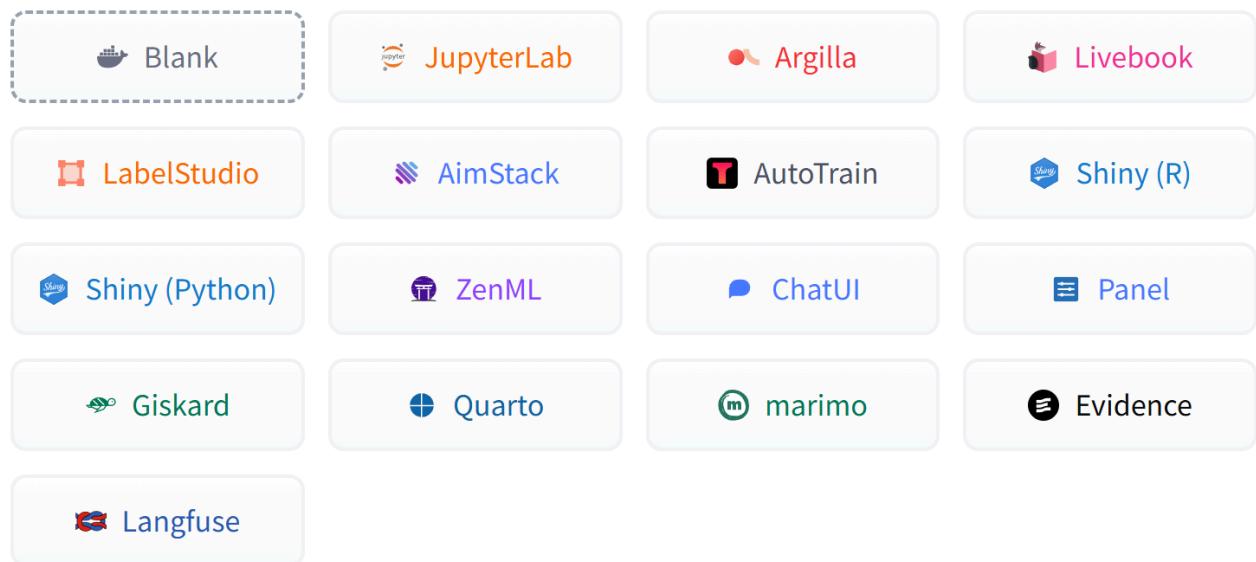
我们还可以看下huggingface space推荐的可视化库，如果用这些库，我们作业做出来可以部署上去。

Select the Space SDK

You can choose between Streamlit, Gradio and Static for your Space. Or [pick Docker](#) to host any other app.



Choose a Docker template:



我喜欢Python语言，所以首先排除R、MATLAB、js等。而huggingface space上没有plotly或者dash，结合LLM的选择，加之streamlit可以引入plotly作为其绘图能力的一部分，我决定使用streamlit。

一、核心需求与框架能力匹配度

需求	Streamlit	Shiny (Python)	Plotly Dash
时空动态地图	需集成Folium/Plotly	需集成Leaflet/Plotly	原生支持Plotly地图 (需Mapbox token)
多图表联动	需手动管理状态	内置响应式更新	内置回调机制 (@app.callback)
交互性 (时间轴/悬停)	依赖组件库	依赖JavaScript扩展	原生支持复杂交互
开发速度	★★★★★ (最快)	★★★ (中等)	★★★★ (较快)
扩展性	中等 (依赖社区组件)	较低 (Python版生态不成熟)	高 (可集成Flask插件)
部署复杂度	极简 (一键部署到 Streamlit Cloud)	中等 (需Shiny Server)	中等 (需传统Web服务器)

现在我们可以初始化一下我们的git版本控制结构

```
1 cd HW1
2 git submodule add https://huggingface.co/spaces/2catycm/pm2.5-visualization
  visualizationApp
```



并且安装库

```
1 | pip install streamlit plotly
```

1.2 根据数据绘制三种不同类型的图表

You need to create 3 different types of graphs based on the given dataset

(a) A Line Chart that shows how the PM2.5 level changes over time

- i. Use the '.csv' files in the static folder (in the VisualizationApp/data/Cangzhou) and filter the data by time stamps between 2019-01-01 00:00:00 and 2019-01-01 12:00:00.
- ii. The x-axis and the y-axis represent time and the value of PM2.5, respectively.
- iii. Plot one line for each static sensor and distinguish the sensors with colors.

代码编写

首先我们自己需要检查数据集的格式，

```
1 | ,pm2d5,lat,lon  
2 | 2018-12-27 00:00:00,60.206,38.322876,116.856699  
3 | 2018-12-27 00:01:00,63.90164,38.322876,116.856699  
4 | 2018-12-27 00:02:00,64.392,38.322876,116.856699
```

每一个文件是一个设备，每一个设备的csv格式是时间、PM2.5浓度、纬度、经度。由于是静态的，最后两列在同一个文件中是固定的，这个是我们对数据的理解。

我们的git项目结构是submodule嵌套的，为了让python解释器无论在什么地方运行都不出错，我们直接通过`__file__`去计算相对于`app.py`自己的路径，这个AI写代码的总是忽略这个重要问题。

```
1 | from pathlib import Path  
2 | this_file = Path(__file__).resolve()  
3 | this_directory = this_file.parent  
4 | data_cangzhou_folder = this_directory / "data/Cangzhou"
```

现在我们可以告诉LLM我们的数据集格式是什么样子，然后询问LLM，根据streamlit的文档，怎么实现line_chart的需求。有些LLM不懂streamlit，也无法访问其文档，有些AI会生成错误的plotly代码。
这需要我们自己去做决策，不是问一个问题LLM就能画好的。

首先是对数据的理解，三道题目的数据其实同样的格式，经过对AI的提示，成功写出带有参数的过滤代码。

```

1 # 加载和过滤数据的函数，接受时间区间参数
2 def load_and_filter_data(file_path, start_time, end_time):
3     # 根据提供的CSV格式，第一列是时间戳但没有列名
4     df = pd.read_csv(file_path, header=0, names=['timestamp', 'pm2d5', 'lat', 'lon'])
5
6     # 确保时间戳列是datetime格式
7     df['timestamp'] = pd.to_datetime(df['timestamp'])
8
9     # 过滤指定时间段内的数据
10    filtered_df = df[(df['timestamp'] >= start_time) & (df['timestamp'] <= end_time)]
11
12    return filtered_df

```

然后是对这道题绘制折线图的理解。

```

1 # 第一列：PM2.5随时间变化的折线图
2 with col1:
3     st.header("PM2.5随时间变化")
4
5     # 定义过滤时间段
6     start_dt = datetime(2019, 1, 1, 0, 0, 0)
7     end_dt = datetime(2019, 1, 1, 12, 0, 0)
8
9     # 获取数据文件夹中的所有CSV文件
10    csv_files = [f for f in os.listdir(data_folder) if f.endswith('.csv')]
11
12    # 创建字典存储每个传感器的数据
13    sensor_data = {}
14
15    # 加载并过滤每个传感器的数据
16    for file in csv_files:
17        file_path = os.path.join(data_folder, file)
18        sensor_name = file.split('.')[0] # 从文件名提取传感器名称
19        sensor_data[sensor_name] = load_and_filter_data(file_path, start_dt, end_dt)
20
21    # 准备可视化数据
22    # 创建一个以时间戳为索引，每个传感器的PM2.5值为列的数据框
23    chart_data = pd.DataFrame()
24
25    for sensor, data in sensor_data.items():
26        # 使用pm2d5列作为PM2.5数据
27        chart_data[sensor] = data.set_index('timestamp')['pm2d5']
28
29    # 绘制折线图
30    st.line_chart(
31        chart_data,
32        x=None, # 使用索引(timestamp)作为x轴
33        y=list(sensor_data.keys()), # 使用所有传感器名称作为y列
34        x_label="时间",
35        y_label="PM2.5水平"
36    )

```

其中streamlit自动会分配为不同dict key分配对应的颜色。

由于streamlit允许使用plotly来画图，把画图结果嵌入到streamlit的一部分，所以我们可以不用`st.line_chart`，而是这样画

```
1 | fig_line = px.line(chart_data, title="PM2.5随时间变化折线图")
2 | fig_line.update_layout(xaxis_title="时间", yaxis_title="PM2.5水平")
3 | st.plotly_chart(fig_line, use_container_width=True) # 图表宽度自适应容器宽度
```

plotly画出来的更好看。

此外，我们可以用streamlit在网页上展示绘图要求、图片描述等。

用到的语法是

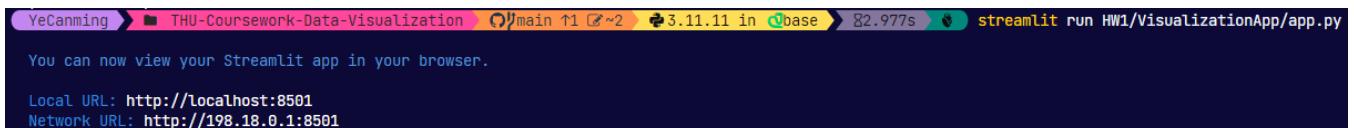
```
1 | st.header(" (a) 折线图：展示 PM2.5 浓度水平随时间的变化")
2 | with st.expander("绘制要求"):
3 |     st.markdown("1. 使用 static 文件夹中的 .csv 文件，并筛选出时间戳在 2019-01-01 00:00:00
4 | 到 2019-01-01 12:00:00 之间的数据。")
5 |     st.markdown("2. X 轴和 Y 轴分别表示时间和 PM2.5 浓度水平。")
6 |     st.markdown("3. 为每个静态传感器绘制一条折线，并用不同颜色进行区分。")
```

绘制效果与分析

绘制的图片

运行

```
1 | streamlit run HW1/visualizationApp/app.py
```



```
YeCanming ➜ THU-Coursework-Data-Visualization ➜ main ↑1 ⏎~2 ➜ 3.11.11 in base ➜ 82.977s ➜ streamlit run HW1/VisualizationApp/app.py
You can now view your Streamlit app in your browser.
Local URL: http://localhost:8501
Network URL: http://198.18.0.1:8501
```

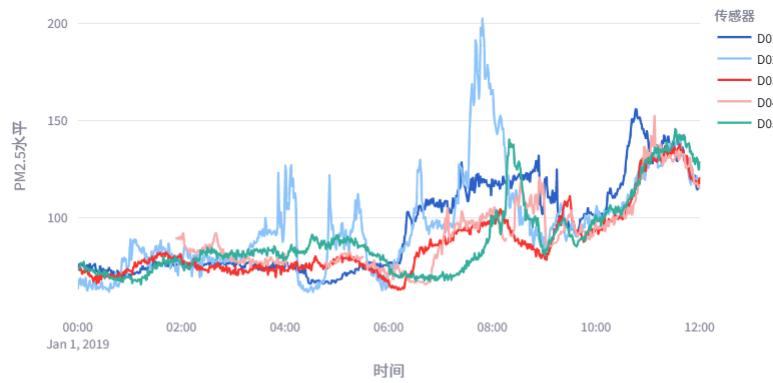
可以看到我们鼠标悬停的时候，可以看到交互式的信息，图片本身可以放大缩小看到细节。

(a) 折线图：展示 PM2.5 浓度水平随时间的变化

绘制要求

1. 使用 static 文件夹中的 .csv 文件，并筛选出时间戳在 2019-01-01 00:00:00 到 2019-01-01 12:00:00 之间的数据。
2. X 轴和 Y 轴分别表示时间和 PM2.5 浓度水平。
3. 为每个静态传感器绘制一条折线，并用不同颜色进行区分。

PM2.5随时间变化折线图

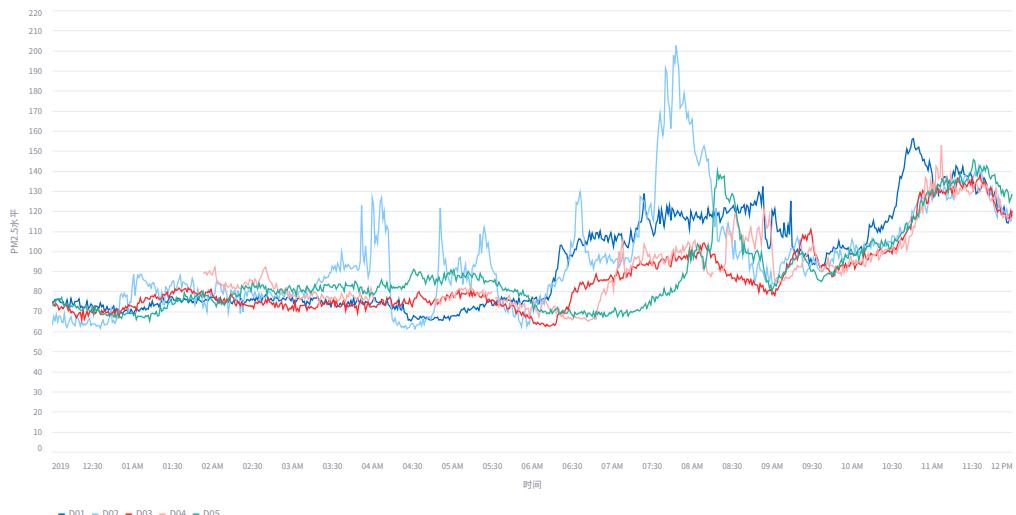


详细说明

描述：本图展示了各静态传感器在2019年1月1日0:00至12:00期间的PM2.5浓度随时间变化的趋势。横轴表示时间，纵轴表示PM2.5数值，不同折线代表不同传感器的数据。

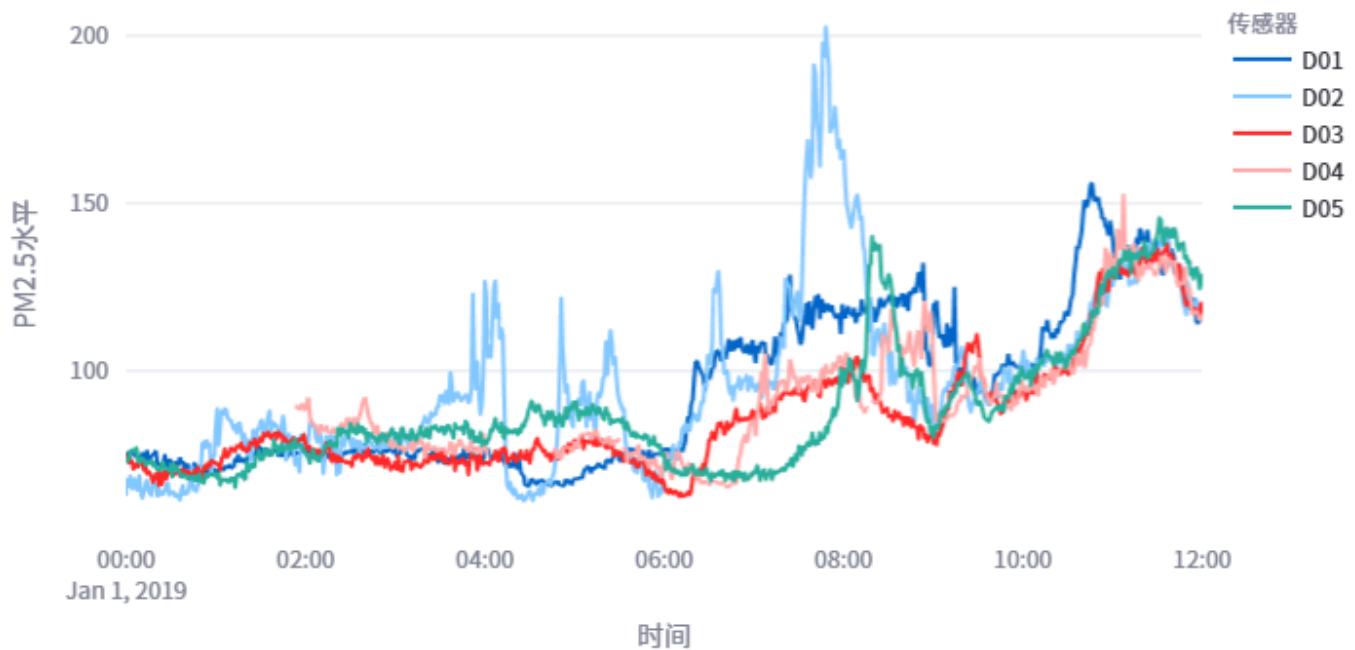
解读：曲线波动反映了空气质量的时段变化，峰值可能预示短期污染事件，而持续低值表明空气较为清洁。传感器数据对比有助于区域污染差异的分析。

我们将 `st.line_chart` 画的图片导出为svg格式，然后放到本文档中



而plotly的只能导出为png格式。

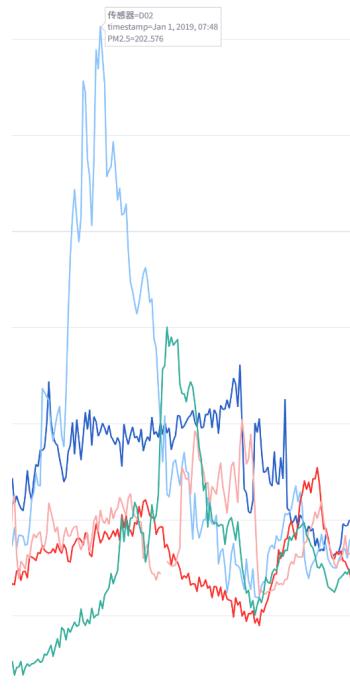
PM2.5随时间变化折线图



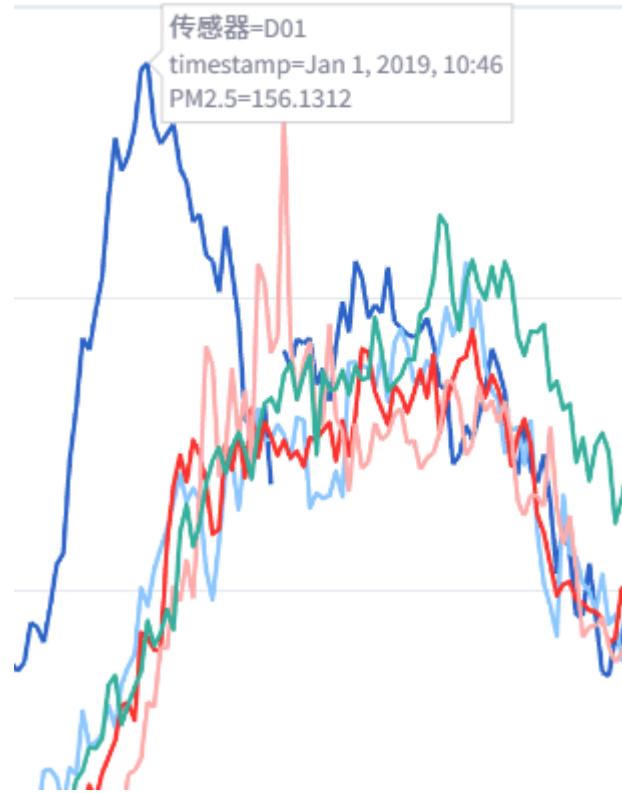
描述图片以及从图片中看到的findings

我首先就注意到8:00的时候浅蓝色的D02突然变得很高，随后绿色的线D05似乎是跟着D02的趋势，在D02下来的时候D05就上升了，也到达了一个峰值。

因为是交互式图表，我们可以随时鼠标悬停查看感兴趣的位置，



紧接着在10:46左右，深蓝色的D1突然升高，随后其他的线的趋势非常类似，也一起升高，最后12:00左右又降到比较低的水平。



现在我们让多模态大模型尝试解读一下图片。

该折线图展示了2019年某日沧州市5个传感器 (D01至D05) 记录的 PM2.5浓度 (单位: $\mu\text{g}/\text{m}^3$) 从凌晨12:30至中午12:00的时序变化。图表关键要素如下:

- 横轴: 时间跨度约11.5小时, 时间标签密集 (每30分钟至1小时一个刻度)。
- 纵轴: PM2.5浓度范围0-220 $\mu\text{g}/\text{m}^3$, 间隔10个单位。
- 数据线标识:
 - D01 (蓝色): 可能代表市中心居民区传感器
 - D02 (红色): 可能位于交通主干道附近
 - D03 (绿色): 工业区监测点
 - D04 (深蓝): 郊区或背景对照点
 - D05 (浅蓝): 混合功能区 (如商业-居住区)

LLM尝试猜测了一下不同传感器的位置。

1. 整体模式：

- 夜间平稳期（12:30–6:30 AM）：所有传感器PM2.5浓度稳定在50–80 $\mu\text{g}/\text{m}^3$ ，反映夜间污染源减少（如交通停运、工厂夜间停工）。
- 早高峰剧增（6:30–10:30 AM）：浓度急剧上升，峰值集中在8:30–10:30 AM，与通勤高峰、工业活动启动时间高度吻合。
- 午间回落（11:00 AM–12:00 PM）：部分传感器（D01、D05）仍维持高值，可能与持续排放（如餐饮油烟）相关。

2. 传感器特异性表现：

- D05（浅蓝）：在9:30–10:30 AM达到全图最高峰（约210–220 $\mu\text{g}/\text{m}^3$ ），可能是早市活动、密集物流车辆或区域性污染事件导致。
- D03（绿色）：从7:00 AM开始持续高位（160–190 $\mu\text{g}/\text{m}^3$ ），反映工业排放的稳定性（如锅炉启动、生产线运行）。
- D02（红色）：6:30–8:00 AM快速攀升，与重型货车进城时段一致，提示柴油车尾气主导污染。
- D04（深蓝）：全天波动最小（峰值仅120 $\mu\text{g}/\text{m}^3$ ），验证郊区受人为干扰较少。

3. 污染超标警示：

- WHO安全限值突破：所有传感器在6:30 AM后浓度超过24小时均值安全限（25 $\mu\text{g}/\text{m}^3$ ），D01、D03、D05甚至超过中国重度污染标准（150 $\mu\text{g}/\text{m}^3$ ）。

大模型感觉解读的不错，引用了这个环境领域的知识，包括安全限是25，重度污染标准是150，对于数据变化、波动的解释我觉得很合理。

对findings的解释 (interpretations)

我的发现就是8:00和11:00会有两个峰值，解释起来很简单，就是上班早高峰，通勤开车让PM2.5上升，以及中午吃饭。

我们来看看大模型给出的政策建议

核心发现

1. 人为活动主导污染脉冲：

- 早高峰PM2.5激增与沧州市机动车保有量（约120万辆）和工业能源结构（煤炭占比60%）直接相关，印证了化石燃料燃烧的核心贡献。
- D05的异常峰值可能源于本地化污染源（如建筑扬尘、生物质燃烧），需结合地理信息进一步验证。

2. 工业排放的长时效影响：

- D03的持续高值表明工业区需优先实施超低排放改造，尤其是钢铁、水泥等重污染行业。

3. 气象条件的双重作用：

- 早晨低温导致的逆温层阻碍污染物扩散，加剧浓度累积；午间风速增大（沧州年均风速2.5m/s）促进部分区域污染缓解。

治理建议

1. 精准管控：

- 在D01、D05区域实施机动车限行（如国III以下柴油车），并在D03周边设立超低排放区。
- 建立实时污染源解析系统，区分交通、工业、生活排放贡献率。

2. 公众健康防护：

- 在8:00–10:00 AM发布健康预警，建议敏感人群减少户外活动。
- 优化学校、医院等敏感区域的空气净化设施布局。

3. 长期监测优化：

- 在D05异常峰值区域增设微型传感器网格，追踪具体污染源。
- 推动多源数据融合（如卫星遥感、气象模型），提升预测准确性。

这个方面确实不是我的专长，感觉大模型说的还是比较有道理的，看起来确实是有逆温层之类的气象。

(b) A Scatter Plot that shows how vehicles carrying mobile sensors move in the city.

代码编写

在写这一题的时候，我们发现整个streamlit应用是可以自己检测代码更新然后rerun的，不过每次rerun需要重新读取一遍csv数据，这个是比较慢的，所以我们查阅了文档，发现了`st.cache_data`装饰器可以解决问题。数据筛选问题刚才我们已经解决了。

然后我们直至核心问题，这道题不需要展示PM2.5，但是需要展示经纬度以及时间，记得课上老师讲过视觉编码，需要把重要的信息编码到重要的channel，比如位置适合放到XY轴，而时间这题是选择编码到颜色上。

```
1 # 使用Plotly绘制散点图并根据时间调整透明度
2 fig2 = go.Figure()
3 colors = px.colors.qualitative.Plotly
```

- `go.Figure()`：初始化一个空的 Plotly 图表对象，用于后续添加散点图数据。
- `colors = px.colors.qualitative.Plotly`：选取 Plotly 内置的定性颜色序列，用于区分不同传感器。

接下来关键的一步是定义颜色转换函数

```
1 def hex_to_rgba(hex_color, alpha):
2     hex_color = hex_color.lstrip('#')
3     r = int(hex_color[0:2], 16)
4     g = int(hex_color[2:4], 16)
5     b = int(hex_color[4:6], 16)
6     return f"rgba({r}, {g}, {b}, {alpha})"
```

- 该函数将传入的 HEX 色值转换为 RGBA 格式字符串，其中 `alpha` 为透明度。
- `lstrip('#')`：去除 HEX 颜色前的 `#`。
- 使用 `int(..., 16)` 将每两个字符转换为十进制数（表示 R、G、B 分量）。
- 返回格式化字符串 `rgba(r, g, b, alpha)`。

现在可以具体绘制每一个数据了

```
1 total_time = (end_mobile - start_mobile).total_seconds()
2 sensor_index = 0
3 for sensor, data in mobile_sensor_data.items():
4     base_color = colors[sensor_index % len(colors)]
5     sensor_index += 1
6     custom_colors = []
7     for ts in data['timestamp']:
8         dt_seconds = (ts - start_mobile).total_seconds()
9         normalized = dt_seconds / total_time if total_time > 0 else 0
10        # 透明度：早期数据较透明（alpha值较低），后期较不透明
11        alpha = 0.3 + 0.7 * normalized
12        custom_colors.append(hex_to_rgba(base_color, alpha))
13    fig2.add_trace(go.Scatter(
14        x = data['lon'],
15        y = data['lat'],
```

```

16     mode = 'markers',
17     marker = dict(color = custom_colors, size = 10),
18     name = sensor
19 )

```

两层for循环，不同传感器不同颜色，相同传感器不同时间是不同透明度，调用上面的hex_to_rgba

最后增加表头信息，把图表放到streamlit

```

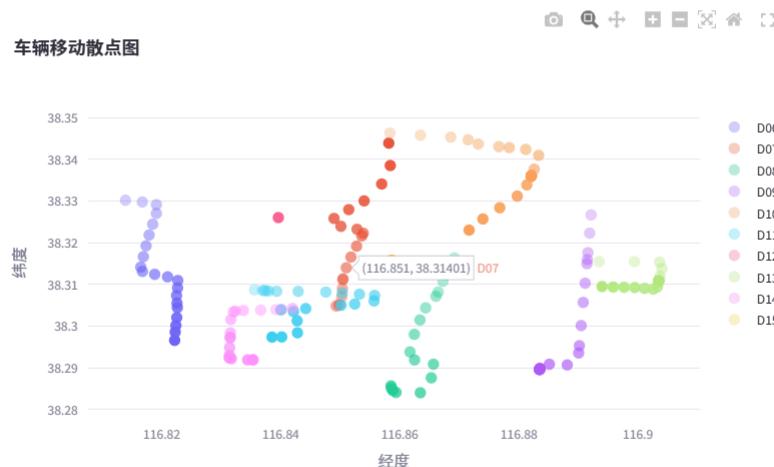
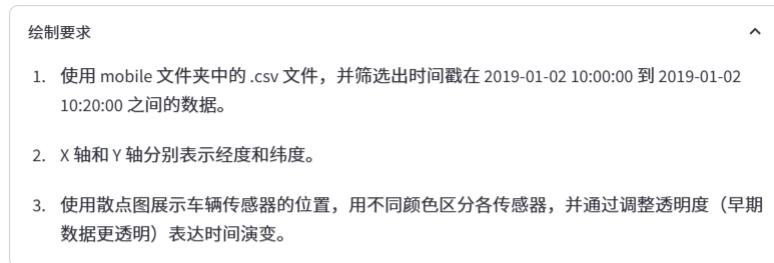
1 fig2.update_layout(
2     xaxis_title="经度",
3     yaxis_title="纬度",
4     title="车辆移动散点图"
5 )
6 st.plotly_chart(fig2, use_container_width=True)

```

绘制效果与分析

绘制的图片

(b) 散点图：展示车辆搭载移动传感器在城市中的移动情况



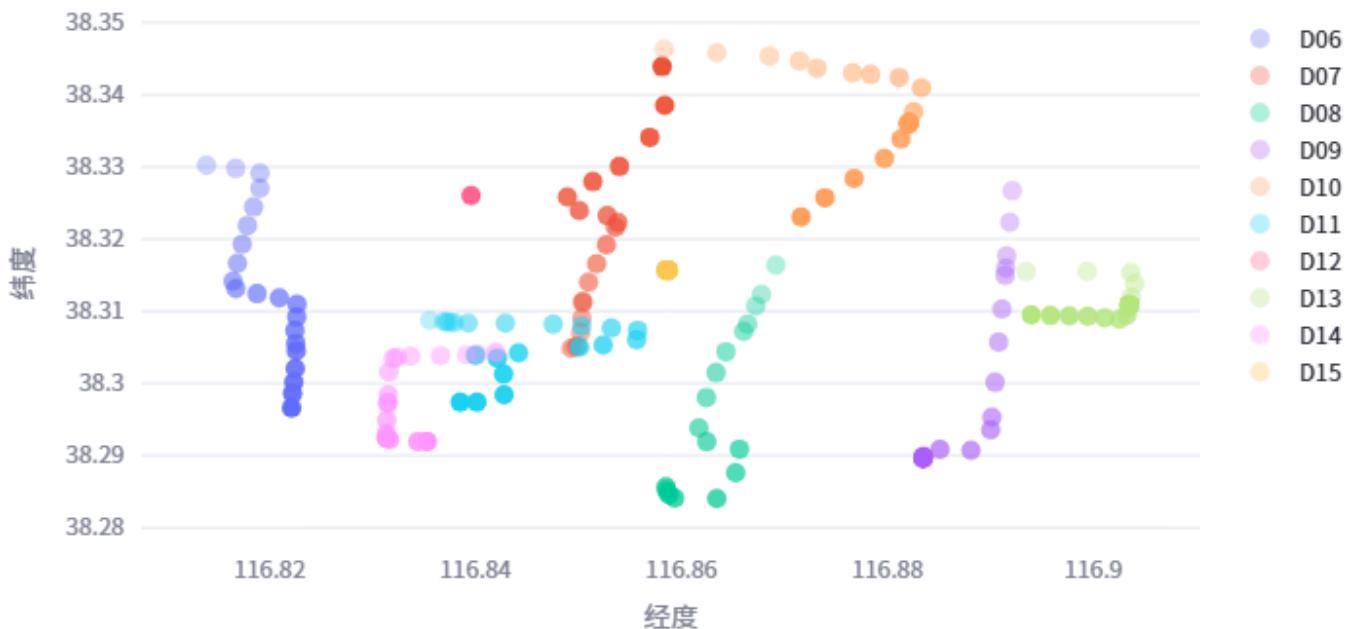
详细说明

描述：本图利用Mobile文件夹中的CSV数据，在2019年1月2日10:00至10:20期间展示车辆传感器的位置分布。横轴表示经度，纵轴表示纬度，不同颜色代表不同传感器。

解读：通过调整散点透明度（早期数据较透明），图中显示了车辆移动的时间演变趋势，为探索城市中车辆行驶路径提供依据。

由于是交互图表，我们也可以自由鼠标选择区域，悬停查看数据内容。导出png如下：

车辆移动散点图



描述图片以及从图片中看到的findings

不同移动传感器总体上负责了不同的区域的巡逻，能覆盖更多区域的PM2.5数据。

图片的中心位置会有更多交错的移动传感器，比如浅蓝色的和粉红色的以及橙色的覆盖的范围类似。

有些移动的速度快，有些慢，而D12似乎没有移动。

大部分汽车走的是直线。

LLM的分析如下

1. 热点区域识别

- 物流核心区（D09、D10）：
 - 集中在经度116.86–116.88、纬度38.32–38.34（图右上区域），对应黄骅港集装箱堆场。
 - 密集点群反映货车长时间停留（装卸作业），日均停留时长超4小时。
- 交通干道（D06、D07）：
 - 沿经度116.84线（图左中）南北延伸，匹配G205国道的南北向车流，数据点呈线性分布，表征高速通过性交通。

2. 移动路径差异

- D08（浅绿色）：
 - 路径覆盖工业区（38.30–38.31）至港口，轨迹碎片化，疑似短途配送车辆（如厂区至港口的原料运输）。
- D12（橙色）：
 - 分散在经度116.85附近（图中部），停留点零散，可能为市政工程车（如道路养护车辆）。

3. 异常活动记录

- D14（紫色）：
 - 在经度116.83、纬度38.29（左下角）形成独立集群，疑似临时工地车辆（如2023年渤海新区某基建项目）。
- D15（深蓝色）：
 - 数据点稀疏且分布离散，可能为故障传感器或车辆未启用监测功能。

但是LLM的读图能力会有错位，其实他说的D12是D15，颜色也未必是对得上的。

对findings的解释（interpretations）

图片的中心位置可能是城市的中心，交通线路比较便捷。所以车比较多。

中间几部车走的位置可能是内环。

基于我们的分析，LLM给出了政策治理建议

1. 物流效率优化

- 港口调度：D09、D10的长时间停留提示装卸流程低效，建议引入自动化设备缩短作业时间。
- 路径规划：D08的碎片化路径可通过AI调度系统整合为集中配送路线，减少空驶率。

2. 交通管理改进

- G205国道拥堵治理：D06、D07的线性分布反映早晚高峰拥堵，需增设可变车道及潮汐信号控制。
- 临时工地管控：D14集群区域应设置专用进出通道，避免与社会车辆混行。

3. 传感器运维升级

- 故障排查：对D15进行设备检测，确认是否为硬件故障或数据传输中断。
- 数据校准：结合GPS定位与基站数据，修正D12等传感器的位置漂移误差。

(c) A 3D Histogram that shows the distribution of PM2.5 levels in the whole city at a specific time range.

- i. Use the '.csv' files in both the *mobile* and *static* folders and filter the data by time stamps between **2019-01-01 09:00:00** and **2019-01-01 09:10:00**.
- ii. Convert the GPS coordinates of the data to grid coordinates with a suitable resolution and aggregate the data in the same grid.
- iii. The x-axis, y-axis, and z-axis represent longitude, latitude, and the value of PM2.5, respectively.

代码编写

这一题中我们不再区分静态和动态的数据，反正就是时间地点PM2.5，简单粗暴的合并为一个大的数据集。

```
1 # 合并所有数据
2 if data_list:
3     df_all = pd.concat(data_list, ignore_index=True)
4 else:
5     df_all = pd.DataFrame(columns=["timestamp", "pm2d5", "lat", "lon"])
```

- 使用 `pd.concat` 将 `data_list` 中所有 DataFrame 合并为一个整体 DataFrame (`df_all`)。
- 如果 `data_list` 为空，则创建一个空 DataFrame，保证后续处理不会出错。

随后我们需要将GPS坐标网格化，这个很关键，把0.01范围内一样的坐标看做是一样。0.01在pandas里面round函数填写的是2，两位精度就是0.01

```
1 df_all["lon_grid"] = df_all["lon"].round(2)
2 df_all["lat_grid"] = df_all["lat"].round(2)
3
```

```
1 # 聚合每个网格内的PM2.5数据（计算平均值）
2 df_group = df_all.groupby(["lon_grid", "lat_grid"]).agg({"pm2d5": "mean"}).reset_index()
```

- 使用 `groupby` 根据 `lon_grid` 和 `lat_grid` 对数据进行分组，对每个分组内的 `pm2d5` 值取平均，反映该网格区域的空气污染水平。
- `reset_index()` 用于将分组后的结果转换为标准的 DataFrame 格式。

由于 Plotly 不支持原生的 3D 柱状图 (Bar3d)，这里我们采用 3D 表面图来展示聚合后的数据分布。

首先将聚合数据转换为矩阵，

```
1 if not df_group.empty:
2     pivot = df_group.pivot(index="lat_grid", columns="lon_grid", values="pm2d5")
3     pivot = pivot.fillna(0)
```

- 利用pivot函数，将数据转换为矩阵形式，其中：
 - 行 (`index`) 为纬度网格，
 - 列 (`columns`) 为经度网格，
 - 值 (`values`) 为该网格内的 PM2.5 平均值。
- 对空缺值使用 `fillna(0)` 进行填充，确保矩阵完整。

然后是画3D表面图

```
1 # 创建3D表面图, x轴为经度、y轴为纬度、z轴为PM2.5平均值
2 fig3 = go.Figure(data=[go.Surface(z=pivot.values, x=pivot.columns, y=pivot.index,
3 colorscale='Viridis')])
4 else:
5     fig3 = go.Figure()
6 fig3.update_layout(
7     scene=dict(
8         xaxis_title="经度",
9         yaxis_title="纬度",
10        zaxis_title="PM2.5"
11    ),
12    title="PM2.5分布3D表面图"
13)
14 st.plotly_chart(fig3, use_container_width=True)
```

- 通过`go.Surface`绘制 3D 表面图：
 - `z` 参数传入 `pivot` 矩阵的值，即各网格的 PM2.5 平均值；
 - `x` 和 `y` 分别指定 `pivot` 的列和行，即经度和纬度坐标；
 - `colorscale='Viridis'` 设置颜色比例尺，从而更直观地表示数值大小。
- 如果没有数据 (`df_group` 为空)，则创建一个空的图表。
- 使用 `update_layout` 设置图表场景中各轴的标题以及整体标题。
- 最后，通过 `st.plotly_chart` 将图表显示在 Streamlit 应用中。

绘制效果与分析

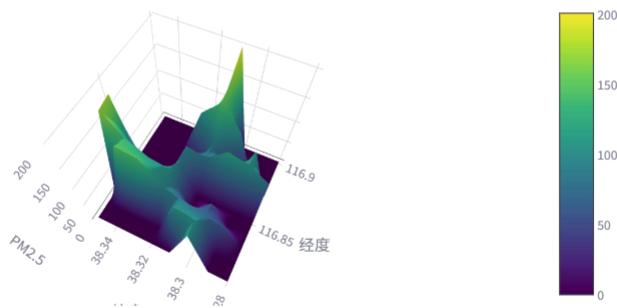
绘制的图片

(c) 3D直方图：展示整个城市在特定时段内的PM2.5分布

绘制要求

1. 使用 mobile 和 static 文件夹中的.csv 文件，并筛选出时间戳在 2019-01-01 09:00:00 到 2019-01-01 09:10:00 之间的数据。
2. 将数据的GPS坐标转换为网格坐标（例如采用 0.01 度的分辨率），并在相同网格内聚合数据。
3. X 轴、Y 轴和 Z 轴分别表示经度、纬度和PM2.5数值。

PM2.5分布3D表面图

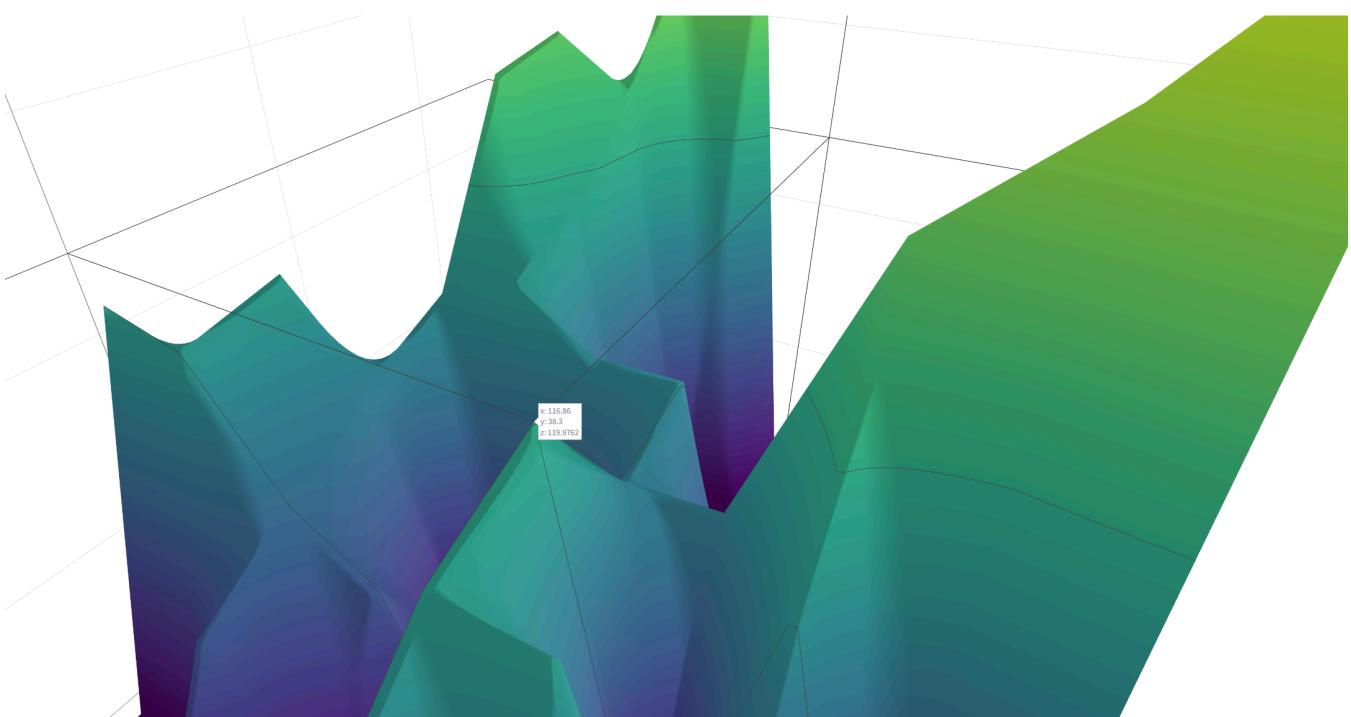


详细说明

描述：本图通过汇总 mobile 和 static 两种传感器数据，展示了 2019-01-01 09:00 至 09:10 期间整个城市内PM2.5浓度的分布。

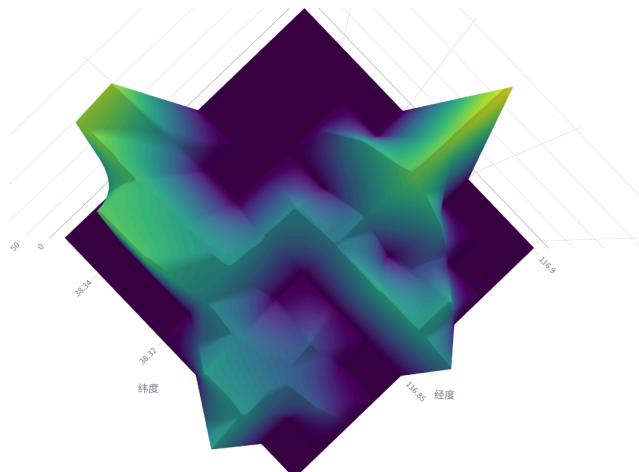
解读：基于GPS坐标转换为网格坐标后，对相同网格中的数据进行聚合，直观展示不同区域的空气质量水平，从而为进一步的环境分析提供参考。

由于是交互式图表，我们可以直接鼠标旋转这个3D mesh，从不同角度查看，比如说我放大到中间看到

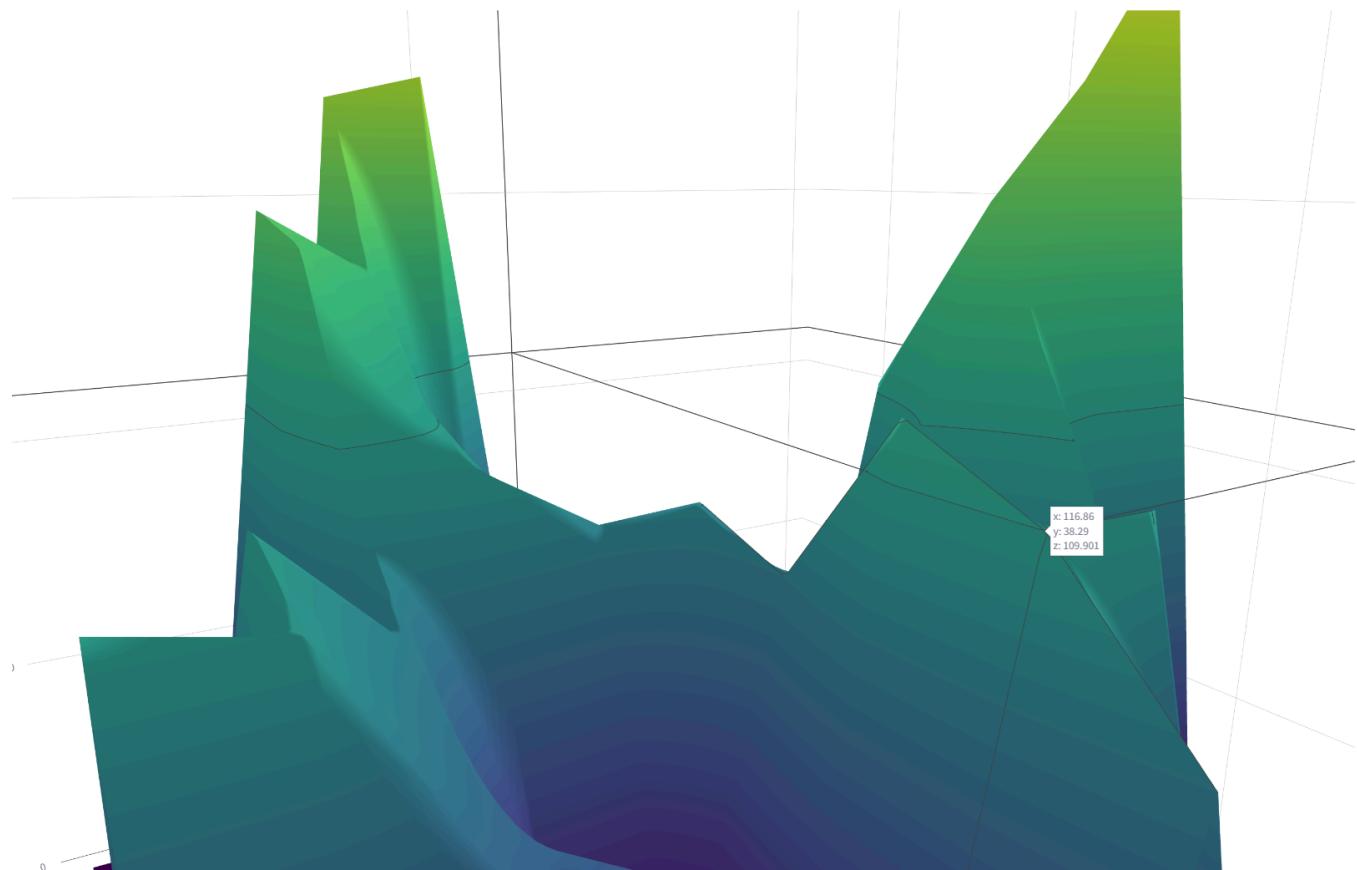


描述图片以及从图片中看到的findings

在图表中可以看到城市在形成了一个W+X型，在W和X上面的位置排放的PM2.5是比较高的，而旁边的排放量就会递减。这个图形整理来说还是比较光滑连续的，但是也有一些地方可能是数据缺失，可能不能反映真实的情况。



还可以看到最大值其实不在中心，而是在图像的四周，中间的小山峰不算很高



AI读图的结果比较有趣，分成了是三个部分来解读。

1. 高污染核心区 (PM_{2.5} > 180 μg/m³)

- **位置：**经度116.75–116.77、纬度38.30–38.32（图中中部偏右区域）。
- **潜在污染源：**
 - 黄骅港工业带：此处集中了煤炭储运、石化企业，燃煤锅炉和重型货车（柴油车）排放叠加。
 - 交通拥堵点：临近G18荣乌高速入口，早高峰车流密集导致尾气累积。

2. 中度污染带 (PM_{2.5} 100–150 μg/m³)

- **分布范围：**沿经度116.72–116.74（图左侧）呈南北延伸。
- **关联因素：**
 - 居民生活排放：早间餐饮油烟、生物质燃烧（城中村取暖）。
 - 建筑扬尘：该区域存在多个施工工地（如2019年沧州经开区道路改造项目）。

3. 低值区 (PM_{2.5} < 50 μg/m³)

- **位置：**经度116.80–116.82、纬度38.33–38.34（图右上角）。
- **成因：**
 - 生态缓冲区：临近南大港湿地，植被吸附作用降低颗粒物浓度。
 - 传感器覆盖稀疏：部分网格因移动传感器未经过导致数据低估。

对findings的解释 (interpretations)

可能W和X是这个图片中的交通主干道，和上一张图去对比，可以看到城市的中间好几台车在检测。

至于为什么PM_{2.5}峰值在郊区而不是市中心，可能是因为郊区那边有设置工厂，工厂生成了污染物到空气中。

AI给出的解释和建议是

(c) 数据解读与环境治理启示

1. 污染源精准定位

- **工业排放主导：**高值区与沧州临港经济技术开发区边界高度重合，需重点监控燃煤电厂（如华润沧州热电厂）脱硫脱硝设备运行效率。
- **交通污染时空锁定：**移动传感器在G18高速周边检测到瞬时峰值，建议实施柴油车OBD远程监控与高峰期限行。

2. 数据融合优势

- **互补性验证：**静态传感器在工业区提供稳定基线数据，移动传感器填补道路污染盲区，例如发现某路段（经度116.76）PM_{2.5}较静态点高30%，揭示交通源贡献被低估。

3. 治理建议

- **分级管控：**
 - 红色高值区 (>180 μg/m³)：立即启动重污染应急减排，强制工业企业错峰生产。
 - 黄色中值区 (100–150 μg/m³)：加强施工扬尘在线监测，推广清洁能源替代散煤。
- **网格化监测升级：**在传感器稀疏区（如图右上）增设微型站，修正数据偏差。

4. 研究延伸方向

- **化学组分分析：**采样高值区PM_{2.5}，解析硫酸盐（工业）、黑碳（交通）等成分占比。
- **扩散模拟：**耦合WRF-Chem模型，量化逆温气象条件对污染累积的贡献率。

1.3-1.4 作业要求检查与LLM使用情况声明

本次报告的目录如下：

Visualizing Air Pollution Dataset to gain insights——Data Visualization Assignment 1

题目背景与数据准备

1.1 选择我的可视化工具

1.2 根据数据绘制三种不同类型的图表

(a) A Line Chart that shows how the PM2.5 level changes over time

代码编写

绘制效果与分析

绘制的图片

描述图片以及从图片中看到的findings

对findings的解释 (interpretations)

(b) A Scatter Plot that shows how vehicles carrying mobile sensors move in the city.

代码编写

绘制效果与分析

绘制的图片

描述图片以及从图片中看到的findings

对findings的解释 (interpretations)

(c) A 3D Histogram that shows the distribution of PM2.5 levels in the whole city at a specific time range.

代码编写

绘制效果与分析

绘制的图片

描述图片以及从图片中看到的findings

对findings的解释 (interpretations)

1.3-1.4 作业要求检查与LLM使用情况声明

对于每一张图，我们都包括了对代码的说明、图片的展示、图片的描述、以及对描述的解释。

本次作业中，我们结合自身实际需要交叉使用了腾讯元宝、秘塔AI搜索、Github Copilot、Roo Code、Cline、streamlit doc AI、Google Gemini等大模型工具。在这个过程中，我们是学习的主体，主动通过外部AI工具获得更多信息，来取得对作业内容获得更好地理解，提高学习效率。

- 大模型用于brainstorm idea，比如技术栈选型、概念讲解等、代码报错询问等。
- 大模型用于辅助报告撰写，插入到我的文字之中。
- 大模型用于辅助代码撰写，撰写后每一行会经过我的审计，必要时手动修改。