VAST challenge 2019 Mini Challenge 1

St. Himark has been hit by an earthquake, leaving officials scrambling to determine the extent of the damage and dispatch limited resources to the areas in most need. They quickly receive seismic readings and use those for an initial deployment but realise they need more information to make sure they have a realistic understanding of the true conditions throughout the city.

In a prescient move of community engagement, the city had released a new damage reporting mobile application shortly before the earthquake. This app allows citizens to provide more timely information to the city to help them understand damage and prioritise their response. In this mini-challenge, use app responses in conjunction with shake maps of the earthquake strength to identify areas of concern and advise emergency planners. Note: the shake maps are from April 6 and April 8 respectively.

With emergency services stretched thin, officials are relying on citizens to provide them with much needed information about the effects of the quake to help focus recovery efforts. By combining seismic readings of the quake, responses from the app, and background knowledge of the city, help the city triage their efforts for rescue and recovery.

Analysis Tasks

- Emergency responders will base their initial response on the earthquake shake map. Use visual analytics to determine how their response should change based on damage reports from citizens on the ground. How would you prioritize neighborhoods for response? Which parts of the city are hardest hit?
- Use visual analytics to show uncertainty in the data. Compare the reliability of neighborhood reports. Which neighborhoods are providing reliable reports? Provide a rationale for your response.
- How do conditions change over time? How does uncertainty in change over time? Describe the key changes you see.

VAST挑战2019迷你挑战1

圣希马克遭遇地震，官员们急忙确定损失的程度，并向最需要的地区调拨有限的资源。他们迅速获取地震测量数据，并用这些数据进行初步部署，但意识到他们需要更多信息以确保对城市真实情况的合理理解。

为了及时与社区沟通，该市在地震发生前不久发布了一款新的损害报告移动应用程序。该应用程序允许市民向城市提供更及时的信息，帮助他们了解损害情况并优先处理响应。在本迷你挑战中，结合应用程序反馈与地震强度的震动图，识别出需要关注的区域并为应急规划者提供建议。注意：这些震动图分别来自4月6日和4月8日。

由于应急服务资源紧张，官员们依靠市民提供急需的信息，以帮助集中救援和恢复工作。通过结合地震测量数据、应用程序反馈，以及对城市的背景知识，帮助城市优化救援和恢复工作的优先级。

分析任务

应急响应人员将根据地震震动图确定其初步反应。

使用可视化分析确定他们的反应应根据基础设施损坏报告如何变化。你会如何优先考虑响应的社区？城市哪些地方受灾最严重？

使用可视化分析展示数据的不确定性。比较各社区报告的可靠性。哪些社区提供了可靠的报告？请为你的回答提供理由。

条件随时间如何变化？不确定性如何随时间变化？

描述你所看到的关键变化。

Visualisation App

You are expected to develop one or more interactive visualisation app(s) to answer these three questions. You can have one app for each question, or answer all three questions in one app. Each app can have one or more (linked) visualisations. Here are two example apps, and you are not allowed to reuse their design or code: example1 and example2.

Implementation

The visualisation app(s) must be implemented in JavaScript and below are some examples. However, you are not allowed to use Observable Notebook (as covered in COMP3021). Instead, you can use Observable Plot, which is a JavaScript library, for your app.

- D3.js - A relatively low-level visualisation library. A must if you want to create a new type of chart. Tutorial by Michael Oppermann, Tutorial by Scott Muarry (code examples)
- Echarts - open-source visualisation library that has many pre-made charts.
- Chart.js: another open-source library with many pre-made charts.
- Vega-Lite.js - A high-level JavaScript visualisation library that allows concise code to create interactive visualisation.
- Observable Plot: a JavaScript visualisation library that includes most of the visualisations available in Observable. And you can use Observable Framework for dashboard.

Final Deliverables:

Written report (80%)

The final deliverable is an implementation of the proposed solution (code repositories) and a report with maximum 3,000 words (not including the visualisations). There is no limit on the number of images included in the report.

The report should cover:

- Methods/Design: 1) A detailed explanation of the techniques and algorithms you used to solve the problem (including data pre-processing), and 2) the rationale behind the visualisation and system interface design (storyboard, etc.).

可视化应用

您需要开发一个或多个交互式可视化应用，以回答这三个问题。您可以为每个问题制作一个应用，或在一个应用中回答所有三个问题。每个应用可以包含一个或多个（关联的）可视化。以下是两个示例应用，您不允许重用它们的设计或代码：example1 和 example2。

实现

可视化应用必须使用 JavaScript
实现，以下是一些示例。然而，您不允许使用 Observable
Notebook（如 COMP3021 中所述）。相反，您可以使用 Observable
Plot，这是一种 JavaScript 库，来构建您的应用。

D3.js -
一种相对底层的可视化库。如果您想创建新类型的图表，这是必需的。Michael Oppermann 的教程，Scott Muarry 的教程（代码示例）

Echarts - 一个开源可视化库，具有许多预制图表。

Chart.js - 另一个开源库，具有许多预制图表。

Vega-Lite.js - 一种高级 JavaScript
可视化库，允许用简洁的代码创建交互式可视化。

Observable Plot - 一种 JavaScript 可视化库，包括 Observable
中大部分可用的可视化。同时您可以使用 Observable Framework
来创建仪表板。

最终交付成果：

书面报告（80%）

最终交付成果是对提出的解决方案（代码库）的实施，以及一份最多3,000字的报告（不包括可视化内容）。报告中包含的图片数量没有限制。

报告应涵盖：

方法/设计：1）详细解释您用于解决问题的技术和算法（包括数据预处理），以及2）可视化和系统界面设计的理由（故事板等）。

- Tool/Implementation: 1) A detailed description of all the visualisations (such as different charts included in the tool) and interactive features with screenshots, and 2) A detailed explanations on how you implement the visualisation (must include the details of all the libraries used).
- Results: detailed description of your visualisations and the answers to the three analysis tasks listed above.
- Reflection/Future Work: 1) A reflection on the project, what went well and what did, covering both the technical (such as the charts used and their implementations) and non-technical aspect (such as project management), and 2) what is still missing/can be improved and how these could be added/further enhanced.
- Appendix: an optional appendix should include all the information needed to run the tool, such as what external libraries are need and any configurations.

Marking Criteria

Methods/Design (30%)

- The effectiveness and sophistication of the data pre-processing and analysis algorithms.
- The justification of the visualisation and interaction design based on data characteristics and analysis task.

Tool/Implementation (30%)

- The sophistication and novelty of the design; new visual representations (first of its kind) and interactive dashboard are highly recommended.
- The app architecture, implementation complexity, app robustness, and interaction responsiveness.

Results (10%)

- How clear the answer is from the visualisation.
- The accuracy of the answer.

Reflection/Future Work (10%)

- The reflection should be in depth and cover all the aspects of the project.
- The future work should cover all the weaknesses of the current app(s), both visual design and implementation, and the proposed solution should be effective and novel.

Presentation (20%)

You will also need to record a presentation of your work of about 10 minutes, which should cover all the main aspects of the report (the first four bullet

工具/实现：1）对所有可视化（例如工具中包含的不同图表）和互动特性的详细描述，并附有截图，以及

2) 关于如何实现可视化的详细说明（必须包括所使用的所有库的细节）。

结果：您的可视化的详细描述以及上述三个分析任务的答案。

反思/未来工作：1）对项目的反思，哪些做得好，哪些不足，包括技术（例如使用的图表及其实现）和非技术方面（例如项目管理），以及2）仍然缺少的内容/可以改进的地方，以及如何进一步增强这些内容。

附录：可选附录应包括运行该工具所需的所有信息，例如需要哪些外部库和任何配置。

评估标准

方法/设计（30%）

数据预处理和分析算法的有效性和复杂性。

针对数据特征和分析任务，对可视化和交互设计的合理性。

工具/实现（30%）

设计的复杂性和新颖性；新的视觉表现（首创）和互动仪表板强烈推荐。

应用架构、实现复杂性、应用稳健性和交互响应性。

结果（10%）

视觉化中答案的清晰度。答案的准确性。

反思/未来工作（10%）

反思应深入并涵盖项目的各个方面。
未来的工作应涵盖当前应用的所有弱点，包括视觉设计和实现，所提议的解决方案应有效且新颖。

演示（20%）

您还需要录制大约10分钟的工作演示，内容应涵盖报告的所有主要方面（前四个要点

points above). Provide clear reference to the report where it is not possible to include all the details in the presentation.

Presentation submission:

- A copy of the presentation slides.
- A video of the visualisation app demonstration: maximum 10 minutes, in the mp4 format, no more than 250 MB

关于圣希马克地震后应急响应的问题，特别是通过可视化分析来帮助确定救援优先级、评估数据可靠性以及分析随时间变化的情况。用户提供了三个分析任务，需要开发一个或多个交互式 JavaScript 可视化应用来回答这些问题。

首先，我需要明确用户的需求。他们希望结合地震震动图、市民的 app 报告以及城市背景信息，来指导应急资源的分配。关键点在于如何整合这些数据，并通过可视化展示优先级、可靠性和时间变化。

接下来，分析每个任务的具体要求：

1.

任务一：确定响应优先级。需要比较地震震动图和市民报告，找出差异较大的区域。比如，某些区域震动强度高但报告少，可能漏报；或者震动低但报告多，可能基础设施脆弱。可能需要地图叠加图层，比如用热力图标示震动强度，点图或柱状图显示市民报告数量，并用颜色区分严重程度。交互方面，可以点击区域查看详情，或筛选不同时间的数据。

2.

任务二：评估报告可靠性。这里需要确定哪些社区的报告可靠。可能需要分析报告的一致性，比如同一区域不同时间报告的变化情况，或者与其他数据（如震动图）的吻合程度。可视化可以用折线图显示报告数量随时间的变化，或散点图比较震动强度与报告数量，离群点可能不可靠。此外，统计方法如标准差或置信区间可能有用，用误差条或箱线图展示。

3.

任务三：随时间变化的情况。需要展示损害报告和不确定性的变化。可能用时间序列图，动态地图或动画显示不同时间点的数据。不确定性可以通过透明度或置信区间来表示，比如随着时间的推移，如果报告数量增加，不确定性降低，可以用渐变色或面积图表示。

接下来考虑技术实现。用户要求使用 JavaScript 库，推荐使用 Observable Plot，因为用户提到不允许用 Observable Notebook，但允许用 Plot。同时，可能需要结合其他库如 D3.js 处理数据，或 Vega-Lite 生成交互图表。需要确保应用是交互式的，比如工具提示、动态过滤、联动视图等。

以上几点). 请明确引用报告，以便在演示中无法包含所有细节的情况下。

演示提交：

演示幻灯片的副本。
可视化应用演示视频：最大10分钟，mp4格式，不超过250 MB

关于圣希马克地震后应急响应的问题，特别是通过可视化分析来帮助确定救援优先级、评估数据可靠性以及分析随时间变化的情况。用户提供了三个分析任务，需要开发一个或多个交互式 JavaScript 可视化应用来回答这些问题。

首先，我需要明确用户的需求。他们希望结合地震震动图、市民的应用报告以及城市背景信息，来指导应急资源的分配。关键在于如何整合这些数据，并通过可视化展示优先级、可靠性和时间变化。

接下来，分析每个任务的具体要求：

1. 任务一：确定响应优先级。需要比较地震震动图和市民报告，找出差异较大的区域。例如，某些区域震动强度高但报告少，可能存在漏报；而震动低但报告多，可能基础设施脆弱。可能需要地图叠加图层，例如用热力图标示震动强度，点图或柱状图显示市民报告数量，并用颜色区分严重程度。交互方面，可以点击区域查看详情，或者筛选不同时间的数据。

2. 任务二：评估报告可靠性。这里需要确定哪些社区的报告可靠。可能需要分析报告的一致性，例如同一区域不同时间报告的变化情况，或者与其他数据（如震动图）的吻合程度。可视化可以用折线图显示报告数量随时间的变化，或散点图比较震动强度与报告数量，离群点可能不可靠。此外，统计方法如标准差或置信区间可能有用，用误差条或箱线图展示。

3. 任务三：随时间变化的情况。需要展示损害报告和不确定性的变化。可能使用时间序列图、动态地图或动画来显示不同时间点的数据。不确定性可以通过透明度或置信区间来表示。例如，随着时间的推移，如果报告数量增加，不确定性降低，可以用渐变色或面积图来展示。

接下来考虑技术实现。用户要求使用 JavaScript 库，推荐使用 Observable Plot，因为用户提到不允许使用 Observable Notebook，但允许使用 Plot。同时，可能需要结合其他库如 D3.js 来处理数据，或使用 Vega-Lite 来生成交互图表。需要确保应用是交互式的，比如工具提示、动态过滤、联动视图等。

数据结构方面，假设有地震震动图的地理数据（GeoJSON 或 TopoJSON），包含不同区域的震动强度；市民报告的数据集，包含时间戳、地理位置、损害程度、可能还有其他元数据如用户 ID（用于评估可靠性）。需要将这两种数据源整合，可能需要地理空间映射，如将报告聚合到各个社区。

对于任务一，可能需要将震动强度与报告数量进行对比。例如，每个社区的颜色表示震动强度，点的大小或数量表示报告的数量。如果某个社区震动高但报告少，可能需要优先调查；反之，如果报告多但震动低，可能结构问题。交互部分，可以悬停显示具体数值，或筛选特定日期（如 4 月 6 日和 8 日的震动图）。

任务二的可靠性评估可能需要计算每个社区报告的一致性。例如，同一社区不同用户的报告是否一致，或者报告数量是否与时间相关（如持续增加可能更可靠）。可以用折线图显示每个社区每天的报告量，波动大的可能不可靠。或者，计算每个社区报告与震动强度的相关系数，相关系数高的社区更可靠。用柱状图显示相关系数，颜色表示可靠性。

任务三的时间变化，可能需要动态图表，如时间滑块控制地图或折线图的显示。不确定性可以用误差带或阴影区域，比如每个时间点的报告数量的标准差。随着时间推移，如果报告趋于一致，不确定性降低，阴影区域变窄。

现在，考虑如何将这些整合到一个或多个应用中。可能的方案是创建一个仪表盘，包含多个联动视图。例如，左侧是地图显示震动和报告，右侧是折线图显示时间变化，底部是柱状图显示可靠性。选择某个社区时，其他视图相应更新。

技术实现步骤：

1. 数据准备：处理地震震动图数据（地理数据）和市民报告数据（CSV 或 JSON）。可能需要将报告按社区聚合，并计算每个社区的指标（如报告数量、平均值、标准差等）。
2. 地图可视化：使用 Observable Plot 或 D3.js 绘制地图。Plot 可能更简单，但定制性可能有限。如果使用 D3，可以更灵活地叠加多个图层。例如，用 GeoJSON 绘制社区边界，填充颜色表示震动强度，叠加圆圈表示报告数量，颜色表示损害程度。
3. 时间序列图：用折线图或面积图展示每个社区的报告数量随时间变化。Observable Plot 的 Line 图表可以轻松实现，并添加交互如悬停提示。
4. 可靠性图表：使用柱状图或散点图，每个柱子代表一个社区，高度表示可靠性指标（如相关系数或标准差）。颜色可以区分可靠性等级。
5.

在数据结构方面，假设存在地震震动图的地理数据（GeoJSON或TopoJSON），其中包含不同区域的震动强度；市民报告的数据集，则包含时间戳、地理位置、损害程度以及可能还有其他元数据，如用户ID（用于评估可靠性）。需要将这两种数据源进行整合，可能还需进行地理空间映射，例如将报告聚合到各个社区。

对于任务一，可能需要将震动强度与报告数量进行对比。例如，每个社区的颜色表示震动强度，点的大小或数量则表示报告的数量。如果某个社区震动强度高但报告数量少，可能需要优先调查；反之，则如果报告数量多但震动强度低，可能存在结构问题。在交互部分，可以设置悬停显示具体数值，或者筛选特定日期（如4月6日和8日的震动图）。

任务二的可靠性评估可能需要计算每个社区报告的一致性。例如，同一社区内不同用户的报告是否一致，或者报告数量是否与时间相关（如持续增加则可能更可靠）。可以用折线图显示每个社区每日的报告量，波动大的可能不可靠。或者，计算每个社区报告与震动强度的相关系数，相关系数高的社区则更可靠。用柱状图显示相关系数，颜色表示可靠性。

任务三的时间变化可能需要动态图表，例如用时间滑块控制地图或折线图的显示。不确定性可以用误差带或阴影区域表示，比如每个时间点的报告数量的标准差。随着时间推移，如果报告趋于一致，则不确定性降低，阴影区域将变窄。

现在，考虑如何将这些整合到一个或多个应用中。可能的方案是创建一个仪表盘，包含多个联动视图。例如，左侧是地图显示震动和报告，右侧是折线图显示时间变化，底部是柱状图显示可靠性。选择某个社区时，其他视图会相应更新。

技术实现步骤：

1. 数据准备：处理地震震动图数据（地理数据）和市民报告数据（CSV 或 JSON）。可能需要将报告按社区进行聚合，并计算每个社区的指标（如报告数量、平均值、标准差等）。
2. 地图可视化：使用 Observable Plot 或 D3.js 绘制地图。Observable Plot 可能更简单，但定制性可能有限。如果使用 D3.js，可以更灵活地叠加多个图层。例如，使用 GeoJSON 绘制社区边界，填充颜色表示震动强度，叠加圆圈表示报告数量，颜色表示损坏程度。
3. 时间序列图：用折线图或面积图展示每个社区的报告数量随时间变化。Observable Plot 的折线图表可以轻松实现，并添加交互功能，如悬停提示。
4. 可靠性图表：使用柱状图或散点图，每个柱子代表一个社区，高度表示可靠性指标（如相关系数或标准差）。颜色可以用于区分可靠性等级。
5.

交互功能：使用 JavaScript 事件监听，如点击地图中的社区，更新其他图表。或者添加时间滑块，过滤数据范围。

潜在的问题包括数据量过大导致性能问题，需要适当聚合；以及如何有效展示不确定性，可能需要视觉编码的权衡，比如透明度、误差线等。

最后，需要确保应用符合用户要求：不使用 Observable Notebook，但允许使用 Observable Plot 和 Framework。可能需要使用 Observable Framework 来构建仪表盘，将多个视图整合，并处理数据联动。

总结，开发步骤可能包括：

- 数据处理和聚合，生成每个社区的指标。
- 使用 Observable Plot 创建地图、折线图、柱状图。
- 添加交互功能，如联动、工具提示。
- 部署为独立的 Web 应用，可能用 Observable Framework 生成静态页面。

```
// Import libraries
import { Plot, dot, line, rect } from "@observablehq/plot";
import { legend, color } from "@observablehq/plot";

// Load data (example structure)
const shakeMap = await d3.json("shake-data.geojson");
const reports = await d3.csv("damage-reports.csv");
const neighborhoods = await d3.csv("neighborhoods.csv");

// 1. Map View: Prioritization
const map = Plot.plot({
 projection: "identity",
 color: { legend: true },
 marks: [
  Plot.geo(shakeMap, {
   fill: (d) => d.properties.intensity,
   title: (d) => d.properties.neighborhood,
  }),
  Plot.dot(reports, {
   x: "longitude",
   y: "latitude",
   fill: "severity",
```

交互功能：使用 JavaScript 事件监听，如点击地图中的社区，更新其他图表。或者添加时间滑块，过滤数据范围。

潜在的问题包括数据量过大导致的性能问题，需要适当聚合；以及如何有效展示不确定性，可能需要在视觉编码上进行权衡，例如透明度、误差线等。

最后，需要确保应用符合用户要求：不使用 Observable Notebook，但允许使用 Observable Plot 和 Framework。可能需要使用 Observable Framework 来构建仪表盘，将多个视图整合，并处理数据联动。

总结，开发步骤可能包括：

- 数据处理和聚合，生成每个社区的指标。
- 使用 Observable Plot 创建地图、折线图、柱状图。
- 添加交互功能，如联动、工具提示。
- 部署为独立的 Web 应用，可能用 Observable Framework 生成静态页面。

```
// 导入库
import { Plot, dot, line, rect } from
'@observablehq/plot'; import { legend, color } from
'@observablehq/plot';

// 加载数据（示例结构）
const shakeMap = await
d3.json('shake-data.geojson'); const reports = await
d3.csv('damage-reports.csv'); const neighborhoods =
await d3.csv('neighborhoods.csv');

// 1. 地图视图：优先级
const map = Plot.plot({
projection: 'identity',
color: { legend: true },
marks: [

  Plot.geo(shakeMap, { fill: (d) =>
  d.properties.intensity, title: (d) =>
  d.properties.neighborhood, }),
  Plot.dot(reports, { x: 'longitude', y:
  'latitude', fill: 'severity',
```

```
      r: 3,
      tip: true,
    }),
  ],
});

// 2. Reliability Scatter Plot
const reliability = Plot.plot({
  marks: [
    Plot.dot(neighborhoods, {
      x: "report_count",
      y: "report_variance",
      fill: "reliability_score",
      title: (d) => d.neighborhood,
    }),
    Plot.text(neighborhoods, {
      filter: (d) => d.reliability_score > 0.8,
      text: "neighborhood",
      dx: 8,
    }),
  ],
});

// 3. Time Series with Uncertainty
const timeSeries = Plot.plot({
  marks: [
    Plot.line(reports, {
      x: "date",
      y: "severity",
      stroke: "neighborhood",
    }),
    Plot.area(reports, {
      x: "date",
      y1: "severity_lower",
      y2: "severity_upper",
      fill: "neighborhood",
      fillOpacity: 0.2,
    }),
  ],
});
```

```
r: 3,
提示:
true, }),
], });



// 2. 可靠性散点图 const
reliability = Plot.plot({

  标记: [
Plot.dot(neighborhoods, { x:
"report_count", y:
"report_variance", fill:
"reliability_score", title: (d) =>
d.neighborhood, }),
Plot.text(neighborhoods, { filter:
(d) => d.reliability_score > 0.8,
text: "neighborhood", dx: 8, }), ],
});



// 3. 带不确定性的时间序列
const timeSeries = Plot.plot({

  标记: [
Plot.line(reports, { x:
"date", y: "severity",
stroke:
"neighborhood", }),
Plot.area(reports, { x:
"date", y1:
"severity_lower", y2:
"severity_upper", fill:
"neighborhood",
fillOpacity: 0.2, }), ], });
```