AI Society

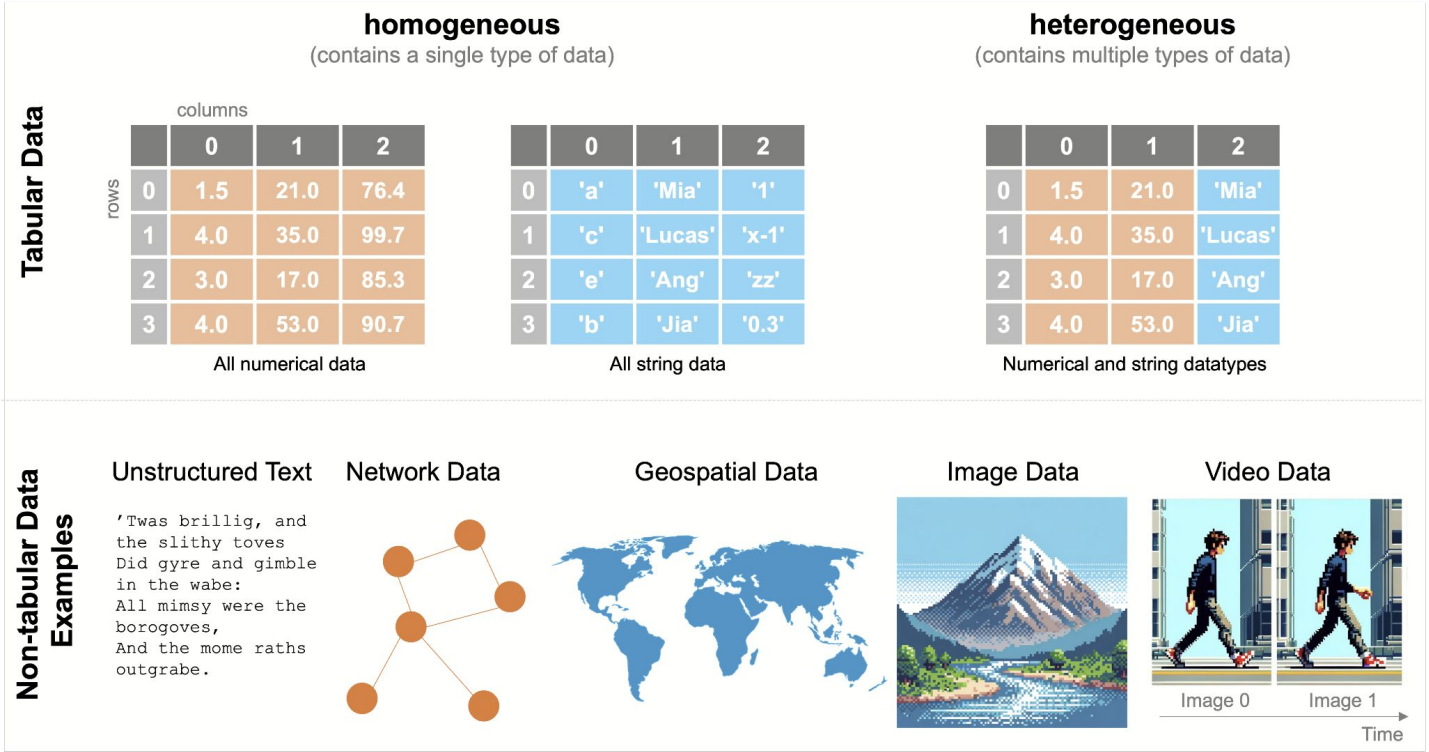
Week 3: AI Student

[February 6, 2024]

Last week's task

How is data stored?

Examples



Credit: [Practical Data Science]

From data to insights

- We have the data but what do we do with it? Start at the right end! DO NOT start by looking in the data to find a problem - you will find things in the data anyways ;)

- Example: A business holds stock for stores. They get new deliveries of product A each week.

  Should we keep more inventory of product A?

How do we answer questions with data?

- For a question like - Should we keep more inventory of product A?

We need to

1. Go through all sales weeks for product A
2. Find out remaining stock in each week
3. Make a judgement on whether we consistently have too much inventory

<code goes here>

How do we answer questions with data?

- For a question like - Should we keep more inventory of product A?

```
SELECT
   *
FROM sales
WHERE
   product = ‘product_a’
```

week	remaining_inventory	product
2025-01-13	10	product_a
2025-01-06	18	product_a
2024-12-30	0	product_a
2024-12-23	4	product_a

How do we find what we are looking for?

- This is such a common task that people have written tools to help us write less code!

- Enter query processing engines and their interfaces:

Structured Query Language (SQL), Polars, Pandas, Dplyr (R)

Basic SQL/Pandas

```sql
-- Select all employees from table
SELECT * FROM employees;
-- Select specific columns
SELECT name, salary FROM employees;
-- Select all employees with a salary greater than
50000
SELECT * FROM employees WHERE salary > 50000;
-- Select all employees with a name that is not null
SELECT * FROM employees WHERE name IS NOT NULL;
-- Select all employees ordered by salary in
descending order
SELECT * FROM employees ORDER BY salary DESC;
```

```python
import pandas as pd
df = pd.DataFrame({
    'name': ['Alice', 'Bob', 'Charlie'],
    'salary': [50000, 60000, 70000]
})

# Select all employees
df

# Select specific columns
df[['name', 'salary']]

# Select rows where salary is greater than
50000
df[df['salary'] > 50000]

# Select rows where name is not null
df[df['name'].notna()]

# Select all employees ordered by salary in
descending order
df.sort_values(by='salary', ascending=False)
```

Aggregations and joins

```python
# Count the number of employees
df.shape[0]

# Select the average salary by department
df.groupby('department')['salary'].mean()

# Select all employees and their departments
df.merge(departments, on='department_id')
```

```sql
-- Count the number of employees
SELECT COUNT(*) FROM employees;
-- Select the average salary by department
SELECT department, AVG(salary) FROM employees GROUP BY
department;
-- Select all employees and their departments
SELECT * FROM employees e
JOIN departments d ON e.department_id = d.id;
```

Practicalities – Cleaning data

Imagine this table is from an Excel sheet

week	remaining_inventory	product
2025-01-13	3	null
2025/01/13	10	product_a
2025-01-06	18	product_a
2024-12-30	0	product_a
2024-12-23	400	product_a

Is this product_a? Might need to talk to someone who knows

Wrong format

This date is a Sunday and not a monday like the rest

We only bought 40 units of product_a for this week, it cannot be 400

Practicalities – Cleaning data

Cleaned table

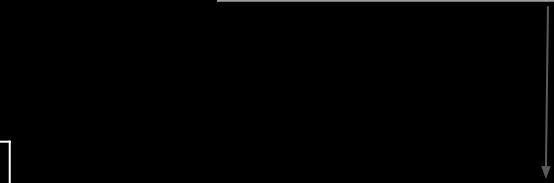
week	remaining_inventory	product
2025-01-13	13	product_a
2025-01-06	18	product_a
2024-12-30	0	product_a
2024-12-23	40	product_a

Practicalities – Merging data sources

We now need revenue based on sales figures. We have the sales in the sales table but the price of the product sold lives in another table.

date	sales	product_id
2025-01-06	8	product_a
2025-01-05	3	product_a
2025-01-04	1	product_b
	…	

product_id	price
product_a	10
product_b	25



product_id	total_revenue
product_a	110
product_b	25

Practicalities – Merging data sources

```
SELECT
    s.product_id,
    (SUM(s.sales) * p.price) AS total_revenue
FROM sales AS s
INNER JOIN pricing AS p ON s.product_id = p.product_id
GROUP BY s.product_id, p.price;
```

product_id	total_revenue
product_a	110
product_b	25

Remarks about this weeks tasks

- Quite extensive homework, use LLMs to help if you are stuck. Recommendation, ask:

  “Why does my approach not work?”

  “Could you give an example of a similar problem this the one I am solving and how to solve it?”

- Second part of homework is hard, it’s from a real job interview