Phân tích dữ liệu cơ bản

Tài liệu này giải thích cách tính các đại lượng thống kê quan trọng: **Giá trị trung bình (Mean)**, **Độ lệch chuẩn (Standard Deviation - std)**, **Ước lượng mật độ hạt nhân (Kernel Density Estimation - KDE)**, **Hệ số tương quan Pearson**, và **Hệ số tương quan Spearman**. Mỗi phần bao gồm công thức, lý do sử dụng và ví dụ mã Python.

---

1. Giá trị trung bình (Mean)

Định nghĩa

Giá trị trung bình là giá trị trung bình cộng của một tập dữ liệu. Đây là một thước đo trung tâm, tóm tắt tập dữ liệu bằng một giá trị đại diện.

Công thức

$$\text{Mean}(\mu) = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Trong đó:

- (n): Số lượng dữ liệu.
- (x_i): Từng giá trị dữ liệu.

Tại sao cần tính?

Giá trị trung bình cung cấp cái nhìn tổng quan nhanh về giá trị trung tâm của tập dữ liệu, cho phép so sánh giữa các tập dữ liệu và giúp xác định phân phối dữ liệu.

Ví dụ Python

```
import numpy as np

# Tập dữ liệu ví dụ
data = [10, 15, 20, 25, 30]

# Tính giá trị trung bình
mean = np.mean(data)
print(f"Giá trị trung bình: {mean}")
```

---

2. Độ lệch chuẩn (Standard Deviation - std)

Định nghĩa

Độ lệch chuẩn đo lường sự phân tán hoặc biến động của tập dữ liệu xung quanh giá trị trung bình.

Công thức

$$\text{Standard Deviation}(\sigma) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

Trong đó:

- (μ): Giá trị trung bình của tập dữ liệu.
- (x_i): Từng giá trị dữ liệu.
- (n): Số lượng dữ liệu.

Tại sao cần tính?

Độ lệch chuẩn giúp đo lường mức độ phân tán của dữ liệu, cho biết liệu các giá trị có gần hay xa so với trung bình. Đây là yếu tố quan trọng để phát hiện giá trị ngoại lai và hiểu sự biến đổi dữ liệu.

Ví dụ Python

```
# Tính độ lệch chuẩn
std = np.std(data)
print(f"Độ lệch chuẩn: {std}")
```

---

3. Ước lượng mật độ hạt nhân (KDE)

Định nghĩa

Ước lượng mật độ hạt nhân (KDE) là phương pháp phi tham số để ước tính hàm mật độ xác suất (PDF) của một tập dữ liệu. Nó làm mịn phân phối dữ liệu để tạo ra biểu diễn liên tục.

Công thức

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

Trong đó:

- (K): Hàm kernel (ví dụ: Gaussian).
- (h): Băng thông (kiểm soát độ mịn).

- (x_i): Dữ liệu.
- (n): Số lượng dữ liệu.

Tại sao cần tính?

KDE hữu ích để trực quan hóa phân phối dữ liệu theo cách mượt mà và liên tục, khắc phục nhược điểm của biểu đồ histogram (phụ thuộc vào cách chia bin).

Ví dụ Python

```
import seaborn as sns
import matplotlib.pyplot as plt

# Vẽ KDE
sns.kdeplot(data, fill=True)
plt.title("Ước lượng mật độ hạt nhân")
plt.show()
```

---

4. Hệ số tương quan Pearson

Định nghĩa

Hệ số tương quan Pearson đo lường mối quan hệ tuyến tính giữa hai biến, giá trị nằm trong khoảng từ (-1) (tương quan nghịch hoàn hảo) đến (+1) (tương quan thuận hoàn hảo).

Công thức

$$\text{Pearson}(X, Y) = \frac{\sum_{i=1}^{n}(x_i - \mu_X)(y_i - \mu_Y)}{\sqrt{\sum_{i=1}^{n}(x_i - \mu_X)^2 \sum_{i=1}^{n}(y_i - \mu_Y)^2}}$$

Trong đó:

- (x_i, y_i): Giá trị dữ liệu của biến (X) và (Y).
- (μ_X, μ_Y): Giá trị trung bình của (X) và (Y).

Tại sao cần tính?

Hệ số tương quan Pearson giúp xác định mối quan hệ tuyến tính giữa các biến, điều này rất quan trọng trong hồi quy và mô hình dự đoán.

Ví dụ Python

```
from scipy.stats import pearsonr

# Dữ liệu ví dụ
```

```
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]

# Tính hệ số tương quan Pearson
corr, _ = pearsonr(x, y)
print(f"Hệ số tương quan Pearson: {corr}")
```

---

5. Hệ số tương quan Spearman

Định nghĩa

Hệ số tương quan Spearman đo lường mối quan hệ đơn điệu (không nhất thiết tuyến tính) giữa hai biến. Nó sử dụng thứ hạng thay vì dữ liệu thô.

Công thức

$$\text{Spearman}(X, Y) = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n(n^2 - 1)}$$

Trong đó:

- (d_i): Hiệu số thứ hạng giữa (x_i) và (y_i).
- (n): Số lượng dữ liệu.

Tại sao cần tính?

Hệ số tương quan Spearman phù hợp cho các mối quan hệ phi tuyến và các tập dữ liệu có ngoại lệ hoặc dữ liệu thứ tự.

Ví dụ Python

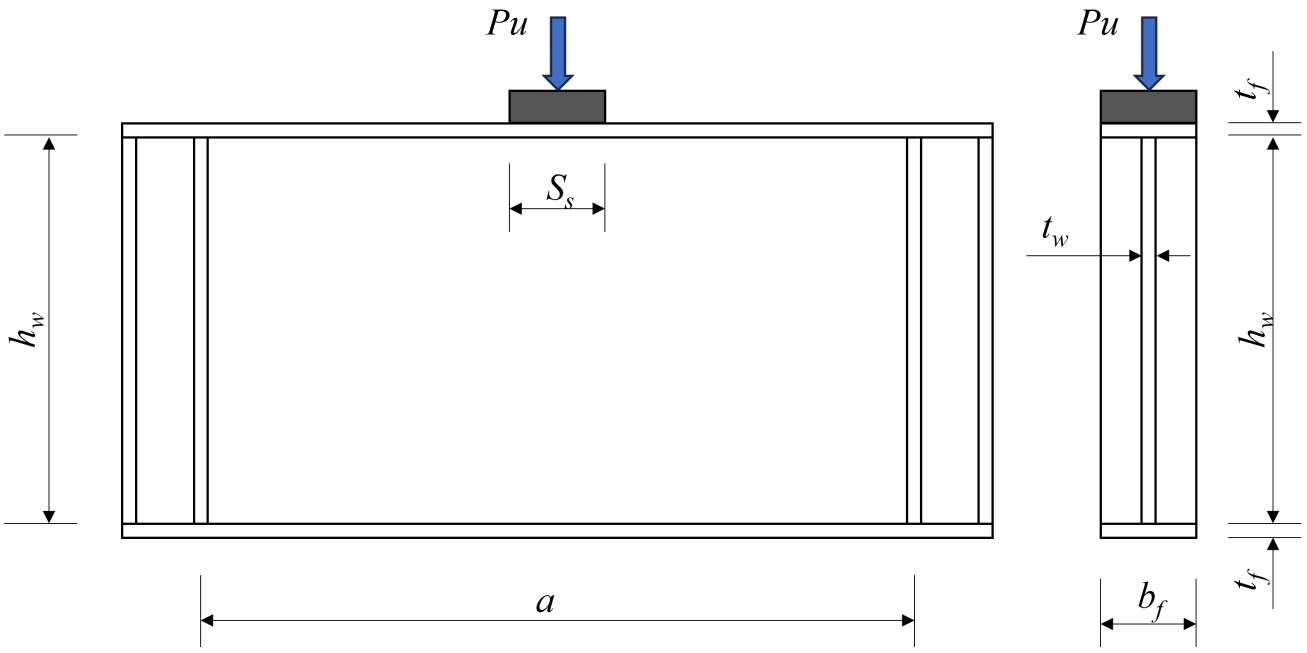
```
from scipy.stats import spearmanr

# Tính hệ số tương quan Spearman
corr, _ = spearmanr(x, y)
print(f"Hệ số tương quan Spearman: {corr}")
```

---

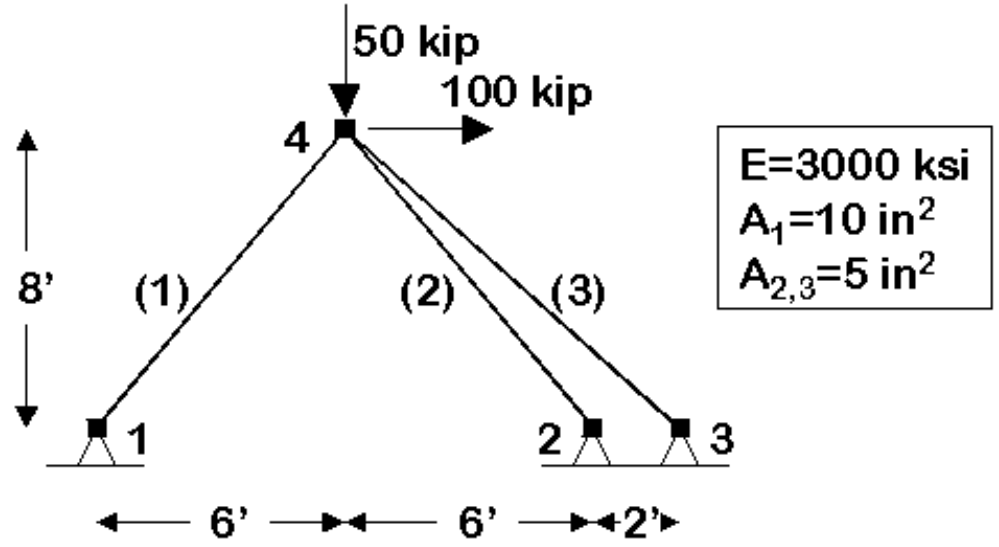
6. Homework

6.1. Phân tích bộ dữ liệu dầm thép chịu tải trọng patch loading.

Dữ liệu lấy trong folder Data. Phân tích bộ dữ liệu và đề xuất các tính chất đang chú ý.



6.2. Tối ưu giàn.



Phân tích giàn trên với code python sau:

```
def analysis_truss(b1, h1, b2, h2, b3, h3):
    s1 = b1*h1
    s2 = b2*h2
    s3 = b3*h3
    # ----------------------------
    # Start of model generation
    # ----------------------------

    # remove existing model
    wipe()

    # set modelbuilder
    model('basic', '-ndm', 2, '-ndf', 2)
```

```
# create nodes
node(1, 0.0, 0.0)
node(2, 144.0,  0.0)
node(3, 168.0,  0.0)
node(4,  72.0, 96.0)

# set boundary condition
fix(1, 1, 1)
fix(2, 1, 1)
fix(3, 1, 1)

# define materials
uniaxialMaterial("Elastic", 1, 3000.0)

# define elements
element("Truss",1,1,4,s1,1)
element("Truss",2,2,4,s2,1)
element("Truss",3,3,4,s3,1)

# create TimeSeries
timeSeries("Linear", 1)

# create a plain load pattern
pattern("Plain", 1, 1)

# Create the nodal load - command: load nodeID xForce yForce
load(4, 100.0, -50.0)

# ----------------------------
# Start of analysis generation
# ----------------------------

# create SOE
system("BandSPD")

# create DOF number
numberer("RCM")

# create constraint handler
constraints("Plain")

# create integrator
integrator("LoadControl", 1.0)

# create algorithm
algorithm("Linear")

# create analysis object
analysis("Static")
```

```
    # perform the analysis
    analyze(1)
    force_1 = abs(basicForce(1)[0])  # Element 1
    force_2 = abs(basicForce(2)[0])  # Element 1
    force_3 = abs(basicForce(3)[0])  # Element 1
    weight1 = 490*(s1/144)*10
    weight2 = 490*(s2/144)*10
    weight3 = 490*(s1/144)*11.3
    weight_all = round(weight1+weight2+weight3,2)
    return round(force_1/s1, 2), round(force_2/s2, 2), round(force_3/s3, 2), weight_all)
```

Trong đó b và h lần lượt là kích thước các thanh. Kết quả hàm trả về ứng suất trên từng thanh và tổng khối lượng của cả giàn.

Ví dụ:

```
analysis_truss(1.1, 1.1 ,5,5,5,5)
```

có kết quả là (30.3, 0.27, 4.19, 938.39)

Yêu cầu: Xác định kích thước của từng thanh để không có thanh nào trong giàn có ứng suất vượt quá 36 và khối lượng của giàn là nhỏ nhất có thể. Đề xuất phương pháp tính để áp dụng cho các hệ giàn bất kỳ.