**SAMPLING METHODS**

**NHÓM 09 – DHKHMT18B**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **Họ** | **Tên** | **Công việc** | **Ghi chú** |
| 41 | 21046661 | Nguyễn Thế | Lực |  |  |
| 42 | 20113851 | Tào Văn | Mạnh |  |  |
| 43 | 21078341 | Phan Thanh | Nam |  |  |
| 44 | 21113621 | Vũ Hải | Nam |  |  |
| 45 | 21130791 | Nguyễn Thị | Nga |  |  |

# **FOUR RANDOM SAMPLING METHODS**

## ***Simple random sampling:*** là phương pháp lấy mẫu trong phân tích dữ liệu, trong đó các phần tử được chọn từ một tập hợp lớn dựa trên một kỳ quy hoạch (khoảng cách) cố định.

## ***Systematic sampling :*** là một phương pháp lấy mẫu trong nghiên cứu thống kê, trong đó mỗi phần tử của tập dữ liệu có cơ hội bằng nhau để được chọn làm mẫu, mà không quan trọng đến bất kỳ đặc điểm nào khác.

## ***Stratified and weighted random sampling***

* Stratified Sampling : Là phương pháp lấy mẫu trong đó chia tập dữ liệu thành các phân khúc (strata) dựa trên một biến quan trọng (như nhãn), sau đó lấy mẫu ngẫu nhiên từ mỗi phân khúc theo tỷ lệ xác định
* Weighted Random Sampling : Là phương pháp lấy mẫu trong đó mỗi mẫu được chọn với xác suất tỷ lệ với trọng số liên kết với nó, nghĩa là mẫu có trọng số lớn sẽ có xác suất lớn hơn được chọn.

## ***Cluster sampling:***là kỹ thuật chọn mẫu trong đó việc lựa chọn ngẫu nhiên các nhóm cá thể (được gọi là chùm) từ nhiều chùm trong 1 quần thể nghiên cứu.

1. **RESULTS AND DISCUSS THE RESULTS**

## ***Simple random sampling***

A close-up of a computer screen

Description automatically generated

**Thảo luận**

Mẫu không đại diện cho dân số, vì các chỉ số mean, median, std, min, max có độ lệch khá đáng kể so với populatio

* Cần sử dụng các phương pháp thống kê để đánh giá độ chính xác của kết quả thu được từ mẫu.

## ***Systematic sampling***

A screenshot of a computer

Description automatically generated

**Thảo luận**

Phương pháp lấy mẫu systematic cho ra mẫu có thông số phản ánh tốt hơn so với simple sampling, yêu cầu phải trộn dữ liệu trước.

Simple sampling không cần sắp xếp, nhưng dữ liệu không phân bố đều, không phản ánh sự phân phối của population.

## ***Stratified and weighted random sampling***

A screenshot of a computer

Description automatically generated

A screenshot of a computer code

Description automatically generated

**Thảo luận** : Cho ra sample có tỷ lệ gần hơn với tỷ lệ của population

## ***Cluster sampling***

A screenshot of a computer

Description automatically generated

**Thảo luận :**

Thực hiện có thể nhanh hơn so với SRS, vì không cần liệt kê tất cả các cá thể trong tổng thể

**KẾT LUẬN**

* Mẫu ngẫu nhiên đơn giản (SRS): đơn giản, đại diện cho tổng thể, nhưng không hiệu quả khi tổng thể lớn.
* Mẫu phân tầng (Stratified): đảm bảo đại diện cho các phân tầng, giảm sai số, nhưng yêu cầu biết trước phân tầng.
* Mẫu có trọng số (Weighted): linh hoạt, đảm bảo đại diện cho các nhóm mong muốn, nhưng yêu cầu ước tính tỷ lệ mong muốn và gán trọng số.
* Mẫu theo cụm (Cluster): tiết kiệm chi phí, thực tế, nhưng có thể dẫn đến sai số cao và giảm tính đại diện.

# **NOTE**

## **EXPLORATORY DATA ANALYSIS – EDA**

### **Steps in Exploratory Data Analysis (EDA)**

#### **General characteristics of the dataset (Đặc điểm chung của tập dữ liệu) :** print(df.info())

#### **Descriptive statistics (univariate) (Thống kê mô tả) :**

* Numerical variables : print(df.describe())
* Categorical variables
* Cardinality: number of unique values : df['column\_name'].nunique()
* Unique counts: number of occurrences of each unique value : df['column\_name'].value\_counts()

#### **Correlation statistics (bivariate) (Thống kê tương quan)**

* Qualitative analysis
* Quantitative analysis

#### **Exploratory visualisation - univariate and bivariate (đơn biến và hai biến)**

* **Histogram:** plt.hist(data, bins=5, edgecolor='black')
* **Box plot:** plt.boxplot([data1, data2], labels=['Group 1', 'Group 2'], notch=True)
* **Bar plot:** plt.scatter(x, y, color='blue', marker='o', edgecolors='black')
* **Heat map:** sns.heatmap(df, annot=True)

#### **Missing values**

**# Dữ liệu mẫu với giá trị thiếu**

data = {'col1': [1, 2, None, 4], 'col2': ['A', None, 'C', 'D'], 'col3': [5, None, 7, 8]}

df = pd.DataFrame(data)

**# Kiểm tra giá trị thiếu**

print(df.isnull().sum()) # Hiển thị số lượng giá trị thiếu cho mỗi cột

**# 1. Xóa (không khuyến khích cho tập dữ liệu lớn)**

df\_dropped = df.dropna() # Xóa các hàng có bất kỳ giá trị nào thiếu

**# 2. Thay thế bằng Trung bình/Trung vị/Mốt**

**# (chọn phương pháp phù hợp dựa trên kiểu dữ liệu - trung bình cho số, mốt cho danh mục)**

imputer = SimpleImputer(strategy='mean') # Thay thế bằng 'median' hoặc 'most\_frequent' cho mốt

df\_imputed\_mean = imputer.fit\_transform(df)

**# 3. Thay thế bằng KNN**

imputer = KNNImputer(n\_neighbors=5) # Sử dụng k láng giềng gần nhất để dự đoán giá trị thiếu

df\_imputed\_knn = imputer.fit\_transform(df)

**# In kết quả (thay thế bằng tên cột của bạn)**

print(df[['col1', 'col2']])

print(df\_dropped[['col1', 'col2']])

print(pd.DataFrame(df\_imputed\_mean, columns=df.columns))

print(pd.DataFrame(df\_imputed\_knn, columns=df.columns))

#### **Other**

* **Find the most common gender and birth country**  
  gender\_mode = laureates\_pop["gender"].mode()[0] birth\_country\_mode = laureates\_pop["bornCountry"].mode()[0]
* **Convert datatype :**

laureates\_pop['gender'] = laureates\_pop['gender'].astype('category')

laureates\_pop['bornCity'] = laureates\_pop['bornCity'].astype('category')

* **Identify the decade with the highest proportion of US-born winner**

**# 1. Filter US winners**

us\_winners = laureates\_pop[laureates\_pop["bornCountry"] == "USA"]

**# 2. Add a "Decade" column**

us\_winners['Decade'] = (us\_winners['year'] // 10) \* 10

us\_winners

**# 3. Caculate proportion of US winners per decade**

decade\_proportions = (us\_winners.groupby('Decade')['year'].count()/len(us\_winners)) \* 100

**# 4. The highest proportion of US-born winner**

highest\_decade = decade\_proportions.idxmax()

highest\_proportion = decade\_proportions.max()

* **Create a dictionary with results**

max\_female\_dict = {'Year': highest\_decade[0], 'Category': highest\_decade[1], 'Proportion': highest\_proportion}

* **Find first woman to win a Nobel Prize**

first\_woman\_win = female\_winner.sort\_values(by='year').iloc[0]