Open in Colab

SEMMA: Intro to Sample & Explore

In [ ]:
```
# Import the libraries we'll need
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# Suppress warnings
import warnings
warnings.filterwarnings('ignore')
```

PART 1: LOADING AND VIEWING THE DATA

In [ ]:
```
# Load the dataset
df = pd.read_csv('https://raw.githubusercontent.com/iamctodd/datasets/refs/heads/ma
```

In [ ]:
```
# Look at the first 5 rows of data
```

In [ ]:
```
# Get information about each column
```

In [ ]:
```
# See summary statistics
```

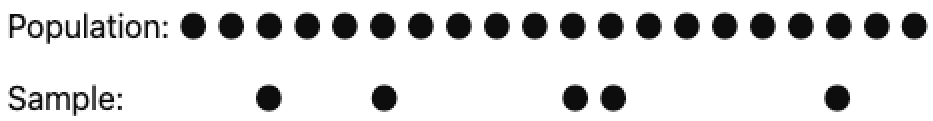
In [ ]:
```
# Count the values in the 'Membership Type' column
```

In [ ]:
```
# Count the values in the 'Genderl' column
```

PART 2: SAMPLING TECHNIQUES

1. Simple Random Sampling

A random selection of rows



In [ ]:
```
# Take a completely random selection of rows
sample_size = 100
random_sample = # INSERT CODE
```

In [ ]:

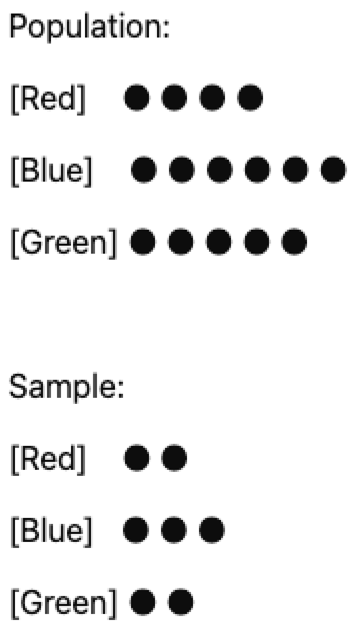
```python
# Save the random sample to a CSV file
```

In [ ]:

```python
# Let's look at Membership types in Random sample
```

2. Stratified Sampling

Sample while maintaining the same proportion of a particular column



In [ ]:

```python
# Look at 'Membership Type'
# What proportion of each membership type do we have?
```

In [ ]:

```python
# Write a function to perform stratified sampling
def stratified_sample(data, column, size):
    """
    Takes a sample that preserves the proportions of values in a column

    Inputs:
    - data: the DataFrame to sample from
    - column: the column to stratify by
    - size: the total sample size

    Returns:
    - A DataFrame containing the stratified sample
    """

    # Get the value counts and calculate proportions
    value_counts = data[column].value_counts(normalize=True)

    # Create an empty DataFrame to store our samples
    result = pd.DataFrame()
    # For each value in our column
    for value, proportion in value_counts.items():
        # Calculate how many samples to take
        n_to_sample = round(proportion * size)

        # Get all rows with this value
        value_data = data[data[column] == value]

        # Sample the right number of rows
        if n_to_sample > 0:
            sampled_rows = value_data.sample(n=min(n_to_sample, len(value_data)), r
            result = pd.concat([result, sampled_rows])

    return result
```

In [ ]:

```
# Take a stratified sample of df and look at the first few rows
```

In [ ]:

```
# Let's check the Membership counts
```

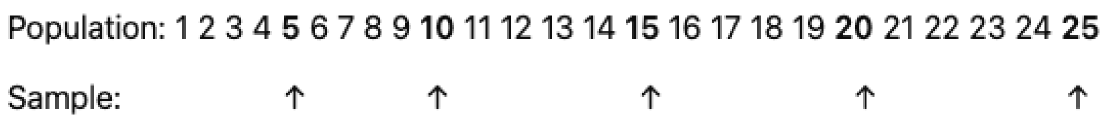
In [ ]:

```
# Save the stratified sample to a CSV file
```

3. Systematic Sampling

Take every Nth row from the data

Example of every 5th row


Population: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
Sample: ↑ ↑ ↑ ↑ ↑

In [ ]:

```
# Calculate the step size
```

In [ ]:

```
# Take every Nth row
```

In [ ]:

```
# Let's look at Membership types in Systematic sample
```

In [ ]:

```
# Save the systematic sample to a CSV file
```

PART 3: EXERCISES

Exercise 1: Basic Exploration

1. Upload the file called Week 2 S1 Sampling Practice Data

2. Create a sample of 300 cases:

    - A simple random sample
    - A stratified sample based on Regions
    - Systematic sample
3. Extract samples as CSV file

4. What is the regional counts or proportion for each sample and compare with original sample:

5. Give a recommendation on the best sampling method for the dataset.

Create Samples and Analyze

In [ ]:
```
# Upload the file called Week 2 S1 Sampling Practice Data
```

In [ ]:
```
# Check info (.info())
```

In [ ]:
```
# Check Column Headers (.head())
```

In [ ]:
```
# Check Regions proportions Overall
```

Simple Random Sample

In [ ]:
```
# Take a complete random selection of rows
```

In [ ]:
```
# Save the random sample as a CSV file
```

In [ ]:
```
# Check the Regions proportions
```

Stratified sample

In [ ]:
```
# Write a function to perform stratified sampling (Ask Gemini)
```

In [ ]:
```
# Reuse the stratified sample function defined earlier
def stratified_sample(data, column, size):
    """
    Takes a sample that preserves the proportions of values in a column

    Inputs:
    - data: the DataFrame to sample from
    - column: the column to stratify by
    - size: the total sample size

    Returns:
    - A DataFrame containing the stratified sample
    """

    # Get the value counts and calculate proportions
    value_counts = data[column].value_counts(normalize=True)

    # Create an empty DataFrame to store our samples
    result = pd.DataFrame()

    # For each value in our column
    for value, proportion in value_counts.items():
        # Calculate how many samples to take
        n_to_sample = round(proportion * size)

        # Get all rows with this value
        value_data = data[data[column] == value]

        # Sample the right number of rows
        if n_to_sample > 0:
            sampled_rows = value_data.sample(n=min(n_to_sample, len(value_data)), r
            result = pd.concat([result, sampled_rows])

    return result

# Take a stratified sample based on 'Regions'
```

```
In [ ]: # Save the stratified sample as a CSV file
```

```
In [ ]: # Check the Regions proportions
```

Systematic Sampling

```
In [ ]: # Calculate the step size
```

```
In [ ]: # Take every Nth row
```

```
In [ ]: # Save the systematic sample to a CSV file
```

```
In [ ]: # Check the Regions proportions for the systematic sample
```