250840125020_Assignment1

November 27, 2025

[1]:

```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv('american_income.csv')

# Parameters
N = len(df)
n_target = int(0.1 * N)  # 10% of 25,000 = 2,500

print(f"Total Population: {N}")
print(f"Target Sample Size: {n_target}")
```

```
Total Population: 25000
Target Sample Size: 2500
```

1. Simple Random Sampling

[2]:

```
sample_simple = df.sample(n=n_target, random_state=42)
```

2. Systematic Sampling

[3]:

```
k = int(N / n_target)  # Interval = 10
start = random.Random(42).randint(0, k - 1)
indices = np.arange(start, N, k)
# Ensure exactly n_target size (though with k=10 it fits perfectly)
indices = indices[:n_target]
sample_systematic = df.iloc[indices]
```

3. Stratified Sampling (Strata = race)

[4]:

```
# Calculate proportional sample size for each stratum
strata_counts = df['race'].value_counts()
stratified_parts = []
for race, count in strata_counts.items():
    # Calculate portion (rounding to nearest integer)
    n_stratum = int(round(count * 0.1))
    stratified_parts.append(df[df['race'] == race].sample(n=n_stratum, random_state=42))

sample_stratified = pd.concat(stratified_parts)

# Adjust if total count is slightly off due to rounding (though here it sums exactly to 2500)
if len(sample_stratified) != n_target:
    sample_stratified = sample_stratified.sample(n=n_target, random_state=42)
```

4. Cluster Sampling (Clusters = marital.status)

[5]:
```
clusters = df['marital.status'].unique()
# Shuffle clusters to ensure random selection
random.Random(42).shuffle(clusters)

sample_cluster_parts = []
current_count = 0
for cluster in clusters:
    cluster_data = df[df['marital.status'] == cluster]
    sample_cluster_parts.append(cluster_data)
    current_count += len(cluster_data)
    if current_count >= n_target:
        break

sample_cluster = pd.concat(sample_cluster_parts)

# "Remove extra records" to match exactly 10%
if len(sample_cluster) > n_target:
    sample_cluster = sample_cluster.sample(n=n_target, random_state=42)
```

Analysis and Comparison

[6]:
```
def calculate_stats(data, method_name):
    return {
        "Method": method_name,
        "Avg Education Num": data['education.num'].mean(),
        "Avg Fnlwgt": data['fnlwgt'].mean()
    }

stats = [
    calculate_stats(df, "Overall Population"),
    calculate_stats(sample_simple, "Simple Random"),
    calculate_stats(sample_systematic, "Systematic"),
    calculate_stats(sample_stratified, "Stratified"),
    calculate_stats(sample_cluster, "Cluster")
]
```

```
results = pd.DataFrame(stats)
```

Calculate absolute difference from population mean

[7]:

```
pop_edu = results.loc[0, "Avg Education Num"]
pop_wgt = results.loc[0, "Avg Fnlwgt"]

results['Diff Education'] = abs(results['Avg Education Num'] - pop_edu)
results['Diff Fnlwgt'] = abs(results['Avg Fnlwgt'] - pop_wgt)

print(results[['Method', 'Avg Education Num', 'Avg Fnlwgt', 'Diff Education',␣
 ↪'Diff Fnlwgt']])
```

```
               Method  Avg Education Num    Avg Fnlwgt  Diff Education  \
0  Overall Population           10.07632  189661.13492         0.00000
1       Simple Random           10.09200  192439.01880         0.01568
2          Systematic           10.08360  193176.18600         0.00728
3          Stratified           10.09040  187852.54880         0.01408
4             Cluster           10.04840  188078.92880         0.02792

   Diff Fnlwgt
0      0.00000
1   2777.88388
2   3515.05108
3   1808.58612
4   1582.20612
```

[ ]:

```

```