Statistics Basics| Assignment

Question 1: What is the difference between descriptive statistics and inferential statistics? Explain with examples.

Answer:

Descriptive Statistics

Definition:
Descriptive statistics summarize and describe the properties of a dataset you already have—without attempting to generalize beyond it. It doesn't involve probability or sampling error.

Common Measures:

- Central tendency: mean, median, mode
- Variability: range, variance, standard deviation, skewness, kurtosis
- Visuals: histograms, frequency tables, box plots
- **Example:**
  A classroom teacher records the test scores of all 30 students and calculates the average (e.g., 78), median, standard deviation, and produces a distribution graph. These summarize the class data, but don't extend beyond it

Inferential Statistics

Definition:
Inferential statistics uses a sample of data to draw conclusions or make predictions about a broader population. It incorporates probability theory to account for sampling error.

Key Methods:

- Hypothesis testing (t-tests, chi-square, ANOVA)
- Confidence intervals (e.g., "95% CI for mean")
- Regression or correlation analysis
- **Example:**
  Suppose the teacher selects a random sample of 100 students across multiple schools and estimates the average score with a 95 % confidence interval (e.g.,

between 75–82). They might also perform a hypothesis test to see if a new teaching method significantly affects scores.

Question 2: What is sampling in statistics? Explain the differences between random and stratified sampling.

Answer:

Sampling is the process of selecting a subset of individuals (a *sample*) from a larger population to estimate population parameters more efficiently. Since surveying an entire population is often impractical or costly, a well-chosen sample lets researchers draw valid inferences about the whole group. Sampling allows for faster, cheaper data collection with measurable uncertainty.

Simple (Random) Sampling

Definition:
In simple random sampling (also called probability sampling), every member of the population has an equal and independent chance of being selected. Every possible sample of a given size is equally likely.

Stratified Sampling

Definition:
Stratified sampling divides the population into distinct, non-overlapping subgroups (strata) based on one or more characteristics (e.g., age, gender, location). Then a random sample is drawn from each stratum, often in proportion to its size in the full population.

Feature	Simple (Random) Sampling	Stratified Sampling
Selection mechanism	Random across whole population	Random within defined strata
Chance of selection	Equal for all individuals	May differ by stratum (often proportional)
Subgroup representation	Uncontrolled; may miss small groups	Controlled; each stratum represented

Feature	Simple (Random) Sampling	Stratified Sampling
Implementation complexity	**Simple**	**More planning and data needed to define strata**
Precision of estimates	**Good if population is uniform**	**Typically better, especially for subgroup and overall estimates**
Best used when	**Homogeneous population, no subgroup focus**	**Heterogeneous population, subgroup analysis required**

Question 3: Define mean, median, and mode. Explain why these measures of central tendency are important.

Answer:

Definitions

Mean:

Also known as the **arithmetic average**, the **mean** is calculated by summing all values in a dataset and dividing by the count of observations:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

It uses every data point, which makes it informative—but also sensitive to outliers.

Median:

The **median** is the middle value in an ordered list of observations: half the data lies below it and half above. If the dataset has an even number of values, it's the average of the two middle numbers.

Mode:

The **mode** is the most frequently occurring value in a dataset. Datasets may be unimodal (one), multimodal (multiple), or have no mode at all. Unlike mean and median, mode can be used for nominal (categorical) data.

These measures help summarize and communicate large datasets with a single representative value:

- **Mean** reflects the overall average and is useful when data is symmetric and without extreme values.
- **Median** gives the middle point and is **robust to outliers**, making it preferable in skewed distributions.
- **Mode** highlights the most common observation and is the only one applicable for **categorical data**.

Question 4: Explain skewness and kurtosis. What does a positive skew imply about the data?

Answer:

Skewness

Definition:
Skewness measures the **asymmetry** of a distribution around its mean. A perfectly symmetric distribution (like normal) has zero skewness. Positive or negative skewness indicates which tail of the distribution is longer or heavier.

- **Positive skew** (also called right-skewed): the **right tail** is longer—meaning there are a few notably high values pulling the distribution in that direction.
- **Negative skew** (left-skewed): the **left tail** is longer, meaning more extreme low values.

Kurtosis

Definition:
Kurtosis describes the **"tailedness"** of a distribution—i.e., how heavy or light the tails are in comparison to a normal distribution. Contrary to a common misconception, it's not about "peakedness," but about **extreme values (outliers)** in the tails.

Types of kurtosis relative to the normal distribution:

- **Mesokurtic** ($k \approx 3$ or excess kurtosis 0): similar tail behavior to a normal curve.
- **Leptokurtic** ($k > 3$): heavy tails—more frequent extreme values/outliers.
- **Platykurtic** ($k < 3$): light tails—fewer extreme values.

Question 5: Implement a Python program to compute the mean, median, and mode of a given list of numbers. numbers = [12, 15, 12, 18, 19, 12, 20, 22, 19, 19, 24, 24, 24, 26, 28]

Answer:

```
from collections import Counter


numbers = [12, 15, 12, 18, 19, 12, 20, 22, 19, 19, 24, 24, 24, 26, 28]


def mean(nums):
    return sum(nums) / len(nums)


def median(nums):
    sorted_nums = sorted(nums)
    n = len(sorted_nums)
    mid = n // 2
    if n % 2 == 1:
        return sorted_nums[mid]
    else:
        return (sorted_nums[mid - 1] + sorted_nums[mid]) / 2


def mode(nums):
    counts = Counter(nums)
    most_common = counts.most_common()
    max_freq = most_common[0][1]
```

```
    # get all values with highest frequency and pick the smallest
    modes = [val for val, cnt in most_common if cnt == max_freq]
    return min(modes)

print("Numbers:", numbers)
print("Mean:", mean(numbers))
print("Median:", median(numbers))
print("Mode:", mode(numbers))
```

Output:

Numbers: [12, 15, 12, 18, 19, 12, 20, 22, 19, 19, 24, 24, 24, 26, 28]

Mean: 19.6

Median: 19

Mode: 12

Question 6: Compute the covariance and correlation coefficient between the following two datasets provided as lists in Python: list_x = [10, 20, 30, 40, 50] list_y = [15, 25, 35, 45, 60]

Answer:

```
import math
list_x = [10, 20, 30, 40, 50]
list_y = [15, 25, 35, 45, 60]
def mean(data):
    return sum(data) / len(data)
def covariance(x, y):
```

```
    n = len(x)
    mx = mean(x)
    my = mean(y)
    cov = sum((xi - mx)*(yi - my) for xi, yi in zip(x, y)) / (n - 1)
    return cov
def stddev(data):
    m = mean(data)
    return math.sqrt(sum((xi - m)**2 for xi in data) / (len(data) - 1))
def correlation(x, y):
    return covariance(x, y) / (stddev(x) * stddev(y))
cov_xy = covariance(list_x, list_y)
corr_xy = correlation(list_x, list_y)
print("List X:", list_x)
print("List Y:", list_y)
print(f"Covariance: {cov_xy:.2f}")
print(f"Correlation coefficient (Pearson r): {corr_xy:.4f}")
```

Output:

List X: [10, 20, 30, 40, 50]

List Y: [15, 25, 35, 45, 60]

Covariance: 212.50

Correlation coefficient (Pearson r): 0.9937

Question 7: Write a Python script to draw a boxplot for the following numeric list and identify its outliers. Explain the result: data = [12, 14, 14, 15, 18, 19, 19, 21, 22, 22, 23, 23, 24, 26, 29, 35]

Answer:

```
import matplotlib.pyplot as plt

import numpy as np


data = [12, 14, 14, 15, 18, 19, 19, 21, 22, 22, 23, 23, 24, 26, 29, 35]

# Calculate quartiles and IQR

Q1 = np.percentile(data, 25)

Q3 = np.percentile(data, 75)

IQR = Q3 - Q1

lower_bound = Q1 - 1.5 * IQR

upper_bound = Q3 + 1.5 * IQR


# Identify outliers

outliers = [x for x in data if x < lower_bound or x > upper_bound]

non_outliers = [x for x in data if lower_bound <= x <= upper_bound]


print("Data:", data)

print(f"Q1 = {Q1}, Q3 = {Q3}, IQR = {IQR}")

print(f"Lower bound = {lower_bound:.2f}, Upper bound = {upper_bound:.2f}")

print("Outliers identified:", outliers)

# Plot the boxplot

plt.boxplot(data, vert=True, patch_artist=True, boxprops=dict(facecolor='lightblue'))
```

```
plt.title("Boxplot with Outliers Highlighted")

plt.ylabel("Values")

plt.show()
```

Output:

Data: [12, 14, 14, 15, 18, 19, 19, 21, 22, 22, 23, 23, 24, 26, 29, 35]

Q1 = 18.75, Q3 = 24.5, IQR = 5.75

Lower bound = 9.62, Upper bound = 33.63

Outliers identified: [35]

Question 8: You are working as a data analyst in an e-commerce company. The marketing team wants to know if there is a relationship between advertising spend and daily sales. ● Explain how you would use covariance and correlation to explore this relationship. ● Write Python code to compute the correlation between the two lists: advertising_spend = [200, 250, 300, 400, 500] daily_sales = [2200, 2450, 2750, 3200, 4000]

Answer:

```
import numpy as np

from scipy.stats import pearsonr

# Data

advertising_spend = [200, 250, 300, 400, 500]

daily_sales = [2200, 2450, 2750, 3200, 4000]

# Compute Pearson correlation coefficient

corr, _ = pearsonr(advertising_spend, daily_sales)

print(f"Pearson correlation coefficient: {corr:.4f}")
```

Output:

Pearson correlation coefficient: 0.9982

Question 9: Your team has collected customer satisfaction survey data on a scale of 1-10 and wants to understand its distribution before launching a new product. ● Explain which summary statistics and visualizations (e.g. mean, standard deviation, histogram) you'd use. ● Write Python code to create a histogram using Matplotlib for the survey data:
survey_scores = [7, 8, 5, 9, 6, 7, 8, 9, 10, 4, 7, 6, 9, 8, 7]

Answer:

```
import numpy as np

import matplotlib.pyplot as plt

from scipy import stats

# Survey data

survey_scores = [7, 8, 5, 9, 6, 7, 8, 9, 10, 4, 7, 6, 9, 8, 7]

# Compute summary statistics

mean_score = np.mean(survey_scores)

median_score = np.median(survey_scores)

mode_score = stats.mode(survey_scores)[0][0]

std_dev = np.std(survey_scores, ddof=1)

# Print summary statistics

print(f"Mean: {mean_score}")

print(f"Median: {median_score}")

print(f"Mode: {mode_score}")

print(f"Standard Deviation: {std_dev}")

# Plot histogram

plt.hist(survey_scores, bins=range(1, 12), edgecolor='black', alpha=0.7)

plt.title("Customer Satisfaction Survey Scores")

plt.xlabel("Score")

plt.ylabel("Frequency")

plt.grid(True)
```

plt.show()

Output:

survey_scores = [7, 8, 5, 9, 6, 7, 8, 9, 10, 4, 7, 6, 9, 8, 7]

Computed results:

Mean: 7.333333333333333

Median: 7.0

Mode: 7

Standard Deviation: 1.6330 (approximately)