PRACTICAL COURSE ON

Python with Statistics

AFTERNOON SESSION HANDBOOK

Case Study: Titanic Survival Analysis

Part 1: Descriptive Statistics in Action
Part 2: Hypothesis Testing in Action

Afternoon Session Overview

Welcome to the afternoon session! This morning you learned Python and Pandas fundamentals. Now you'll apply those skills to answer real questions using the famous Titanic dataset.

The Titanic Dataset

On April 15, 1912, the RMS Titanic sank after hitting an iceberg. Of the 2,224 passengers and crew aboard, only 722 survived. This dataset contains information about passengers: their demographics, ticket class, fare paid, and whether they survived.

Why this dataset?

- Rich and varied: Mix of numeric (age, fare) and categorical (class, sex) variables
- Clear outcome: Survived (yes/no) — perfect for statistical testing
- Real-world relevance: Raises questions about fairness and social inequality
- Engaging story: Everyone knows the Titanic

Session Structure

13:30-15:00	**Part 1: Descriptive Statistics (90 min)**
	• Explore the dataset
	• Distribution analysis
	• Central tendency and variability
	• Group comparisons
	• Visual exploration
15:00-15:15	☕ Afternoon Break
15:15-16:45	Part 2: Hypothesis Testing (90 min)
	• Independent t-test
	• Mann-Whitney U test
	• Chi-square test
	• Binomial test
16:45-17:00	Wrap-up & Next Steps

Learning Approach

We'll follow a question-driven approach:

1. Ask a business/research question
2. Load and explore the relevant data
3. Apply statistical techniques step-by-step
4. Visualize the results
5. Interpret findings in plain language
6. Practice with checkpoint exercises

💡 **TIP:** This mirrors how real data analysts work. You don't memorize formulas—you ask questions and use tools to find answers.

Part 1: Descriptive Statistics in Action

⏰ Time: 13:30 - 15:00 (90 minutes)

What Are Descriptive Statistics?

Descriptive statistics summarize and describe data. They answer questions like:

- What's typical? (Central tendency: mean, median, mode)
- How spread out are the values? (Variability: standard deviation, range)
- What does the distribution look like? (Shape: skewness, outliers)
- How do different groups compare? (Group comparisons)

Before you can test hypotheses, you must first understand your data. That's what descriptive statistics are for.

Setup: Loading the Titanic Dataset

📋 **Initial Setup:**

```
# Import libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# Load Titanic dataset
df = sns.load_dataset('titanic')

# First look
print("Dataset shape:", df.shape)
print("\nFirst 5 rows:")
print(df.head())
```

Output:

```
Dataset shape: (891, 15)

First 5 rows:
   survived  pclass     sex   age  sibsp  parch     fare embarked  class
who  adult_male deck  embark_town alive  alone
0         0       3    male  22.0      1      0   7.2500        S  Third
man        True  NaN  Southampton    no  False
1         1       1  female  38.0      1      0  71.2833        C  First
woman       False    C    Cherbourg   yes  False
2         1       3  female  26.0      0      0   7.9250        S  Third
woman       False  NaN  Southampton   yes   True
3         1       1  female  35.0      1      0  53.1000        S  First
woman       False    C  Southampton   yes  False
4         0       3    male  35.0      0      0   8.0500        S  Third
man        True  NaN  Southampton    no   True
```

Understanding the Variables

Let's understand what each column means:

Variable	Meaning
survived	0 = Died, 1 = Survived (our outcome variable)

pclass	Passenger class: 1 = First, 2 = Second, 3 = Third
sex	Gender: male or female
age	Age in years
sibsp	Number of siblings/spouses aboard
parch	Number of parents/children aboard
fare	Ticket price in British pounds
embarked	Port of embarkation: S = Southampton, C = Cherbourg, Q = Queenstown
class	Passenger class (same as pclass, just in words)
who	man, woman, or child
deck	Deck number (mostly missing)
alive	yes or no (same as survived)

Initial Exploration

📋 **Explore the data:**

```
# Check for missing values
print("Missing values per column:")
print(df.isnull().sum())

# Statistical summary
print("\nStatistical summary:")
print(df.describe())
```

Output:

```
Missing values per column:
survived         0
pclass           0
sex              0
age            177
sibsp            0
parch            0
fare             0
embarked         2
class            0
who              0
adult_male       0
deck           688
embark_town      2
alive            0
alone            0
dtype: int64

Statistical summary:
       survived      pclass         age       sibsp       parch        fare
count  891.000000  891.000000  714.000000  891.000000  891.000000  891.000000
mean     0.383838    2.308642   29.699118    0.523008    0.381594   32.204208
std      0.486592    0.836071   14.526497    1.102743    0.806057   49.693429
min      0.000000    1.000000    0.420000    0.000000    0.000000    0.000000
25%      0.000000    2.000000   20.125000    0.000000    0.000000    7.910400
50%      0.000000    3.000000   28.000000    0.000000    0.000000   14.454200
75%      1.000000    3.000000   38.000000    1.000000    0.000000   31.000000
max      1.000000    3.000000   80.000000    8.000000    6.000000  512.329200
```

🔑 **KEY INSIGHT:** Notice: 177 missing ages, 688 missing deck numbers, 2 missing

embarkation ports. We'll need to handle this in our analysis.

Question 1: What Does the Age Distribution Look Like?

BUSINESS QUESTION

Understanding the age distribution helps us know who was aboard the Titanic. Were most passengers young adults? Were there many children? This context matters when analyzing survival patterns.

Step 1: Visualize the Distribution

Create histogram:

```
# Histogram of age distribution
sns.histplot(data=df, x='age', bins=30, kde=True, color='skyblue',
edgecolor='black')
plt.title('Age Distribution of Titanic Passengers', fontsize=14,
fontweight='bold')
plt.xlabel('Age (years)', fontsize=12)
plt.ylabel('Number of Passengers', fontsize=12)
plt.axvline(df['age'].mean(), color='red', linestyle='--', linewidth=2,
label='Mean Age')
plt.axvline(df['age'].median(), color='green', linestyle='--', linewidth=2,
label='Median Age')
plt.legend()
plt.show()
```

INTERPRETATION

The histogram shows:

- Most passengers were between 20-40 years old
- The distribution is right-skewed (long tail toward older ages)
- Mean age (red line) is slightly higher than median (green line) due to older outliers
- Small peak around age 0-5 (young children)

Step 2: Calculate Distribution Statistics

Descriptive statistics:

```
# Remove missing ages for accurate calculation
age_clean = df['age'].dropna()

# Calculate statistics
mean_age = age_clean.mean()
median_age = age_clean.median()
std_age = age_clean.std()
min_age = age_clean.min()
max_age = age_clean.max()
range_age = max_age - min_age

print(f"Mean age: {mean_age:.2f} years")
print(f"Median age: {median_age:.2f} years")
print(f"Standard deviation: {std_age:.2f} years")
print(f"Range: {min_age:.2f} to {max_age:.2f} years ({range_age:.2f} years)")
```

Output:

```
Mean age: 29.70 years
Median age: 28.00 years
Standard deviation: 14.53 years
Range: 0.42 to 80.00 years (79.58 years)
```

INTERPRETATION

- Average passenger was about 30 years old
- Half of passengers were younger than 28 (median)

• Typical variation from average: ±14.5 years (std dev)
• Age range: from infants (0.42 years) to elderly (80 years)

Step 3: Understand Skewness

📋 **Check skewness:**

```
# Calculate skewness
from scipy import stats

age_clean = df['age'].dropna()
skewness = age_clean.skew()

print(f"Skewness: {skewness:.2f}")

if skewness > 0:
    print("The distribution is right-skewed (tail extends toward higher
values)")
elif skewness < 0:
    print("The distribution is left-skewed (tail extends toward lower
values)")
else:
    print("The distribution is symmetric")
```

Output:

```
Skewness: 0.39
The distribution is right-skewed (tail extends toward higher values)
```

💡 **TIP:** Skewness tells you about distribution shape:
• Skewness ≈ 0: Symmetric (bell-shaped)
• Skewness > 0: Right-skewed (mean > median)
• Skewness < 0: Left-skewed (mean < median)

✓ CHECKPOINT EXERCISE 1

Analyze the distribution of 'fare' (ticket price):
1. Create a histogram with KDE curve
2. Calculate mean, median, std, min, max
3. Calculate skewness
4. Is the fare distribution skewed? Why might this be?

💡 *Hint:* Use df['fare'].dropna() and follow the same pattern as age analysis

Question 2: What Was the Typical Survival Rate?

🎯 BUSINESS QUESTION

The survival rate is the proportion of passengers who survived. This gives us a baseline before we investigate which factors affected survival.

Step 1: Calculate Overall Survival Rate

📋 Calculate survival rate:

```
# Count survivors
total_passengers = len(df)
survivors = df['survived'].sum()  # survived is 0 or 1, so sum gives count
survival_rate = (survivors / total_passengers) * 100

print(f"Total passengers: {total_passengers}")
print(f"Survivors: {survivors}")
print(f"Deaths: {total_passengers - survivors}")
print(f"Survival rate: {survival_rate:.2f}%")
```

Output:

```
Total passengers: 891
Survivors: 342
Deaths: 549
Survival rate: 38.38%
```

📊 INTERPRETATION

Only 38.38% of passengers survived. This means roughly 6 out of 10 passengers died. This is our baseline—now we can investigate which groups had better or worse survival rates.

Step 2: Visualize Survival Counts

📋 Visualize survival:

```
# Bar chart of survival counts
sns.countplot(data=df, x='alive', palette=['red', 'green'])
plt.title('Titanic Survival Counts', fontsize=14, fontweight='bold')
plt.xlabel('Survived', fontsize=12)
plt.ylabel('Number of Passengers', fontsize=12)

# Add count labels on bars
for i, count in enumerate(df['alive'].value_counts()):
    plt.text(i, count + 10, str(count), ha='center', fontsize=12,
fontweight='bold')

plt.show()
```

Step 3: Survival Rate by Passenger Class

📋 Compare by class:

```
# Survival rate by class
survival_by_class = df.groupby('class')['survived'].agg(['sum', 'count',
'mean'])
survival_by_class['survival_rate'] = survival_by_class['mean'] * 100
survival_by_class = survival_by_class.rename(columns={
    'sum': 'Survivors',
    'count': 'Total',
    'survival_rate': 'Survival Rate (%)'
})
```

```
print(survival_by_class[['Survivors', 'Total', 'Survival Rate (%)']])
```

Output:

```
       Survivors  Total  Survival Rate (%)
class
First          136    216          62.962963
Second          87    184          47.282609
Third          119    491          24.236253
```

INTERPRETATION

MAJOR finding:
• First class: 63% survived
• Second class: 47% survived
• Third class: Only 24% survived

First-class passengers were nearly 3 times more likely to survive than third-class passengers. This suggests class significantly affected survival chances.

Step 4: Visualize Survival by Class

Create bar chart:

```
# Grouped bar chart
survival_pct = df.groupby('class')['survived'].mean() * 100

plt.figure(figsize=(10, 6))
bars = plt.bar(survival_pct.index, survival_pct.values, color=['gold',
'silver', 'brown'],
               edgecolor='black', linewidth=1.5)
plt.title('Survival Rate by Passenger Class', fontsize=14, fontweight='bold')
plt.xlabel('Passenger Class', fontsize=12)
plt.ylabel('Survival Rate (%)', fontsize=12)
plt.ylim(0, 100)

# Add percentage labels on bars
for bar, pct in zip(bars, survival_pct.values):
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2., height + 2,
             f'{pct:.1f}%', ha='center', fontsize=12, fontweight='bold')

plt.axhline(y=38.38, color='red', linestyle='--', linewidth=2, label='Overall
Survival Rate')
plt.legend()
plt.show()
```

✓ CHECKPOINT EXERCISE 2

Calculate and visualize survival rates by gender:
1. Group by 'sex' and calculate survival rate
2. Create a bar chart showing survival rate by gender
3. What do you observe? Which gender had a higher survival rate?

Hint: Use df.groupby('sex')['survived'].mean() * 100

Question 3: How Much Did Ticket Prices Vary?

🎯 BUSINESS QUESTION
Understanding price variability tells us about the economic diversity of passengers. Were most tickets similarly priced, or was there huge variation?

Step 1: Measures of Variability

📋 **Calculate variability:**

```
# Calculate variability measures
fare_clean = df['fare'].dropna()

mean_fare = fare_clean.mean()
median_fare = fare_clean.median()
std_fare = fare_clean.std()
var_fare = fare_clean.var()
min_fare = fare_clean.min()
max_fare = fare_clean.max()
q1 = fare_clean.quantile(0.25)
q3 = fare_clean.quantile(0.75)
iqr = q3 - q1

print(f"Mean fare: £{mean_fare:.2f}")
print(f"Median fare: £{median_fare:.2f}")
print(f"Standard deviation: £{std_fare:.2f}")
print(f"Variance: £{var_fare:.2f}")
print(f"Range: £{min_fare:.2f} to £{max_fare:.2f}")
print(f"Interquartile range (IQR): £{iqr:.2f}")
```

Output:

```
Mean fare: £32.20
Median fare: £14.45
Standard deviation: £49.69
Variance: £2469.44
Range: £0.00 to £512.33
Interquartile range (IQR): £23.09
```

📊 INTERPRETATION
- Average fare was £32, but median was only £14 (huge right skew!)
- Standard deviation (£49.69) is larger than the mean—indicates extreme variability
- Some passengers paid nothing (£0), others paid £512 (1,024x more!)
- The middle 50% of passengers paid between £7.91 and £31 (IQR)

💡 **TIP:** When std deviation > mean, you have high variability and likely outliers. When mean >> median, the distribution is right-skewed (pulled up by high values).

Step 2: Visualize with Box Plot

📋 **Create box plot:**

```
# Box plot of fares
plt.figure(figsize=(10, 6))
sns.boxplot(y=df['fare'], color='lightblue')
plt.title('Distribution of Ticket Fares (Box Plot)', fontsize=14,
fontweight='bold')
plt.ylabel('Fare (£)', fontsize=12)
plt.show()
```

📊 INTERPRETATION
The box plot shows:

• Box (Q1 to Q3): Middle 50% of fares between ~£8 and ~£31
• Median line: £14.45
• Whiskers: Extend to ~£65 (1.5 × IQR rule)
• Many dots above: Outliers (expensive first-class tickets)

Step 3: Compare Fare by Class

📋 **Compare across groups:**

```
# Box plot comparing fare across classes
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='class', y='fare', palette='Set2')
plt.title('Ticket Fare by Passenger Class', fontsize=14, fontweight='bold')
plt.xlabel('Passenger Class', fontsize=12)
plt.ylabel('Fare (£)', fontsize=12)
plt.show()

# Statistical summary by class
print("\nFare statistics by class:")
print(df.groupby('class')['fare'].describe())
```

Output:

```
Fare statistics by class:
         count        mean         std   min     25%     50%       75%       max
class
First    216.0   84.154687   78.380373   0.0  30.923  60.2875   93.5000  512.3292
Second   184.0   20.662183   13.417399   0.0  13.000  14.2500   26.0000   73.5000
Third    491.0    13.675550   11.778142   0.0   7.750   8.0500   15.5000   69.5500
```

📊 **INTERPRETATION**

Clear class differences:
• First class: Average £84, some paid over £500
• Second class: Average £21, much more consistent pricing
• Third class: Average £14, tightly clustered around £8-£15

The variability in first-class fares (std = £78) was enormous—some luxury suites, some modest first-class tickets.

✓ **CHECKPOINT EXERCISE 3**

Analyze age variability by passenger class:
1. Create a box plot showing age distribution by class
2. Calculate mean and std of age for each class
3. Which class had the youngest passengers on average?

💡 *Hint:* Use sns.boxplot(data=df, x='class', y='age') and df.groupby('class')['age'].agg(['mean', 'std'])