Laboratory 03

Table of contents

1 Reply on the feedback of Lab 2

Thanks for the feedback on my Lab 2 Assignment! Here's my response:

1.1 *Normally* on Line 4, Section 3.2 Question 2, Page 5

> "normally" have higher median?

What I mean is that based on the sample and the box-plot, we can generally infer ~~(AND I TRULY BELIEVE)~~ that "cat people have a higher median" if we put the same measurement on a larger sample.

Apologies for the earlier miswording and ambiguity in the interpretation.

1.2 About the cats

They usually come out in the early morning and late at night, as cats normally do. However, they are all around at various times. I even once witnessed the police and firefighters at our faculty, rescuing a young cat trapped on the roof!

2 Main scenario story (Context)

Figure 1: Warning - Health and Safety

There are reports of increases in injuries related to playing games consoles. These injuries were attributed mainly to muscle and tendon strains. A researcher collected data from 120 participants who played on a Nintendo Switch or watched others playing. The outcome was a pain score from 0 to 10, where 0 is no pain and 10 is severe pain. The data are in `switch.sav`.

3 Main scenario quests (Objectives)

1. Describe the distribution of the pain scores. Do you think the one-sample t test is suitable for this variable?
2. A pain score of 2 is considered as minor pain. Test whether the mean pain score is equal to 2 (two tailed, 5% level) and obtain the corresponding effect size.
3. Obtain a 95% confidence interval for the mean pain score.
4. Summary your findings from the previous questions in several sentences.

5. (Extra credit) Obtain a 95% confidence interval for the mean pain score of those who played on a Nintendo Switch. That is, exclude those who only watched others playing. (Hint: You learned how to exclude cases in Laboratory Assignment 1.)

4 Solutions

4.1 A peek on the dataset

As usual, I load modules that I may need in this laboratory assignment, then the dataset to my RAM and check attributes of the given dataset.

```python
import pandas as pd

# Load the dataset
switch = pd.read_spss('./datasets/switch.sav')
```

```python
# Descriptions
print(f'Shape: \n', switch.shape, '\n')
print(f'Columns: \n', switch.columns, '\n')
print(f'First 5 rows: \n', switch.head(5), '\n')
print(f'Describe the column `injury`: \n', switch.describe(), '\n')
```

```
Shape:
 (120, 5)

Columns:
 Index(['id', 'athlete', 'stretch', 'switch', 'injury'], dtype='object')

First 5 entries:
     id  athlete     stretch          switch  injury
0  ytv  Athlete  Stretching  Playing switch    2.0
1  wel  Athlete  Stretching  Playing switch    2.0
2  qfs  Athlete  Stretching  Playing switch    1.0
3  oln  Athlete  Stretching  Playing switch    2.0
4  wxi  Athlete  Stretching  Playing switch    0.0

Describe:
             injury
count  120.000000
mean     2.891667
std      1.994934
min      0.000000
```

```
25%        2.000000
50%        2.000000
75%        4.000000
max       10.000000
```

4.2 Q1: Distribution of the pain scores

4.2.1 Q1a: Describe the distribution

Answer

To describe the distribution of the pain scores, I use histogram with a kernel density estimation curve as shown in Figure 2 as well as a description of central tendency with mean, mode and median (see **?@tbl-centrality**).

```python
#| label: tbl-centrality
#| tbl-cap: Descriptive Statistics of this Dataset

from IPython.display import Markdown
from tabulate import tabulate

table = [["Mode","2.00"],
         ["Mean","2.89"],
         ["Median","2.00"]]

Markdown(
    tabulate(
        table,
        headers=["Measurement","Value"]
        )
    )
```

According to the graph:

1. Most of the observations are clustered around the lower pain scores (between 1 and 4), we can say that the distribution of pain scores is positively skewed rather than a perfect normal distribution.
2. There is a noticeable peak at a score of 2, which means the most frequent score is around 2.
3. A long tail extends to the higher scores, indicating the frequency of pain scores gradually decreases as the scores increase.

Solution

```python
injury = switch['injury']

# Calculate measurements of central tendency
injury_mean = injury.mean()
injury_mode = injury.mode()[0]
injury_median = injury.median()

# Tell the result
print(f'Central Tendency: \n')
print(f'Mean: ', injury_mean)
print(f'Mode: ', injury_mode)
print(f'Median: ', injury_median)
```

```
Central Tendency:

Mean:  2.8916666666666666
Mode:  2.0
Median:  2.0
```

```python
import matplotlib.pyplot as plt
import seaborn as sns

# Plot the histogram
injury_hist = sns.histplot(switch, x='injury', stat='count', bins=10 ,kde=True)
# Dashed line for Mean, Median and Mode
injury_hist.axvline(injury_mean, color='blue', linestyle='--', linewidth=1)
injury_hist.axvline(injury_median, color='red', linestyle='--', linewidth=1)
# Set title and labels
injury_hist.set_title('Distribution of the pain scores')
injury_hist.set_xlabel('Pain score (out of 10)')
# Show the plot
plt.show()
```

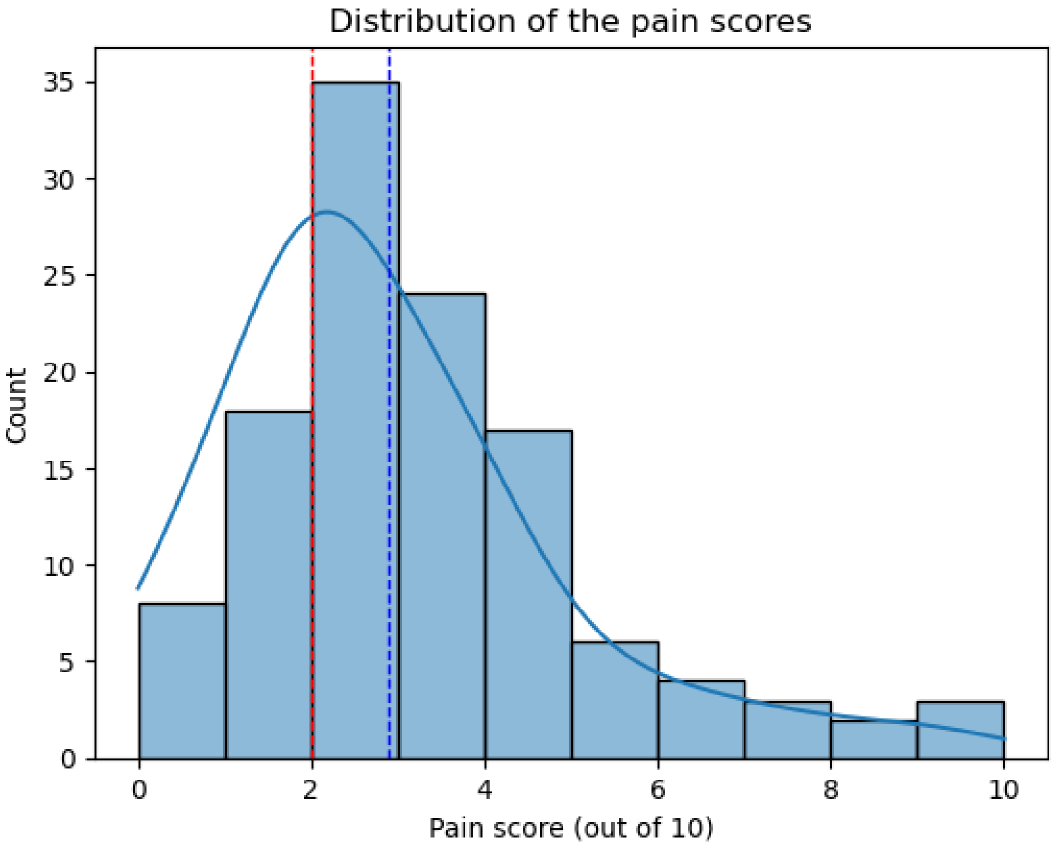

Figure 2: Distribution of the pain scores

4.2.2 Q1b: One-sample t test on this variable

Answer

Recall back the the slides in the lecture notes:

> One - sample ☒ test requires that:
>
> - Sample mean describes central tendency.

The sample mean is slightly higher than the mode and median (see **?@tbl-centrality** and Figure 2, red dashed lines for the median and mode, blue for the mean) since the data is right-skewed. However, they are fairly close to each other, so the sample mean can still represent central tendency.

> - Scores in the sample are randomly selected from the population

According to the description, the data was "collected from 120 participants who played on a Nintendo Switch or watched others playing." For the sake of this assignment, I will assume that the participants were randomly selected from patients worldwide to fulfill the random sampling assumption.

- Either N is large or X follows a normal distribution

Given the right-skewed distribution as seen in Figure 2, the data may violate the assumption of normality required for the one-sample t-test. However, the Central Limit Theorem suggests that if the sample size is large (typically $N > 30$), the sampling distribution of the sample mean tends to approach normality. Therefore, despite the skewed distribution, the sample size ($N = 120$) makes the one-sample t-test acceptable in this case.

Additionally, question 2 specifically asks for a one-sample t-test without requiring further preprocessing of data (e.g., a log transformation), which further supports the applicability of the one-sample t-test to this data. If the data were unusable, there would be no reason to include the following questions.

In conclusion, a one-sample t-test is suitable for this dataset.

4.3 Question 2: Mean of pain score tested

Answer

1. p value

   $p \approx 3.11 \times 10^{-6}$

   At a 5% significance level ($\alpha = 0.05$), $p < 0.001$, we reject the null hypothesis. The mean pain score is significantly different from 2 (a minor pain) at the 5% level.

2. The Cohen's d value

   $d \approx 0.45$

   The Cohen's d value indicates a medium effect. This suggests that the difference between the mean pain score ($M = 2.89$) and the a minor pain ($\mu_{hyp} = 2$) is meaningful in practical terms.

Solution

Given $N = 120$, $M \approx 2.89$, $SD \approx 1.99$, $\mu_{hyp} = 2$, the standard error SE_M is:

$$SE_M = \frac{SD}{\sqrt{N}} \approx 0.18$$

```python
from math import sqrt

# Standard Error Mean
# Note: I can use injury.sem() directly to get the result,
# but I shall calculate by my own for this assignment.

injury_sem = injury.std(ddof=1) / sqrt(120)
print(f'Standard Error Mean: ', injury_sem)
```

```
Standard Error Mean:  0.1821117309227567
```

With $SE_M \approx 0.18$, the t ratio is:

$$t = \frac{M - \mu_{hyp}}{SE_M} \approx 4.90$$

```python
# t statistic
injury_t = (injury.mean() - 2) / injury.sem()
print(f't: ', injury_t)
```

```
t:  4.8962615540943375
```

Unfortunately, I can' t calculate the p-value on hand, so in this part I' ll call `scipy.stats.t` for help. the degree of freedom (df) is:

$$df = N - 1 = 120 - 1 = 119$$

With $t \approx 4.90$ and $df = 119$, then use survivor function to reach the p-value:

$$p \approx 3.11 \times 10^{-6}$$

```python
import scipy.stats as stats

injury_p = stats.t.sf(injury_t, 119 ) * 2 # 119 is the degree of freedom; Two-sided times

print(f'p: ', injury_p)
```

```
p:  3.1051091723547962e-06
```

The p-value is much smaller than 0.001 ($p < 0.001$), the null hypothesis should be rejected.

I also did a sanity check with the ready-to-use function `scipy.stats.ttest_1samp`:

```python
# A san-check on my calculation result:

injury_ttest_1samp = stats.ttest_1samp(injury, 2, alternative='two-sided')

print(f't: ', injury_ttest_1samp.statistic, '\n'
      'df: ', injury_ttest_1samp.df, '\n'
      'p-value: ', injury_ttest_1samp.pvalue)
```

```
t:  4.8962615540943375
df:  119
p-value:  3.1051091723547962e-06
```

The Cohen' s d value is:

$$d = \frac{M - \mu_{hyp}}{SD} = \frac{t}{\sqrt{N}} \approx 0.45$$

```python
# Effect Size d

injury_d = injury_t / sqrt(120)
print(f'Cohen\'s d:', injury_d)
```

```
Cohen's d: 0.4469654834371283
```

4.4 Q3: 95% confidence interval for the mean pain score:

Answer

Based on the sample of $N = 120$ pain scores, with $M = 2.89$ and $SD = 1.99$, the 95% CI for pain scores is $[2.53, 3.24]$.

Solution

Given $c = 1.96$ for a 95% confidence interval and $SE_M \approx 0.18$ as calculated in the last section, a 95% confidence interval of the mean pain score is:

$$[M - c \times SE_M, M + c \times SE_M] \approx [2.89 - 1.96 \times 0.18, 2.89 + 1.96 \times 0.18] \approx [2.53, 3.25]$$

```python
# CI for two-tailed t-statistics
def confidence_interval_2tailed(alpha, mean, sem, df):
    c = stats.t.interval(1 - alpha, df)[1]
    ci_upper = mean + (c * sem)
    ci_lower = mean - (c * sem)
    print(f'CI (Lower): ', ci_lower)
    print(f'CI (Upper): ', ci_upper)

confidence_interval_2tailed(0.05, injury_mean, injury_sem, 119)
```

```
CI (Lower):  2.53106725077079
CI (Upper):  3.252266082562543
```

5 Q4:

```python
import scipy.stats as stats
from math import sqrt

def one_sample_ttest(data, popmean, alpha=0.05, tails='two-tailed'):
    """
    Perform a one-sample t-test on the given data.

    Parameters:
    - data: list or numpy array of sample data
    - popmean: population mean to test against
    - alpha: significance level (default is 0.05)
    - tails: 'one-tailed' or 'two-tailed' (default is 'two-tailed')

    Returns:
    A dictionary containing the t statistic, df, p-value, significance stars, confidence i
    """

    # Calculate descriptive statistics
    N = len(data)
    sample_mean = np.mean(data)
    sample_std = np.std(data, ddof=1)  # ddof=1 for sample standard deviation
    std_err_mean = sample_std / sqrt(N)

    # Calculate t-statistic and p-value
    t_stat = (sample_mean - popmean) / std_err_mean
    df = N - 1

```

```
    if tails == 'two-tailed':
        p_value = stats.t.sf(np.abs(t_stat), df) * 2  # Two-tailed p-value
    elif tails == 'one-tailed':
        p_value = stats.t.sf(np.abs(t_stat), df)  # One-tailed p-value
    else:
        raise ValueError("tails must be either 'one-tailed' or 'two-tailed'")

    # Calculate the confidence interval
    t_crit = stats.t.ppf(1 - alpha / 2 if tails == 'two-tailed' else 1 - alpha, df)
    margin_of_error = t_crit * std_err_mean
    ci_lower = sample_mean - margin_of_error
    ci_upper = sample_mean + margin_of_error

    # Determine significance level in SPSS style
    if p_value < 0.001:
        significance = '***'
    elif p_value < 0.01:
        significance = '**'
    elif p_value < 0.05:
        significance = '*'
    else:
        significance = 'n.s.'  # Not significant

    # Prepare the output dictionary
    result = {
        'N': N,
        'Mean': sample_mean,
        'Std': sample_std,
        'Std Error Mean': std_err_mean,
        't': t_stat,
        'df': df,
        'p-value': p_value,
        'Significance': significance,
        'Confidence Interval': (ci_lower, ci_upper) if tails == 'two-tailed' else (sample_
    }

    return result

result = one_sample_ttest(switch['injury'], 2)

for key, value in result.items():
    print(f'{key}: {value}')
```

```
N: 120
Mean: 2.8916666666666666
```

```
Std: 1.994934060254099
Std Error Mean: 0.1821117309227567
t: 4.8962615540943375
df: 119
p-value: 3.1051091723547962e-06
Significance: ***
Confidence Interval: (2.53106725077079, 3.252266082562543)
```

```python
switch.to_csv('switch.csv')
```