STAT 201B HW6

Oliver Maynard

Problem 2

Part a

First I converted the provided R code in pvals.R into a python chunk (preferance).

```
import numpy as np
from scipy import stats

n = 100  # Sample size in each simulation
B = 100000  # Number of simulations
nullprop = 0.99  # Proportion of times the null is actually true (mu = 0)
nullcount = int(np.floor(nullprop * B))
mu = np.concatenate([
    np.repeat(0, nullcount),
    np.random.uniform(-1, 1, B - nullcount)
])  # Can replace the uniforms with something else

pvals = np.zeros(B)
for i in range(B):
    x = np.random.normal(loc=mu[i], scale=1, size=n)
    w = np.mean(x) / np.sqrt(np.var(x, ddof=1) / n)
    pvals[i] = 2 * stats.norm.cdf(-np.abs(w))

check = np.where((pvals > 0.01) & (pvals < 0.05))[0]  # Indices for tests to look at
print(np.mean(check <= nullcount))  # Condition satisfied means H0 was true
```

Output: (Don't want to re run)

```
0.9843230403800475
```

Experiment 1: Very Wide - Unif(-5, 5)

```
import numpy as np
from scipy import stats

np.random.seed(42)

n = 100
B = 100000
nullprop = 0.99
nullcount = int(np.floor(nullprop * B))

# Very wide uniform
mu = np.concatenate([
    np.repeat(0, nullcount),
    np.random.uniform(-5, 5, B - nullcount)
])
```

```
pvals = np.zeros(B)
for i in range(B):
    x = np.random.normal(loc=mu[i], scale=1, size=n)
    w = np.mean(x) / np.sqrt(np.var(x, ddof=1) / n)
    pvals[i] = 2 * stats.norm.cdf(-np.abs(w))

check = np.where((pvals > 0.01) & (pvals < 0.05))[0]
false_discovery_prop = np.mean(check < nullcount)

print(f"Very Wide - Unif(-5, 5)")
print(f"False discovery proportion: {false_discovery_prop:.4f}")
print(f"Number of 'strong evidence' tests: {len(check)}")
```

```
Very Wide - Unif(-5, 5)
False discovery proportion: 0.9973
Number of 'strong evidence' tests: 4041
```

Experiment 2: Normal Distribution - N(0, 1.5)

```
import numpy as np
from scipy import stats

np.random.seed(42)

n = 100
B = 100000
nullprop = 0.99
nullcount = int(np.floor(nullprop * B))

# Normal distribution
mu = np.concatenate([
    np.repeat(0, nullcount),
    np.random.normal(0, 1.5, B - nullcount)
])

pvals = np.zeros(B)
for i in range(B):
    x = np.random.normal(loc=mu[i], scale=1, size=n)
    w = np.mean(x) / np.sqrt(np.var(x, ddof=1) / n)
    pvals[i] = 2 * stats.norm.cdf(-np.abs(w))

check = np.where((pvals > 0.01) & (pvals < 0.05))[0]
false_discovery_prop = np.mean(check < nullcount)

print(f"Normal - N(0, 1.5)")
print(f"False discovery proportion: {false_discovery_prop:.4f}")
print(f"Number of 'strong evidence' tests: {len(check)}")
```

```
Normal - N(0, 1.5)
False discovery proportion: 0.9928
Number of 'strong evidence' tests: 4026
```

Experiment 3: Constant - = 0.3

```python
import numpy as np
from scipy import stats

np.random.seed(42)

n = 100
B = 100000
nullprop = 0.99
nullcount = int(np.floor(nullprop * B))

# Constant value
mu = np.concatenate([
    np.repeat(0, nullcount),
    np.repeat(0.3, B - nullcount)
])

pvals = np.zeros(B)
for i in range(B):
    x = np.random.normal(loc=mu[i], scale=1, size=n)
    w = np.mean(x) / np.sqrt(np.var(x, ddof=1) / n)
    pvals[i] = 2 * stats.norm.cdf(-np.abs(w))

check = np.where((pvals > 0.01) & (pvals < 0.05))[0]
false_discovery_prop = np.mean(check < nullcount)

print(f"Constant -  = 0.3")
print(f"False discovery proportion: {false_discovery_prop:.4f}")
print(f"Number of 'strong evidence' tests: {len(check)}")
```

```
Constant -  = 0.3
False discovery proportion: 0.9519
Number of 'strong evidence' tests: 4200
```

Summary of Results The simulations demonstrate that when the null hypothesis is true 99% of the time, the vast majority (95-99.7%) of tests showing "strong evidence" ($0.01 <$ p-value < 0.05) are actually false discoveries. The wide uniform distribution Unif(-5,5) produced the highest false discovery rate at 99.7%, counterintuitively exceeding the original Unif(-1,1) at 98.4%. This occurs because larger true effect sizes generate very small p-values (< 0.01) rather than falling into the 0.01-0.05 range, leaving this "strong evidence" region even more dominated by false positives from true nulls. The Normal(0, 1.5) distribution showed similar behavior with 99.3% false discoveries. The constant effect size ($= 0.3$) produced the lowest false discovery proportion at 95.2%, though still alarmingly high. The fixed moderate effect ensures some truly non-null tests consistently land in the 0.01-0.05

range, providing genuine signal among the noise. This illustrates that even with seemingly convincing p-values, when most tested hypotheses are truly null, most "significant" findings will be false positives.

Part b - experiment changing null prop

From the first part I found that with null prop 0.99 that the proportion of "strong evidence against" that was actually from true nulls was `0.9843`

```
import numpy as np
from scipy import stats

n = 100
B = 100000

# Test different null proportions
nullprops = [0.90, 0.70, 0.50]

for nullprop in nullprops:
    np.random.seed(42)  # Same seed for fair comparison

    nullcount = int(np.floor(nullprop * B))

    mu = np.concatenate([
        np.repeat(0, nullcount),
        np.random.uniform(-1, 1, B - nullcount)
    ])

    pvals = np.zeros(B)
    for i in range(B):
        x = np.random.normal(loc=mu[i], scale=1, size=n)
        w = np.mean(x) / np.sqrt(np.var(x, ddof=1) / n)
        pvals[i] = 2 * stats.norm.cdf(-np.abs(w))

    check = np.where((pvals > 0.01) & (pvals < 0.05))[0]
    false_discovery_prop = np.mean(check < nullcount)

    print(f"nullprop = {nullprop}")
    print(f"False discovery proportion: {false_discovery_prop:.4f}")
    print(f"Number of 'strong evidence' tests: {len(check)}")
    print()
```

```
nullprop = 0.9
False discovery proportion: 0.8546
Number of 'strong evidence' tests: 4238

nullprop = 0.7
False discovery proportion: 0.6091
```

```
Number of 'strong evidence' tests: 4746

nullprop = 0.5
False discovery proportion: 0.4009
Number of 'strong evidence' tests: 5151
```

The false discovery proportion in the "strong evidence" range ($0.01 < \text{p-value} < 0.05$) is heavily dependent on the prior probability that the null hypothesis is true. When 99% of tested hypotheses are truly null (nullprop = 0.99), 98.4% of tests showing strong evidence are false discoveries. As the null proportion decreases, the false discovery rate drops substantially but remains surprisingly high: at nullprop = 0.90, still 85.5% are false discoveries; at nullprop = 0.70, it's 60.9%; and even when only half the hypotheses are truly null (nullprop = 0.50), 40.1% of "strong evidence" results are still false positives. This nearly linear relationship between the prior null probability and false discovery proportion illustrates the interpretation of p-values critically depends on the prior likelihood of the hypothesis being tested, not just the data itself.

Problem 3

LRT

```
import numpy as np
from scipy import stats

# H0: Admission status and population group are independent
# H1: Admission status and population group are NOT independent

X = np.array([
    [11252, 1110, 666],      # Row 1: Admitted
    [26830, 5199, 3593]      # Row 2: Not Admitted
])

n = np.sum(X)
row_totals = np.sum(X, axis=1)
col_totals = np.sum(X, axis=0)

lrt_lambda = 0

for i in range(X.shape[0]):        # Loop over rows
    for j in range(X.shape[1]):    # Loop over columns
        numerator = n * X[i, j]
        denominator = row_totals[i] * col_totals[j]
        lrt_lambda += 2 * X[i, j] * np.log(numerator / denominator)

print(f"Likelihood Ratio Test Statistic = {lrt_lambda:.4f}")

# Calculating p value using our test statistic
pvalue = 1 - stats.chi2.cdf(lrt_lambda, df=2)
print(f"p-value = {pvalue:.10f}")
```

```
Likelihood Ratio Test Statistic = 743.2693
p-value = 0.0000000000
```

Pearson Chi^2 Test

```
import numpy as np
from scipy import stats

# Data
X = np.array([
    [11252, 1110, 666],
    [26830, 5199, 3593]
])
```

```
# Expected counts
n = np.sum(X)
row_totals = np.sum(X, axis=1)
col_totals = np.sum(X, axis=0)
E = np.outer(row_totals, col_totals) / n

# Pearson's Chi-Squared Statistic
T = np.sum((X - E)**2 / E)

# P-value
pvalue = 1 - stats.chi2.cdf(T, df=2)

print(f"T = {T:.4f}")
print(f"p-value = {pvalue}")
```

```
T = 689.8647
p-value = 0.0
```

Problem 4

Part a

```
import numpy as np
from scipy import stats

X = np.array([
    [147, 186, 101],
    [25, 26, 11],
    [32, 39, 15],
    [94, 105, 37],
    [59, 74, 28],
    [18, 10, 10]
])

# Sample sizes for each novel (column totals)
n_j = np.sum(X, axis=0)  # [375, 440, 202]

# Total count for each word across all novels (row totals)
X_i_dot = np.sum(X, axis=1)

# Total word count
n = np.sum(X)

# LRT statistic
lrt_lambda = 0
for i in range(X.shape[0]):  # Loop over words
    for j in range(X.shape[1]):  # Loop over novels
        if X[i, j] > 0:
            lrt_lambda += 2 * X[i, j] * np.log((n * X[i, j]) / (n_j[j] * X_i_dot[i]))

# P-value
pvalue = 1 - stats.chi2.cdf(lrt_lambda, df=10)

print(f"  = {lrt_lambda:.4f}")
print(f"p-value = {pvalue:.6f}")
```

```
  = 12.5873
p-value = 0.247671
```

Part b

```
import numpy as np
from scipy import stats
```

```python
# Original data
X_original = np.array([
    [147, 186, 101, 83],
    [25, 26, 11, 29],
    [32, 39, 15, 15],
    [94, 105, 37, 22],
    [59, 74, 28, 43],
    [18, 10, 10, 4]
])

# Collapse: sum first three columns (Austen), keep fourth column (new author)
X = np.column_stack([
    np.sum(X_original[:, 0:3], axis=1),
    X_original[:, 3]
])

n_j = np.sum(X, axis=0)
X_i_dot = np.sum(X, axis=1)
n = np.sum(X)

# LRT statistic
lrt_lambda = 0
for i in range(X.shape[0]):
    for j in range(X.shape[1]):
        if X[i, j] > 0:
            lrt_lambda += 2 * X[i, j] * np.log((n * X[i, j]) / (n_j[j] * X_i_dot[i]))

# P-value
pvalue = 1 - stats.chi2.cdf(lrt_lambda, df=5)

print(f"  = {lrt_lambda:.4f}")
print(f"p-value = {pvalue:.6f}")
```

```
  = 31.7368
p-value = 0.000007
```