Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и вычислительной техники

Отчет

По лабораторной работе № 3

по дисциплине «Математическая статистика»

Вариант 3

Выполнила: Старостина Е. П., группа P3219

Преподаватель: Лимар И. А.

г. Санкт-Петербург
2025

Цель работы:

Попрактиковаться в проведении статистических тестов.

Задание 1:

Предположение (H_0): суммарный балл за вступительные экзамены имеет распределение Пуассона

H_1: not H_0

Результаты экзаменов – дискретные значения, используем критерий согласия Пирсона хи-квадрат (уровень значимости a=0.05)

Код:

```
from scipy.stats import chi2
from math import exp, log
from scipy.special import gammaln



def poisson_pmf(m, lambda_):
#     return ((lambda_ ** m) * (e ** (-lambda_))) / factorial(m)
    log_pmf = m * log(lambda_) - lambda_ - gammaln(m + 1)
    return exp(log_pmf)



with open('exams_dataset.csv', encoding='utf8') as f:
    data_raw = f.readlines()[1:]


data = []
for i in data_raw:
    line = i.strip('\n').split(',')
    data.append((int(line[5].strip('"')) + int(line[6].strip('"')) +
int(line[7].strip('"'))))



observed = dict()
for i in data:
    if i not in observed:
        observed[i] = 1
    else:
```

```
        observed[i] += 1

mean_moment = sum(data) / len(data)
lambda_poisson = mean_moment



expected = {}
n = len(data)
for key in observed.keys():
    expected[key] = poisson_pmf(int(key), lambda_poisson) * n

combined_observed = {}
combined_expected = {}
temp_obs = 0
temp_exp = 0
for k in sorted(observed.keys()):
    temp_obs += observed[k]
    temp_exp += expected[k]
    if temp_exp >= 5:
        combined_observed[k] = temp_obs
        combined_expected[k] = temp_exp
        temp_obs = 0
        temp_exp = 0
if temp_exp > 0:
    last_key = max(combined_observed.keys())
    combined_observed[last_key] += temp_obs
    combined_expected[last_key] += temp_exp



chi_square_stat = sum((combined_observed[k] - combined_expected[k])**2 /
combined_expected[k] for k in combined_observed)
degrees_of_freedom = len(combined_observed) - 1 - 1

critical_value = chi2.ppf(0.95, degrees_of_freedom)
p_value = chi2.sf(chi_square_stat, degrees_of_freedom)
print(f"p-value: {p_value}")
```

```
print(f"Хи-квадрат: {chi_square_stat}")
print(f"Критическое значение: {critical_value}")
if chi_square_stat < critical_value:
    print("Принять H0")
else:
    print("Отвергнуть H0")
```

Результат работы программы:

```
p-value: 0.0
Хи-квадрат: 15151.479527510113
Критическое значение: 77.93052380523042
Отвергнуть H0
```

Обработка с готовым решением:

```
a = list(combined_observed.values())
a[0] -= 3 * (10 ** -5)
chi_square_stat, p_value = chisquare(f_obs=a,
f_exp=list(combined_expected.values()))
degrees_of_freedom = len(combined_observed) - 1 - 1
critical_value = chi2.ppf(0.95, degrees_of_freedom)


print(f"p-value: {p_value}")
print(f"Хи-квадрат: {chi_square_stat}")
print(f"Критическое значение: {critical_value}")
if chi_square_stat < critical_value:
    print("Принять H0")
else:
    print("Отвергнуть H0")
```

Результат:

```
p-value: 0.0
Хи-квадрат: 15151.477323885101
Критическое значение: 77.93052380523042
Отвергнуть H0
```

Задание 2:

Воспользуемся f-test + t-test

H_0: дисперсии и мат ожидания совпадают

H_1: not H_0

Код:

```
from scipy.stats import t, f


with open('exams_dataset.csv', encoding='utf8') as file:
    data_raw = file.readlines()[1:]


data_math = []
data_reading = []
for i in data_raw:
    line = i.strip('\n').split(',')
    data_math.append((int(line[5].strip('"'))))
    data_reading.append((int(line[6].strip('"'))))


math_mean = sum(data_math) / len(data_math)
reading_mean = sum(data_reading) / len(data_reading)


math_var = sum((x - math_mean) ** 2 for x in data_math) / len(data_math)
reading_var = sum((x - reading_mean) ** 2 for x in data_reading) /
len(data_reading)


f_stat = max(math_var, reading_var) / min(math_var, reading_var)
t_stat = abs(math_mean - reading_mean) / ((math_var / len(data_math) +
reading_var / len(data_reading)) ** 0.5)


t_critical = t.ppf(1 - 0.05 / 2, len(data_math) + len(data_reading) - 2)
f_critical = f.ppf(1 - 0.05, len(data_math), len(data_reading))


p_value_t = 2 * (1 - t.cdf(abs(t_stat), len(data_math) + len(data_reading) -
2))
p_value_f = 1 - f.cdf(f_stat, len(data_math) - 1, len(data_reading) - 1)



print(f"Математика: мат ожидание {math_mean}, дисперсия {math_var}")
print(f"Чтение: мат ожидание {reading_mean}, дисперсия {reading_var}")


print(f"Статистика: f {f_stat}, t {t_stat}")
print(f"Критические значения: {f_critical}, {t_critical}")
```

```
print(f"p-value: f {p_value_f}, t {p_value_t}")
```

Результат работы программы:

```
Математика: мат ожидание 65.802, дисперсия 264.79279600000064
Чтение: мат ожидание 68.838, дисперсия 233.62375599999982
Статистика: f 1.1334155418681002, t 4.3003671829049
Критические значения: 1.1096882902429686, 1.9611520148367056
p-value: f 0.023963305294374404, t 1.787298520872227e-05
```

С использованием библиотек:

```
t_stat, p_value_t = ttest_ind(data_math, data_reading, equal_var=False)
math_mean = sum(data_math) / len(data_math)
math_var = sum((x - math_mean) ** 2 for x in data_math) / len(data_math)
reading_mean = sum(data_reading) / len(data_reading)
reading_var = sum((x - reading_mean) ** 2 for x in data_reading) /
len(data_reading)
f_stat = max(math_var, reading_var) / min(math_var, reading_var)
p_value_f = 1 - f.cdf(f_stat, len(data_math) - 1, len(data_reading) - 1)
t_critical = t.ppf(1 - 0.05 / 2, len(data_math) + len(data_reading) - 2)
f_critical = f.ppf(1 - 0.05, len(data_math), len(data_reading))

print(f"Математика: мат ожидание {math_mean}, дисперсия {math_var}")
print(f"Чтение: мат ожидание {reading_mean}, дисперсия {reading_var}")

print(f"Статистика: f {f_stat}, t {t_stat}")
print(f"Критические значения: {f_critical}, {t_critical}")

print(f"p-value: f {p_value_f}, t {p_value_t}")
```

Результат работы программы:

```
Математика: мат ожидание 65.802, дисперсия 264.79279600000064
Чтение: мат ожидание 68.838, дисперсия 233.62375599999982
Статистика: f 1.1334155418681002, t -4.298216461498611
Критические значения: 1.1096882902429686, 1.9611520148367056
p-value: f 0.023963305294374404, t 1.8048940569315806e-05
```

Вывод: отвергаем H_0

Задание 3:

H_0: correlation > 0 (есть корреляция между посещением курсов и баллом за экзамены)

H_1: not H_0

Код:

```
from scipy.stats import t


with open('exams_dataset.csv', encoding='utf8') as file:
    data_raw = file.readlines()[1:]

data_course = []
data_without_course = []
for i in data_raw:
    line = i.strip('\n').split(',')
    ex_res = int(line[5].strip('"')) + int(line[6].strip('"')) +
int(line[7].strip('"'))
    if line[4] == '"none"':
        data_without_course.append(ex_res)
    else:
        data_course.append(ex_res)

mean_course = sum(data_course) / len(data_course)
mean_without_course = sum(data_without_course) / len(data_without_course)

numerator = sum((x - mean_course) * (y - mean_without_course) for x, y in
zip(data_course, data_without_course))
denominator_x = sum((x - mean_course) ** 2 for x in data_course)
denominator_y = sum((y - mean_without_course) ** 2 for y in
data_without_course)

correlation = numerator / (denominator_x ** 0.5 * denominator_y ** 0.5)

t_stat_corr = (correlation * ((len(data_course) - 2) ** 0.5)) / ((1 -
correlation ** 2) ** 0.5)
p_value_corr = 2 * t.sf(abs(t_stat_corr), len(data_course) - 2)
critical_value_corr = t.ppf(1 - 0.05 / 2, len(data_course) - 2)

print(f"Коэффициент корреляции: {correlation}")
print(f"Статистика корреляции: {t_stat_corr}")
print(f"p-value корреляции: {p_value_corr}")
print(f"Критическое значение: {critical_value_corr}")
```

Результат работы программы:

```
Коэффициент корреляции: 0.05644061333356837

Статистика корреляции: 1.091789658671634

p-value корреляции: 0.2756302666831346

Критическое значение: 1.966344297279209
```

С использованием библиотек:

Код:

```
data_course.extend([*data_course])

data_course = data_course[:len(data_without_course)]
```

```
correlation, p_value_corr = pearsonr(data_course, data_without_course)

t_stat_corr = (correlation * ((len(data_course) - 2) ** 0.5)) / ((1 -
correlation ** 2) ** 0.5)

critical_value_corr = t.ppf(1 - 0.05 / 2, len(data_course) - 2)


print(f"Коэффициент корреляции: {correlation}")

print(f"Статистика корреляции: {t_stat_corr}")

print(f"p-value корреляции: {p_value_corr}")

print(f"Критическое значение: {critical_value_corr}")
```

Результат работы программы:

```
Коэффициент корреляции: 0.03745090345432241

Статистика корреляции: 0.9354295828679146

p-value корреляции: 0.3499293960674843

Критическое значение: 1.9637790861729503
```

Вывод: принимаем H_0

Выводы по работе:

В ходе лабораторной работы я попрактиковалась в проведении статистических тестов.

Приложение:

Ссылка на код:

https://github.com/Starostina-elena/math_stat_lab3