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 Р3219

Преподаватель: Лимар И. А.

Цель работы:

Попрактиковаться в проведении статистических тестов.

Задание 1:

Предположение (H0): суммарный балл за вступительные экзамены имеет распределение Пуассона

H1: not H0

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

H0: дисперсии и мат ожидания совпадают

H1: not H0

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 H0

Задание 3:

H0: correlation > 0 (есть корреляция между посещением курсов и баллом за экзамены)

H1: not H0

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 H0

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>