Задача 3
Вариант 1

от Татаринова Никиты Алексеевича

к 17.03.2021

Условие

В былые времена возле станции метро "Площадь Революции" проводились различные обследования (опросы и дегустации), за участие в которых можно было получить коробку конфет, шоколадку, бутылку пива или даже торт. Представьте себя на месте добрых людей, раздающих эти продукты.

Производитель газированной воды, планирующий продвижение нового товара (вода A), заказал вам малое обследование потенциальных потребителей. Вы уже собрали нужные данные и записали их в файл "Данные к задаче 3.ods":

- **a** – общая оценка респондентом воды A по семибалльной шкале (1 – совсем не понравилось, 7 – превосходно);
- **b** – оценка респондентом воды B (предполагаемого конкурента);
- **sex** – пол респондента (0 – мужской, 1 – женский).

Требуется ответить на 2 вопроса.

1. Есть ли основание считать, что предполагаемый потребитель предпочитает воду A воде B?
2. Связано ли отношение к воде A с полом потребителя? От этого зависит стратегия продвижения товара.

Для ответа на первый вопрос решено использовать критерий знаков, для ответа на второй – критерий ранговых сумм Уилкоксона. Выбран уровень значимости 10%.

Дайте ответы на поставленные вопросы. В каждом случае сформулируйте основную и альтернативную гипотезы, рассчитайте статистику, приведите критическое значение (или значения) и сделайте вывод.

Решение

В качестве языка программирования для решения данной задачи используется C++ (исходные коды представлены в виде файлов "question1.cpp" (для первого вопроса) и "question2.cpp" (для второго вопроса); скриншоты исходных кодов в приложении).

Вопрос 1

Введём обозначения:

- A_i – оценка воды A i-м человеком
- B_i – оценка воды B i-м человеком
- n^+ – число наблюдений, в которых $A_i > B_i$
- n^- – число наблюдений, в которых $B_i > A_i$

Основная гипотеза – у предполагаемого потребителя нет предпочтений между водой A и водой B. Альтернативная гипотеза – предполагаемый потребитель предпочитает воду A воде B.

$$\begin{cases} H_0\colon\ P\Big\{A_i > B_i\Big\} = P\Big\{A_i < B_i\Big\} \\ H_A\colon\ P\Big\{A_i > B_i\Big\} > P\Big\{A_i < B_i\Big\} \end{cases}$$

Воспользуемся критерием знаков.

Для этого сначала отбросим все наблюдения, в которых $A_i = B_i$. Во-первых, они бесполезны, так как разница в оценках нам не важна – важен только факт превосходства одной воды над другой. Во-вторых, они нарушают распределение n^+ и n^-.

[В методе get_data считываем данные из файла "data.txt", игнорируя наблюдения с одинаковыми оценками для воды A и воды B. Отсеянные наблюдения сохраняем в файл "data_without_equal_appraisals.txt".]

Удалив вышеописанные наблюдения, получаем:

$$\begin{cases} n^+ \overset{H_0}{\sim} Bi(n^+ + n^-;\, 0.5) \\ n^- \overset{H_0}{\sim} Bi(n^+ + n^-;\, 0.5) \end{cases}$$

В качестве статистики используем $T = n^-$.

[В методе calculate_n_plus_and_n_minus считаем n^+ и n^-, сохраняя их в файл "n_plus_and_n_minus.txt". Из файла получаем $n^+ = 26$ и $n^- = 14$.]

Критическая область:

$$T \leqslant T_{n^+ + n^-\,;\,\alpha} \;=\; max\Big\{k\colon\ P_{H_0}\{n^- \leqslant k\} \leqslant \alpha\Big\} \;=\; max\Big\{k\colon\ \sum_{i=0}^{k} C^i_{n^+ + n^-} \cdot 0.5^{n^+ + n^-} \leqslant \alpha\Big\}$$

[В методе calculate_critical_value вычисляем критическое значение для области, сохраняя его в файл "critical_value.txt". Из файла получаем $T_{n^+ + n^-\,;\,\alpha} = 15$.]

В таком случае, мы попадаем в критическую область, так как $T = 14 \leqslant 15 = T_{40\,;\,0.1}$. Значит, мы отвергаем основную гипотезу в пользу альтернативной, то есть есть основание считать, что предполагаемый потребитель предпочитает воду A воде B.

Вопрос 2

Введём обозначения:

- n – исходное количество наблюдений
- n_0 – количество наблюдений, где пол человека - мужской
- n_1 – количество наблюдений, где пол человека - женский
- sex_i – пол i-го человека
- A_i – оценка воды A i-м человеком
- $rank_i$ – ранг i-го наблюдения

Основная гипотеза - отношение к воде A не связано с полом (выборки по полу однородны). Альтернативная гипотеза - отношение к воде связано с полом (выборки неоднородны - двусторонняя альтернатива).

Воспользуемся критерием Уилкоксона.

Для этого сначала отбросим все оценки для воды B (так как мы анализируем не предпочтение между водой A и водой B в зависимости от пола, отношение к воде A в зависимости от пола), отсортируем наблюдения в порядке возрастания по оценке воды A и посчитаем ранги наблюдений.

[В методе "get_data" считываем данные из файла "data.txt", игнорируя оценки для воды B. В этом же методе происходит сортировка наблюдений по возрастанию оценки воды A и присвоение рангов этим наблюдениям. Полученные данные сохраняем в файл "ranked_data.txt".]

Для выбора статистики вычислим n_0 и n_1.

[В методе calculate_n0_and_n1 вычисляем n_0 и n_1, сохраняя их в файл "n0_and_n1.txt". Из файла получаем $n_0 = 20$ и $n_1 = 24$.]

Рассмотрим ранги наблюдений, где пол человека - мужской, и используем статистику Уилкоксона $W_{\text{набл}} = \sum_{i=0}^{n_0} sex_i = 0 \, rank_i$.

[В методе calculate_Wilcoxon_statistics вычисляем значение W-статистики, сохраняя его в файл "Wilcoxon_statistics.txt". Из файла получаем $W_{\text{набл}} = 318.5$.]

Критическая область:

$$\begin{bmatrix} W_{\text{набл}} \geqslant W(\frac{\alpha}{2}, n_1, n_0) \\ W_{\text{набл}} \leqslant n_0 \cdot (n_1 + n_0 + 1) - W(\frac{\alpha}{2}, n_1, n_0) \end{bmatrix} \Leftrightarrow \begin{bmatrix} W_{\text{набл}} \geqslant n_0 \cdot (n_1 + n_0 + 1) - W(\frac{\alpha}{2}, n_0, n_1) \\ W_{\text{набл}} \leqslant W(\frac{\alpha}{2}, n_0, n_1) \end{bmatrix}$$

Воспользовавшись таблицей критических значений критерия Уилкоксона, получаем $W(\frac{\alpha}{2}, n_0, n_1) = W(0.05, 20, 24) = 379$. Тогда, критическая область имеет вид:

$$\begin{bmatrix} W_{\text{набл}} \geqslant 521 \\ W_{\text{набл}} \leqslant 379 \end{bmatrix}$$

В таком случае, мы попадаем в критическую область, так как $W_{\text{набл}} = 318.5 \leqslant 379 = W(\frac{\alpha}{2}, n_0, n_1)$. Значит, мы отвергаем основную гипотезу в пользу альтернативной, то есть отношнение к воде A связано с полом потребителя.

Приложение

Код программы

question1.cpp

```cpp
#include <iostream>
#include <fstream>
#include <vector>

///Из файла get_path получает наблюдения, первые значения которых сохраняет
///в массив a (оценка воды A), вторые - в массив b (оценка воды B). Если внутри
///одного наблюдения оценке воды A соответствует с оценкой воды B, то наблюдение игнорируется.
///Выборка без наблюдений с равными оценками сохраняется в файл save_path.
void get_data(const std::string &get_path, const std::string &save_path,
              std::vector<int> &a, std::vector<int> &b) {
    //Если массивы были непусты, то очищаем их.
    a.erase(a.begin(), a.end());
    b.erase(b.begin(), b.end());

    std::ifstream fin;
    fin.open(get_path);
    std::ofstream fout;
    fout.open(save_path);

    //Переменные для считывания наблюдений.
    int cur_sex, cur_a, cur_b;
    while (fin >> cur_sex >> cur_a >> cur_b) {
        //Если оценки не равны, то добавляем данное наблюдение
        //(иначе - игнорируем).
        if (cur_a != cur_b) {
            a.push_back(cur_a);
            b.push_back(cur_b);
            fout << cur_sex << " " << cur_a << " " << cur_b << "\n";
        }
    }

    fin.close();
    fout.close();
}
```

Получение и редактирование данных

```cpp
///Вычисляет и сохраняет в файл path: количество наблюдений, в которых оценка воды A больше
///оценки воды B (n_plus); количество наблюдений, в которых оценка воды B больше
///оценки воды A (n_minus); суммарное количество наблюдений (n_plus + n_minus).
///Если встречается наблюдение с одинаковыми оценками, выбрасывается std::logic_error.
///Возвращает массив, первый элемент которого - n_plus, второй - n_minus.
int *calculate_n_plus_and_n_minus(const std::string &path, const std::vector<int> &a,
                                  const std::vector<int> &b) {
    std::ofstream fout;
    fout.open(path);

    int n_plus = 0, n_minus = 0;
    for (int i = 0; i < a.size(); i++) {
        if (a[i] > b[i]) {
            n_plus++;
        } else if (a[i] < b[i]) {
            n_minus++;
        } else {
            throw std::logic_error("There should be no equal appraisals. ");
        }
    }
    fout << "n^+ = " << n_plus << "\nn^- = " << n_minus << "\n" <<
         "n + n^+ + n^- = " << n_plus + n_minus;

    fout.close();
    int *n = new int[2];
    n[0] = n_plus;
    n[1] = n_minus;
    return n;
}
```

Вычисление n^+ и n^-

```cpp
///Вычисляет и возвращает число сочетаний из n по k.
long long calculate_number_of_combinations(int k, int n) {
    if (k == 0) {
        return n;
    }
    long long res = n - k + 1;
    for (int i = n - k + 2; i <= n; i++) {
        res *= i;
        res /= (i - n + k);
    }
    return res;
}
```

Вычисление C_n^k

```cpp

///Вычисляет, сохраняет в файл path и возвращает критическое значение для n наблюдение
///на уровне значимости alpha.
int calculate_critical_value(const std::string &path, const int n, const double alpha) {
    std::ofstream fout;
    fout.open(path);

    //Так как во всех слагаемых присутствует постоянный коэффициент 0.5^n,
    //его можно вынести. Поделив обе части неравенства на этот коэффициент,
    //в правой части получаем alpha / 0.5^n = 2^n * alpha.
    long long helper = 1;
    for (int i = 0; i < n; i++) {
        helper *= 2;
    }
    //В таком случае, в левой части останутся только целые числа. Тогда, не имеет
    //смысла хранить нецелое число - округлив в меньшую сторону, мы не повлияем
    //на ответ.
    helper = static_cast<long long>(static_cast<double>(helper) * alpha);

    int k;
    long long p = 0;
    for (k = -1; p + calculate_number_of_combinations( k: k + 1, n) <= helper; k++) {
        p += calculate_number_of_combinations( k: k + 1, n);
    }
    fout << "T_{n; alpha} ";
    if (k == -1) {
        fout << " does not exist";
    } else {
        fout << "= " << k;
    }

    fout.close();
    return k;
}
```

Вычисление критического значения

```cpp
int main() {
    //Массивы для хранения наблюдение. a - оценка воды A; b - оценка воды B. Пол в данном
    //вопросе не фигурирует, поэтому мы не будем сохранять его программно.
    std::vector<int> a, b;

    get_data( get_path: "data.txt",  save_path: "data_without_equal_appraisals.txt",  &: a,  &: b);
    int *n = calculate_n_plus_and_n_minus( path: "n_plus_and_n_minus.txt", a, b);
    calculate_critical_value( path: "calculate_critical_value.txt",  n: n[0] + n[1],  alpha: 0.1);

    delete[] n;
    return 0;
}
```

Запуск всех операций

question2.cpp

```cpp
#include <iostream>
#include <fstream>
#include <algorithm>
#include <vector>

///Компаратор для сортировки по возрастанию оценки воды A.
bool cmp(std::pair<std::pair<int, int>, int> i,
         std::pair<std::pair<int, int>, int> j) {
    return i.first.second < j.first.second;
}
```

Метод сравнения наблюдений (для сортировки)

```cpp
///На файле get_path получает наблюдения. Значения оценки для воды B генерируется.
///Значения пола и оценок для воды A сохраняется в массив sex_a_rank в отсортированном
///по оценке воды A порядке (нулевой элемент - пол, первый элемент - оценка для воды A,
///второй элемент - ранг наблюдения). Полученные данные сохраняет в файл save_path.
void get_data(const std::string &get_path, const std::string &save_path,
              std::vector<std::pair<std::pair<int, int>, double>> &sex_a_rank) {
    //Если массив был непуст, то очищаем его.
    sex_a_rank.erase(sex_a_rank.begin(), sex_a_rank.end());

    std::ifstream fin;
    fin.open(get_path);
    std::ofstream fout;
    fout.open(save_path);

    //Переменные для считывания наблюдений.
    int cur_sex, cur_a, cur_b;
    while (fin >> cur_sex >> cur_a >> cur_b) {
        sex_a_rank.emplace_back( x: std::pair<int, int>(cur_sex, cur_a),  y: 0);
    }
    //Сортировка по возрастанию оценки воды A.
    sort(sex_a_rank.begin(), sex_a_rank.end(), cmp);
    //Заполнение ранга.
    for (int i = 0, i, in_a_row = 1; i < sex_a_rank.size(); i++) {
        if (i == sex_a_rank.size() - 1 ||
            sex_a_rank[i].first.second != sex_a_rank[i + 1].first.second) {
            double helper = i + 1 - (in_a_row - 1) / 2.0;
            for (j = i - in_a_row + 1; j < i + 1; j++) {
                sex_a_rank[j].second = helper;
            }
            in_a_row = 1;
        } else {
            in_a_row++;
        }
    }
    for (auto &i : sex_a_rank) {
        fout << i.first.first << " " << i.first.second << " " << i.second << "\n";
    }

    fin.close();
    fout.close();
}
```

Получение и редактирование данных

```cpp
///Вычисляет и сохраняет в файл path: количество наблюдений, в которых пол
///человека - мужской (n0); количество наблюдений, в которых пол человека -
///женский (n1); суммарное количество наблюдений (n0 + n1).
///Возвращает массив, первый элемент которого - n0, второй - n1.
int *calculate_n0_and_n1(const std::string &path,
                         const std::vector<std::pair<std::pair<int, int>, double>> &sex_a_rank) {
    std::ofstream fout;
    fout.open(path);

    int n0 = 0, n1 = 0;
    for (auto &i : sex_a_rank) {
        if (i.first.first) {
            n1++;
        } else {
            n0++;
        }
    }
    fout << "n0 = " << n0 << "\nn1 = " << n1 << "\nn = n0 + n1 = " << n0 + n1;

    int *n = new int[2];
    n[0] = n0;
    n[1] = n1;
    return n;
}
```

Вычисление n_0 и n_1

```cpp
///Вычисляет, сохраняет в файл path и возвращает значение W-статистики для набора
///наблюдений sex_a_rank.
double calculate_Wilcoxon_statistics(const std::string &path,
                                     const std::vector<
                                             std::pair<std::pair<int, int>, double>> &sex_a_rank) {
    std::ofstream fout;
    fout.open(path);

    double res = 0;
    //Во избежание проверки пола можно домножить на (1 - пол): так как женский пол
    //закодирован нулём, а мужской единицей, то при домножении каждого ранга на
    //(1 - пол) получаем сумму рангов наблюдений, где пол человека - мужской, то есть
    //получаем искомую статистику.
    for (auto &i : sex_a_rank) {
        res += (1 - i.first.first) * i.second;
    }
    fout << "W = " << res;

    fout.close();
    return res;
}
```

Вычисление W-статистики

```cpp
int main() {
    std::vector<std::pair<std::pair<int, int>, double>> sex_a_rank;

    get_data( get_path: "data.txt",  save_path: "ranked_data.txt",  &: sex_a_rank);
    delete[] calculate_n0_and_n1( path: "n0_and_n1.txt", sex_a_rank);
    calculate_Wilcoxon_statistics( path: "Wilcoxon_statistics.txt", sex_a_rank);

    return 0;
}
```

Запуск всех операций

Названия и содержимое файлов, связанных с программой

1) "question1.txt" – исходный код программы для 1-го вопроса
2) "question2.txt" – исходный код программы для 2-го вопроса
3) "data.txt" – входной файл с наблюдениями (каждое наблюдение – строка из 3-х чисел, разделённых пробелами: 1-е число – пол человека (0 или 1); 2-е число – оценка воды A (от 1 до 7 включительно); 3-е число – оценка воды B (от 1 до 7 включительно))
4) "data_without_equal_appraisals.txt" (1-й вопрос) – файл с наблюдениями, в которых оценка для воды A не равна оценке для воды B
5) "n_plus_and_n_minus.txt" (1-й вопрос) – файл, содержащий: количество наблюдений, к которых оценка воды A больше оценки воды B; количество наблюдений, к которых оценка воды B больше оценки воды A; суммарное количество наблюдений
6) "critical_value.txt" (1-й вопрос) – файл, содержащий критическое значение статистики
7) "ranked_data.txt" (1-й вопрос) (2-й вопрос) – файл содержащий наблюдения в виде строки из 3-х чисел: 1-е – пол человека; 2-е – оценка воды A; 3-е – ранг наблюдения
8) "n0_and_n1.txt" (2-й вопрос) – файл, содержащий: количество наблюдений, в которых пол человека - мужской; количество наблюдений, в которых пол человека - женский; суммарное количество наблюдений
9) "Wilcoxon_statistics.txt" (2-й вопрос) – файл, содержащий значение W-статистики