Задача 2
Вариант 1

от Татаринова Никиты Алексеевича

к 21.03.2021

Условие

Датчики псевдослучайных чисел разрабатываются так, чтобы генерируемые ими последовательности можно было считать реализациями независимых случайных величин, равномерно распределённых на единичном отрезке. Ваше задание - реализовать такой датчик и проверить генерируемую последовательность на равномерность и независимость. Выполните следующие шаги:

1. Рассчитайте 100 псевдослучайных чисел методом, соответстующим вашему варианту, - методом средних квадратов ($z_1 = 1661$).
2. Приведите первые 10 чисел этой последовательности.
3. Постройте гистограмму с 10 столбцами для полученной последовательности.
4. Проверьте гипотезу о том, что последовательность имеет распределение $R(0;1)$, критерием хи-квадрат, разбив интервал $[0;1)$ на 10 равных интервалов.
5. Повторите шаги 3 и 4 для последовательности длиной в 10000 чисел.
6. Изучите тест перестановок и проверьте этим тестом первые 9999 вашей последовательности, разбив их на тройки.

Используйте уровень значимости 5%.

Решение

Шаг 1

Для вычисления чисел используем язык программирования C++ (исходный код в файле "random_sample_test.cpp", скриншоты кода в приложении). Используется метод generate_random_sample, в качестве параметров которому передаётся первый элемент, количесво элементов и имя файла, в который будут выводиться элементы - в данном случае, "sample100.txt".

Шаг 2

Из файла "sample100.txt" берём первые 10 элементов, являющихся искомыми:
0.1661; 0.7589; 0.5929; 0.1530; 0.3409; 0.6212; 0.5889; 0.6803; 0.2808; 0.8848

Шаг 3

Используя метод interval_distribution, в который передаются имя файла с числами и имя файла для вывода количеств чисел в каждом из интервалов, получаем файл с количествами чисел в каждом из интервалов - в данном случае, "interval_distribution100.txt". Тогда, вероятность вхождения в интервал будет равна количеству чисел в нём, делённое на 100 (суммарное количество чисел). Высота столбца будет равна полученной вероятности, поделённой на длину интервала.

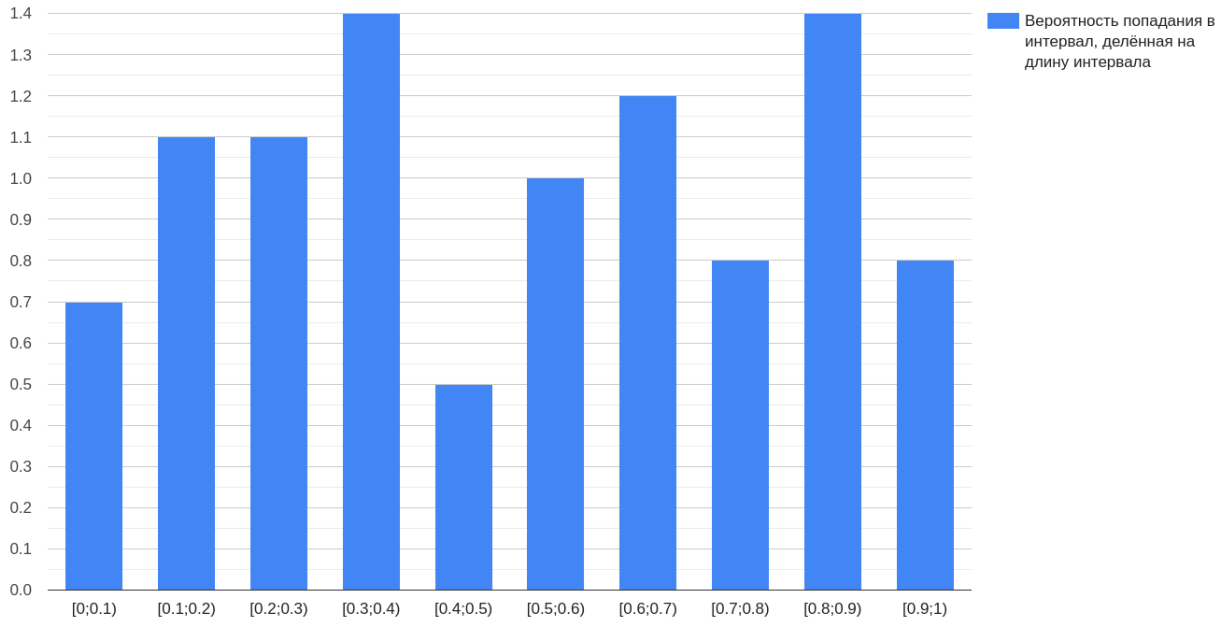

Рис 2.3.1: Гистограмма распрделения 100 чисел по интервалам

Шаг 4

H_0 - последовательность имеет распределение $R(0;1)$, H_A - распределение отличается от $R(0;1)$. Тогда, на основании критерия хи-квадрат выбираем следующую статистику:

$$\chi^2 = \sum_{j=1}^{k} \frac{\left(O_j - E_j\right)^2}{E_j} \overset{H_0}{\sim} \chi^2_{k-1}$$

, где O_j - наблюдаемые частоты; E_j - ожидаемые частоты. В данном случае, $k = 10$, $E_j = 10 \quad \forall j = \overline{1,k}$, а O_j хранятся в файле "interval_distribution100.txt". Тогда, получаем $\chi^2 = 8 < 16.919 = \chi^2_{9;0.05}$.

В таком случае, на уровне значимости 5% гипотеза о том, что последовательность имеет распределение $R(0;1)$, не опровергается.

Шаг 5

В этом пункте получаем 10000 чисел вместо 100. Сохраним числа в файл "sample10000.txt", количества чисел в каждом из интервалов в файл "interval_distribution10000.txt".

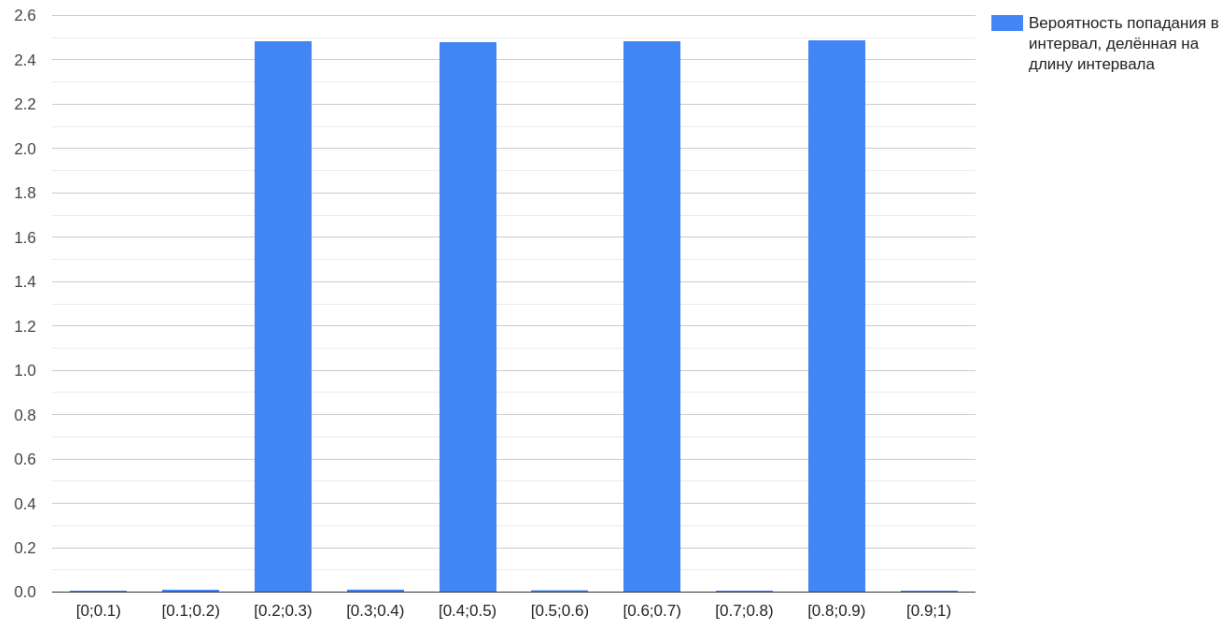

Рис 2.5.1: Гистограмма распрделения 10000 чисел по интервалам

Основная и альтернативная гипотезы остаются неизменными, как и статистика. Изменились только наблюдаемые частоты O_j. Тогда, получаем $\chi^2 = 14711.48 > 16.919 = \chi^2_{9;0.05}$. В таком случае, на уровне значимости 5% гипотеза о том, что последовательность имеет распределение $R(0;1)$, опровергается.

Шаг 6

Для получения троек используем метод "permutation_distribution", в который передаются имя файла с числами и имя файла для вывода количеств каждого из видов троек - в данном случае, "permutation_distribution.txt".

H_0 - элементы последовательности независимы (что возможно при равновероятном появлении троек), H_A - элементы последовательности зависимы. Тогда, на основании критерия хи-квадрат выбираем следующую статистику:

$$\chi^2 = \sum_{j=1}^{k} \frac{(O_j - E_j)^2}{E_j} \overset{H_0}{\sim} \chi^2_{k-1}$$

, где O_j - наблюдаемые частоты; E_j - ожидаемые частоты. В данном случае, $k=6$, $E_j = \frac{3333}{6} = 555.5 \quad \forall j = \overline{1,k}$, O_j хранятся в файле "permutation_distribution.txt". Тогда, получаем

$\chi^2 = \frac{910817.5}{555.5} \approx 1639.63546 > 11.070 = \chi^2_{5;0.05}$.

В таком случае, на уровне значимости 5% гипотеза о том, что элементы последовательности независимы, отвергается.

Приложение

Код программы

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

void generate_random_sample(const int first_num, const int size, const char *path) {
    if (first_num < 0 || first_num > 9999) {
        throw std::logic_error(
                "Все элементы выборки должны лежать в диапазоне [0; 9999].");
    }
    std::ofstream fout;
    fout.open(path);
    for (int i = 0, cur_num = first_num; i < size; i++) {
        fout << "0." << cur_num / 1000 << (cur_num % 1000) / 100 <<
             (cur_num % 100) / 10 << cur_num % 10 << "\n";
        cur_num *= cur_num;
        cur_num /= 100;
        cur_num %= 10000;
    }
    fout.close();
}
```

1

```cpp
void interval_distribution(const char *num_path, const char *distribution_path) {
    int *intervals = new int[10];
    for (int i = 0; i < 10; i++) {
        intervals[i] = 0;
    }

    std::ifstream fin;
    fin.open(num_path);
    double num;
    while (fin >> num) {
        intervals[static_cast<int>(num * 10)]++;
    }
    fin.close();

    std::ofstream fout;
    fout.open(distribution_path);
    for (int i = 0; i < 10; i++) {
        fout << intervals[i] << "\n";
    }
    fout.close();

    delete[] intervals;
}
```

2

```cpp
void permutation_distribution(const char *num_path,
                              const char *permutation_distribution_path) {
    int *permutations = new int[6];
    for (int i = 0; i < 6; i++) {
        permutations[i] = 0;
    }

    std::ifstream fin;
    fin.open(num_path);
    double num1, num2, num3;
    while ((fin >> num1) && (fin >> num2) && (fin >> num3)) {
        if (num1 <= num2 && num2 <= num3) {
            permutations[0]++;
        } else if (num1 <= num3 && num3 <= num2) {
            permutations[1]++;
        } else if (num2 <= num1 && num1 <= num3) {
            permutations[2]++;
        } else if (num2 <= num3 && num3 <= num1) {
            permutations[3]++;
        } else if (num3 <= num1 && num1 <= num2) {
            permutations[4]++;
        } else {
            permutations[5]++;
        }
    }
    fin.close();

    std::ofstream fout;
    fout.open(permutation_distribution_path);
    for (int i = 0; i < 6; i++) {
        fout << permutations[i] << "\n";
    }
    fout.close();
    delete[] permutations;
}
```

3

```cpp
int main() {
    generate_random_sample( first_num: 1661,  size: 100,  path: "sample100.txt");
    interval_distribution( num_path: "sample100.txt",  distribution_path: "interval_distribution100.txt");
    generate_random_sample( first_num: 1661,  size: 10000,  path: "sample10000.txt");
    interval_distribution( num_path: "sample10000.txt",  distribution_path: "interval_distribution10000.txt");
    permutation_distribution( num_path: "sample10000.txt",
                              permutation_distribution_path: "permutation_distribution.txt");
    return 0;
}
```

4

Названия и содержимое файлов, полученных в результате работы программы

1) "sample100.txt" - содержит первые 100 чисел последовательности
2) "sample10000.txt" - содержит первые 10000 чисел последовательности
3) "interval_distribution100.txt" - содержит разбиение первых 100 чисел последовательности по интервалам $\left[{}^{i}/_{10}; {}^{(i+1)}/_{10}\right) \quad i = \overline{0,9}$
4) "interval_distribution10000.txt" - содержит разбиение первых 10000 чисел последовательности по интервалам $\left[{}^{i}/_{10}; {}^{(i+1)}/_{10}\right) \quad i = \overline{0,9}$
5) "permutation_distribution.txt содержит разбиение троек чисел последовательности по перестановкам их рангов