Министерство образования и науки Российской Федерации

ФГБОУ ВО Рыбинский государственный авиационный технический университет имени П.А. Соловьева

Факультет радиоэлектроники и информатики

Кафедра математического и программного обеспечения электронных вычислительных средств

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

по дисциплине

Математические методы анализа данных

по теме

Студент группы ИПБ-13 Иванов Р.А.

Преподаватель, доцент Воробьев К. А.

Рыбинск 2017

Содержание

1. Гистограммы

Ниже представлена гистограмма для выборки, каждый элемент которой представляет среднее арифметическое выборки с равномерным распределением объёмом 3 (рис. 1). Распределение имеет нормальный вид.

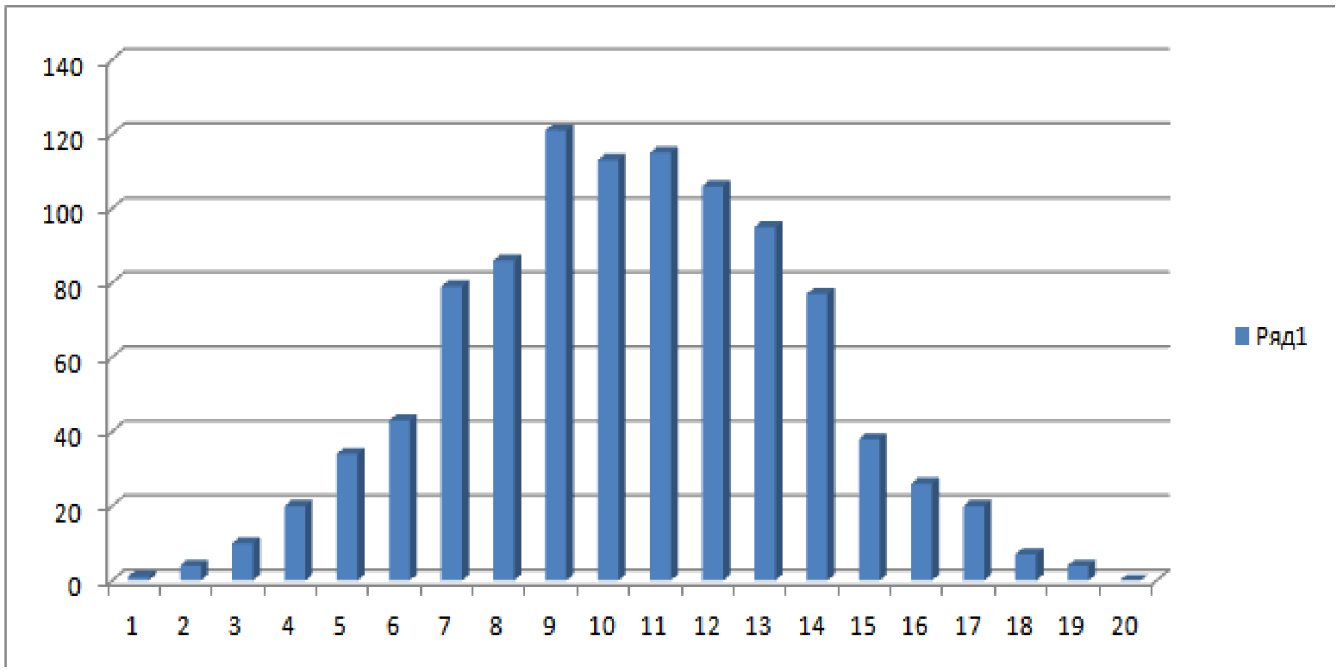

Рис. 1 – Гистограмма для выборки из 3 элементов сложения

Хотя на графики видно, что распределение имеет нормальный вид, но нам бы хотелось более ярко выраженный "колокол".

При выборке из 20 элементов сложения (рис. 2) видно, что среднее квадратичное отклонение стало меньше.

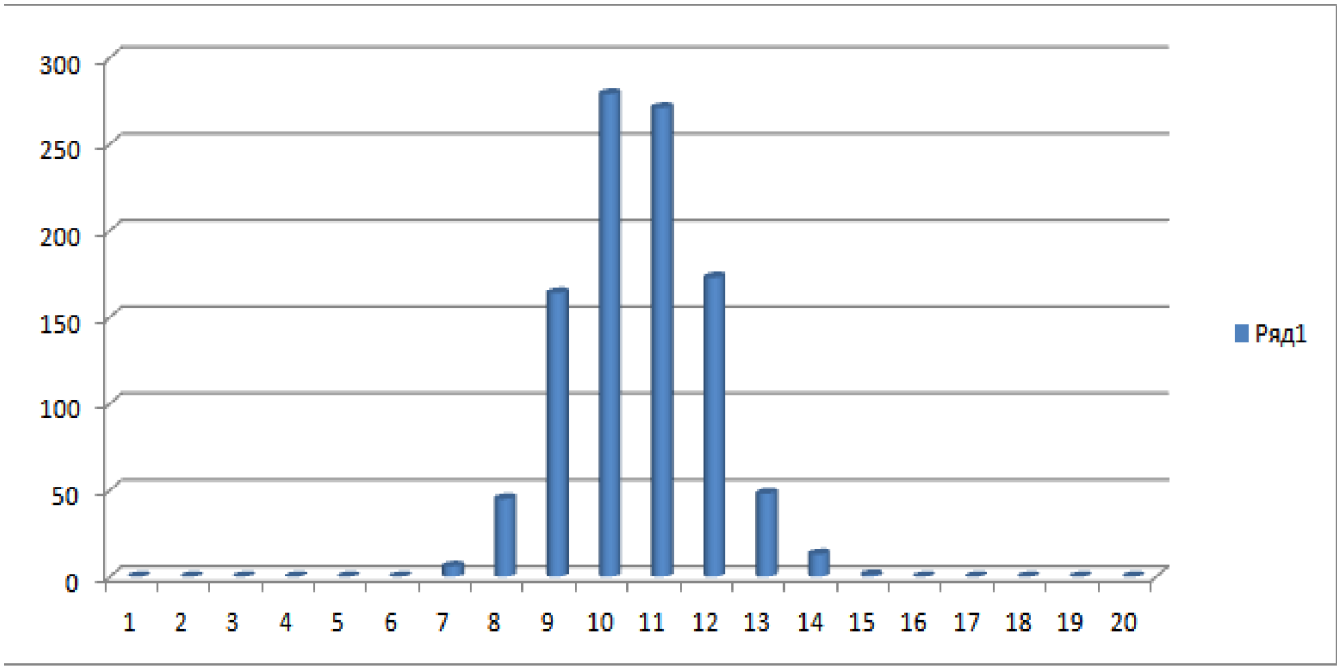

Рис. 2 – Гистограмма для выборки из 20 элементов сложения

"Колокол"графика стал уже, но все еще оставляет желать лучшего, т. к. мы можем на нем видеть, что некторые столбцы чуть выше или чуть ниже. А значит нам нужно, по следствию из центральной предельной теоремы, увеличить число слагаемых.

При выборке из 10 элементов сложения (рис. 3) видно, что "колокол"стал ещё выше и уже. Из центральной предельной теоремы следует, что среднее квадратичное отклонение этого распределения должно быть в 2 раза меньше, чем в предыдущего, что и видно на гистограмме.

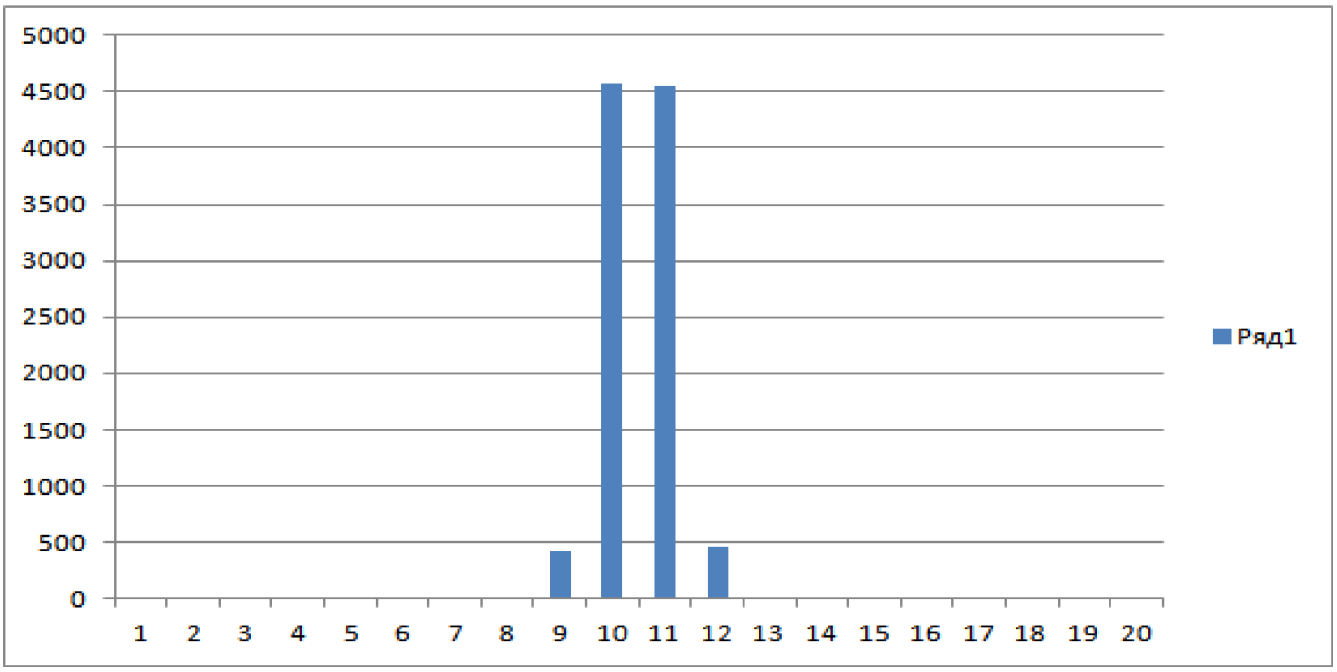

Рис. 3 – Гистограмма для выборки из 100 элементов сложения

2. Коэффициенты ковариации

Ковариация – мера линейной зависимости двух случайных величин.

Знак коэффицента ковариации указывает на вид линейной связи между рассматриваемыми величинами: если он > 0 - это означает прямую связь (при росте одной величины растет и другая), < 0 указывает на обратную связь. При коэффициенте $= 0$ линейная связь между выборками при нулевом сдвиге отсутствует.

$$cov(X, Y) = \frac{1}{n}\sum_{i=1}^{n} x_i y_i - \left(\frac{1}{n}\sum_{i=1}^{n} x_i\right)\left(\frac{1}{n}\sum_{i=1}^{n} y_i\right)$$

Ниже представлена ковариацилнная матрица для выборки из 100 элементов сложения.

$$\begin{bmatrix}
7.912E-4 & 3.47E-5 & -2.69E-5 & -8.33E-5 & 1.5E-6 & -6.0E-5 & -1.82E-5 & 5.4E-5 & 2.29E-5 & 2.26E-5 \\
3.47E-5 & 7.808E-4 & -7.68E-5 & 3.29E-5 & 1.13E-5 & 4.76E-5 & 5.98E-5 & -2.8E-6 & -3.39E-5 & -2.81E-5 \\
-2.69E-5 & -7.68E-5 & 9.15E-4 & 7.1E-6 & -1.94E-5 & -1.15E-5 & -4.89E-5 & 5.07E-5 & 2.68E-5 & -4.6E-6 \\
-8.33E-5 & 3.29E-5 & 7.1E-6 & 8.472E-4 & 1.44E-5 & -3.52E-5 & -2.38E-5 & -1.1E-5 & -2.93E-5 & -9.9E-6 \\
1.5E-6 & 1.13E-5 & -1.94E-5 & 1.44E-5 & 7.754E-4 & -6.49E-5 & 2.88E-5 & 5.0E-6 & -8.6E-6 & -6.9E-6 \\
-6.0E-5 & 4.76E-5 & -1.15E-5 & -3.52E-5 & -6.49E-5 & 9.101E-4 & 4.35E-5 & -1.17E-5 & 4.62E-5 & 2.68E-5 \\
-1.82E-5 & 5.98E-5 & -4.89E-5 & -2.38E-5 & 2.88E-5 & 4.35E-5 & 9.212E-4 & -1.79E-5 & 4.99E-5 & 2.55E-5 \\
5.4E-5 & -2.8E-6 & 5.07E-5 & -1.1E-5 & 5.0E-6 & -1.17E-5 & -1.79E-5 & 8.448E-4 & 6.22E-5 & -4.34E-5 \\
2.29E-5 & -3.39E-5 & 2.68E-5 & -2.93E-5 & -8.6E-6 & 4.62E-5 & 4.99E-5 & 6.22E-5 & 8.386E-4 & 4.3E-6 \\
2.26E-5 & -2.81E-5 & -4.6E-6 & -9.9E-6 & -6.9E-6 & 2.68E-5 & 2.55E-5 & -4.34E-5 & 4.3E-6 & 8.227E-4
\end{bmatrix}$$

Т.к. мы можем увидеть, что каждый элемент матрицы приближается к 0 из этого делаем вывод, что переменные линейно независимы.

3. Выборочные моменты

В данном случае можно сказать, что начальным моментом k-го порядка случайной величины X называется среднее арифметическое k-ой степени этой случайной величины.

$$\alpha_k(X) = \sum_{i=1}^{n} x_i^k p_i$$

Центральным моментом k-го порядка случайной величины X называется начальный момент k-ой степени отклонения случайной величины X от её мат. ожидания.

$$\mu_k(X) = \alpha_k \left((X - M(X))^k \right)$$

Ввиду того, что все элементы выборки нормированны, теоретическое значение мат. ожидания составляет 0.5. Теоретическая дисперсия для каждой выборки исходя из свойств равномерно распределенной С.В. и центральной предельной теоремы составляет 0.02778, 0.0041667, 0.0008334 для выборок из 3, 20, 100 элементов сложения соответственно. Ниже представлены рассчетные значения моментов для каждой выборки (таблицы 1-3). Можно заметить, что реальные значения дисперсии и мат. ожиданий близки к теоретическим значениям.

Таблица 1 – Вычисленные моменты для выборки из 3 элементов сложения

Порядок момента	Начальный момент	Центральный момент
1	0.49638439340750873	4.951594689828198e-17
2	0.2722292309595661	0.0258317649410256
3	0.16079444249574035	1.913087629077591e-05
4	0.10078517219460696	0.0018461873150255987

Таблица 2 – Вычисленные моменты для выборки из 20 элементов сложения

Порядок момента	Начальный момент	Центральный момент
1	0.5017406913785922	2.2204460492503132e-17
2	0.255828559317913	0.00408483793284526
3	0.1324864894567112	2.783241296176622e-05
4	0.06964790298961453	4.71493015071018e-05

Так же мы можем заметить, что 3 центральный момент близок к 0, из-за чего мы можем утверждать, что относительно мат. ожидание наши выборки симметричны.

Таблица 3 – Вычисленные моменты для выборки из 100 элементов сложения

Порядок момента	Начальный момент	Центральный момент
1	0.4992792925566728	9.270362255620058e-18
2	0.2500710439934647	0.000791232017573033
3	0.1256477706088138	2.3851508840121784e-06
4	0.0633305312395699	1.9141436304834823e-06

4. Доверительный интервал

Доверительный интервал (d) уровня 0.95 определяет интервал, в который с вероятностью 95% попадет мат. ожидание (α_1) элементов выборки.

$$d = t\sqrt{\frac{m_2}{n}}$$

Где t в данном случае равно 1.96.

Во всех случая мат. ожидания находятся в пределах доверительных интервалов (таблица 4).

Таблица 4 – Вычисленные значения доверительных интервалов и мат. ожиданий

Кол-во сложений	0.5 - d	мат. ожидание(a_1)	0.5 + d
3	0.490038308	0.49638439340750873	0.509961692
20	0.496038648	0.5017406913785922	0.503961352
100	4.98e-01	0.4992792925566728	5,02e-01

5. Общая оценка качества генератора

При достаточно большом размерер выборки(эмперически установлено, что лучше брать выборки не меньше 10000 отсчётов), рассмотренный генератор случайных чисел на основе линейного конгруэнтного метода даёт распределение очень близкое к нормальному с характерестиками близкими к тем, которые следуют из центральной предельной теоремы. Мат. ожидания находятся в пределах доверительных интевалов с вероятностью 95%. Достаточно низкий коэффициент ковариации говорит об отсутствии линейной зависимости выборок с при нулевом сдвиге.

6. Приложение

Генератор случайных чисел из прошлой работы

```java

package ravnrasp;


/**
 * Клвсс для генерации рандомного числа
 * Created by Roman on 03.01.2017.
 */
public class Random {

    public static double a = 8121;
    public static double c = 28411;
    public static int m = 134456;
    public static double seed = 2;

    /**
     * Метод генерирует рандомное число от 0 до 1
     */
    public static double getRand() {
        seed = (a * seed + c) % m;
        return seed / (double) m;
    }

    /**
     * Генерирует рандомное число в диапазоне от a до b
     */
    public static double getRand(int a, int b) {
        return (double) a + ((double) b - (double) a) * getRand();
    }
}
```

Генератор случайных чисел

```java

package ravnrasp;

import java.util.ArrayList;

/**
 *
 * @author Roman
 */
public class RavnRandom {

    public static double ravnRandom(int k) {
        ArrayList<Double> array = new ArrayList<>();
        for (int i = 0; i < k; i++) {
            array.add(Random.getRand());
        }
        double result = array.stream()
                .reduce((s1, s2) -> s1 + s2)
                .orElse(0d);
```

```
20
21          return result / k;
22      }
23  }
```

Рассчет и отображение параметров распределения

```
1
2   package ravnrasp;
3
4   import java.util.ArrayList;
5   import java.util.List;
6
7   import static java.util.stream.Collectors.toList;
8
9   /**
10   * Класс для вспомогательных вычислений
11   *
12   * @author Roman
13   */
14  public class CalculationHelper {
15
16      /**
17       * Расчитывает математическое ожидание случайной велечины
18       *
19       * @param list список случайных велечин
20       * @return математическое ожидание
21       */
22      public static double calculateExpectedValue(List<Double> list) {
23          double sum = list.stream()
24                  .mapToDouble(s -> s)
25                  .sum();
26          return sum / list.size();
27      }
28
29      /**
30       * Расчитать начальный момент случайной велечины
31       *
32       * @param list список случайныъ велечин
33       * @param k степень
34       * @return начальный момент
35       */
36      public static double calculateInitialMoment(List<Double> list, int k) {
37          return calculateExpectedValue(list.stream()
38                  .map(s -> Math.pow(s, k))
39                  .collect(toList())
40          );
41      }
42
43      /**
44       * Метод для подсчета матрицы коварриации
45       * @param arrays
46       * @param n
47       * @return
48       */
49      public static double[][]
```

```
    calculateCovarMatr(ArrayList<ArrayList<Double>> arrays,
     int n) {
        double[][] result = new double[n][n];
        for (int i = 0; i < n; i++) {
            List<Double> x = arrays.get(i);
            for (int j = 0; j < n; j++) {
                List<Double> y = arrays.get(j);
                ArrayList<Double> array = new ArrayList<>();
                for (int k = 0; k < arrays.get(i).size(); k++) {
                    array.add(x.get(k) * y.get(k));
                }
                result[i][j] = calculateExpectedValue(array)
                        - (calculateExpectedValue(x) *
                         calculateExpectedValue(y));
            }
        }
        return result;
    }

    /**
     * Расчитать центральный момент случайной велечины
     *
     * @param list список случайных велечин
     * @param k степень
     * @return центральный момент
     */
    public static double calculateCentralMoment(List<Double>
    list, int k) {
        double expValue = calculateExpectedValue(list);
        return calculateExpectedValue(list.stream()
                .map(s -> Math.pow(s - expValue, k))
                .collect(toList())
        );
    }

    /**
     * Расчитать выборочную дисперсию случайной велечины
     *
     * @param list список случайных велечин
     * @return дисперсия
     */
    public static double calculateDispersion(List<Double> list) {
        return calculateCentralMoment(list, 2);
    }

    /**
     * Посчитать коэффициент автокорелляции
     *
     * @param list список случайных велечин
     * @param k шаг
     * @return коэффициент автокорелляции
     */
    public static double calculateAutocorrelation(List<Double> list,
    int k) {
        List<Double> forCalculationList
```

```
                = list.stream()
                .map(s -> s -= 0.5)
                .collect(toList());
        // количество пар использованых для расчета
        int manyPairs = 0;Random.getRand()
        List<Double> resultList = new ArrayList<>();
        for (int i = 0; i < forCalculationList.size(); i++) {
            double mult = forCalculationList.get(i);
            if (i + k < forCalculationList.size()) {
                mult *= forCalculationList.get(i + k);
                manyPairs++;
                resultList.add(mult);
            }
        }

        for (int i = 0; i < resultList.size(); i++) {
            double res = resultList.get(i);
            resultList.set(i, res / manyPairs);
        }

        double sum = resultList.stream()
                .mapToDouble(s -> s)
                .sum();

        return sum / calculateDispersion(list);
    }

    public static double calculateConfidenceInterval(List<Double>
    list, Double t) {
        return t * Math.sqrt(calculateDispersion(list) / list.size());
    }
}
```