

Федеральное государственное бюджетное образовательное учреждение высшего
образования «Сибирский государственный университет
телекоммуникаций и информатики»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №0

Выполнил:
Студент гр. ИВ-622
Бубнов С.Ю.

Проверила:
Ассистент Кафедры ВС
Петухова Я.В.

Новосибирск 2020

СОДЕРЖАНИЕ

Постановка задачи	2
Теоретические сведения	2
Ход работы	5
Заключение	7
Листинг программы	8

ПОСТАНОВКА ЗАДАЧИ

Необходимо взять готовую реализацию трех генераторов псевдослучайных чисел на и убедиться в их равномерном распределении, используя такие параметры как критерий Пирсона и автокорреляция.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Пусть X - исследуемая случайная величина. Требуется проверить гипотезу H_0 о том, что данная случайная величина подчиняется закону распределения $F(x)$. Для этого необходимо произвести выборку из n независимых наблюдений над случайной величиной X : $X^n = (x_1, \dots, x_n)$, $x_i \in [a, b]$, $\forall_i = 1 \dots n$, и по ней построить эмпирический закон распределения $F'(x)$ случайной величины X .

Гипотеза H'_0 : X^n порождается ранее упомянутой функцией $F'(x)$. Разделим $[a, b]$ на k непересекающихся интервалов $[a_i, b_i]$, $i = 1 \dots k$.

Пусть n_j — количество наблюдений в j -м интервале;

$p_j = F(b_j) - F(a_j)$ - вероятность попадания наблюдения в j -й интервал при выполнении гипотезы H'_0 ;

$E_j = np_j$ - ожидаемое число попаданий в j -й интервал; тогда распределение Хи-квадрат с числом степеней свободы $k-1$ будет иметь следующую статистику:

$$\chi^2 = \sum_{i=1}^k \frac{(n_i - p_i N)^2}{p_i N} \sim \chi_{k-1}^2$$

В зависимости от значения критерия χ^2 гипотеза H_0 может приниматься либо отвергаться:

- $\chi_1^2 < \chi^2 < \chi_2^2$ - гипотеза H_0 выполняется.
- $\chi^2 \leq \chi_1^2$ — попадает в левый «хвост» распределения. Следовательно, теоретические и практические значения очень близки. Если, к примеру, происходит проверка генератора случайных чисел, который сгенерировал n чисел из отрезка $[0,1]$ и выборка X^n распределена равномерно на $[0,1]$, то генератор нельзя назвать случайным (гипотеза случайности не выполняется), т.к. выборка распределена слишком равномерно, но гипотеза H_0 выполняется.
- $\chi^2 \geq \chi_2^2$ — попадает в правый «хвост» распределения, гипотеза H_0 отвергается.

Автокорреляция уровней ряда – корреляционная между последовательными уровнями одного и того же ряда динамики (сдвинутыми на определенный промежуток времени L – лаг). Автокорреляция может быть измерена коэффициентом автокорреляции.

Лаг определяет порядок коэффициента автокорреляции. Если $L = 1$, то имеем коэффициент автокорреляции 1-го порядка $r_{t,t-1}$, если $L = 2$, то коэффициент автокорреляции 2-го порядка $r_{t,t-2}$ и т.д.

Рассчитав несколько коэффициентов автокорреляции, можно определить лаг (I), при котором автокорреляция ($r_{t,t-L}$) наиболее высокая, выявив тем самым структуру временного ряда. Если наиболее высоким оказывается значение $r_{t,t-1}$, то исследуемый ряд содержит только тенденцию. Если наиболее высоким оказался $r_{t,t-L}$, то ряд содержит (помимо тенденции) колебания периодом L . Если ни один из ($l=1;L$) не является значимым, можно сделать одно из двух предположений:

- либо ряд не содержит тенденции и циклических колебаний, а его уровень определяется только случайной компонентой;

- либо ряд содержит сильную нелинейную тенденцию, для выявления которой нужно провести дополнительный анализ.

Значение (по модулю)	Интерпретация
до 0,2	очень слабая корреляция
до 0,5	слабая корреляция
до 0,7	средняя корреляция
до 0,9	высокая корреляция
свыше 0,9	очень высокая корреляция

Автокорреляционная функция – предназначена для оценки корреляции между смещенными копиями последовательностей.

$$a(\tau) = \frac{\sum_{i=1}^{N-\tau} (x_i - Ex)(x_{i+\tau} - Ex)}{(N - \tau) * S^2(x)}, \text{ где}$$

Ex - математическое ожидание — среднее значение случайной величины при стремлении количества выборок или количества её измерений (иногда говорят — количества испытаний) к бесконечности:

$$E(x) = \frac{1}{N} \sum_{i=1}^N x_i$$

$S^2(x)$ - выборочная дисперсия случайной величины — это оценка теоретической дисперсии распределения, рассчитанная на основе данных выборки:

$$S^2(x) = \frac{1}{N} \sum_{i=1}^N (x_i - x')^2 = \frac{1}{N} \sum_{i=1}^N x_i^2 - (x')^2$$

$x_{i+\tau}$ - множество значений другой случайной величины (полученной из значений прошлой случайной величины, но с некоторым смещением);

n - мощность множества случайных величин;

τ - смещение последовательности.

ХОД РАБОТЫ

В качестве генераторов случайных чисел были выбраны:

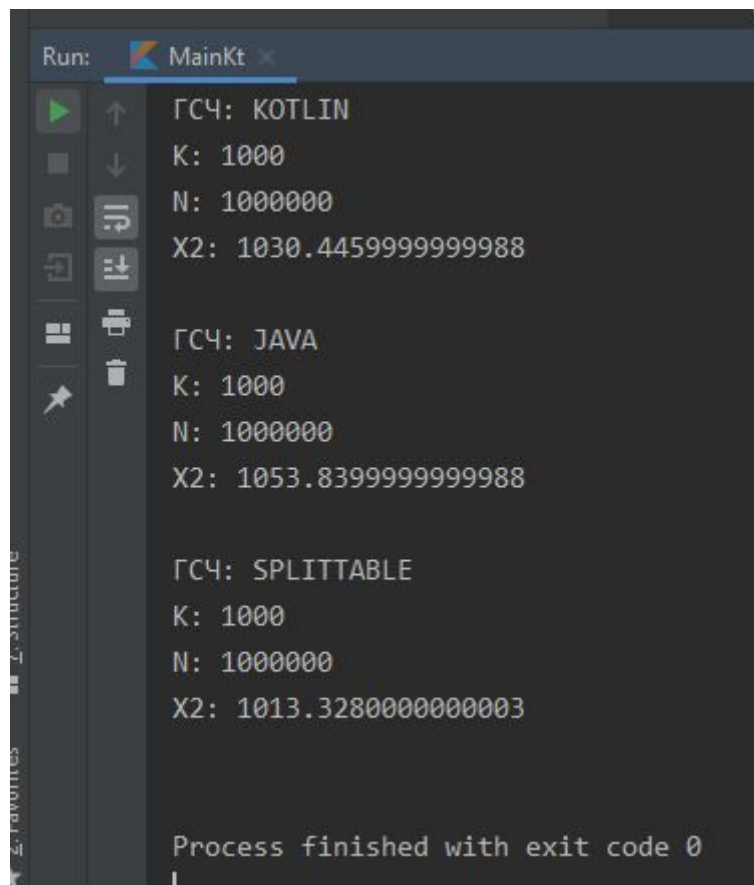
1. Генератор из стандартной библиотеки языка программирования Kotlin (`kotlin.random.Random`).
2. Генератор из стандартной библиотеки языка программирования Java (`java.util.Random`)
3. Генератор однородных псевдослучайных значений, применимых для использования в изолированных параллельных вычислениях языка Java (`java.util.SplittableRandom`)

Каждый генератор имеет собственный алгоритм генерации случайных чисел и метод генерации случайного числа с плавающей точкой в диапазоне от 0 до 1.

Для расчета значения критерия Пирса необходимо выбрать количество интервалов k , количество генерируемых значений N и генератор случайных чисел. Выбранные аргументы необходимо передать в методе `generateGraph`. Затем происходит генерация чисел, их подсчет количеств попаданий на интервалы, после чего рассчитывается “хи-квадрат” и автокорреляция.

Далее построим таблицу расчета критерия Пирса при варьировании значений количества интервалов k и количества генерируемых значений N :

<i>Генератор случайных чисел</i>	<i>$k = 10\,000$ $N = 1\,000$</i>	<i>$k = 1\,000\,000$ $N = 1\,000$</i>	<i>$k = 1\,000\,000$ $N = 100$</i>
Kotlin Random	$\chi^2_{\text{эксп}} = 997.4$	$\chi^2_{\text{эксп}} = 1030.446$	$\chi^2_{\text{эксп}} = 109.716$
Java Random	$\chi^2_{\text{эксп}} = 993$	$\chi^2_{\text{эксп}} = 1053.839$	$\chi^2_{\text{эксп}} = 94.753$
SplittableRandom	$\chi^2_{\text{эксп}} = 1052.4$	$\chi^2_{\text{эксп}} = 1013.328$	$\chi^2_{\text{эксп}} = 91.609$
Табличный χ^2	$\chi^2_{\text{табл}} = 1142.848$	$\chi^2_{\text{табл}} = 1142.848$	$\chi^2_{\text{табл}} = 134,642$



The screenshot shows the 'Run' console of an IDE with a tab labeled 'MainKt'. The console output is as follows:

```
ГСЧ: KOTLIN
K: 1000
N: 1000000
X2: 1030.44599999999988

ГСЧ: JAVA
K: 1000
N: 1000000
X2: 1053.83999999999988

ГСЧ: SPLITTABLE
K: 1000
N: 1000000
X2: 1013.32800000000003

Process finished with exit code 0
```

Рисунок 1 - Результат расчета критерия Пирса с использованием генераторов случайных чисел при значениях $k = 1000$ и $N = 1000000$

Рассчитаем коэффициент автокорреляции. Для этого генерируется массив чисел количество которого равно N . Определим параметр τ (смещение в последовательности от 1 до половины наших интервалов).

```
Генератор: Kotlin  
a(1) = 0,001200  
a(2) = -0,001025  
a(3) = 0,000264  
a(4) = 0,000393  
a(5) = -0,000564  
a(6) = -0,000936  
a(7) = 0,000063  
a(8) = -0,000220  
a(9) = -0,000662  
a(10) = 0,001954  
a(11) = -0,000107  
a(12) = -0,000346  
a(13) = 0,000214  
a(14) = -0,000090  
a(15) = -0,000443  
a(16) = 0,001984  
a(17) = -0,002058  
a(18) = 0,000325  
a(19) = 0,001221  
a(20) = 0,000475
```

Рисунок 1 - Результат расчета коэффициента корреляции генераторам Kotlin при изменении τ ($k = 1000$, $N = 1\,000\,000$)


```
Генератор: Java
a(1) = 0,002265
a(2) = 0,000812
a(3) = 0,000061
a(4) = -0,000904
a(5) = -0,001765
a(6) = -0,001376
a(7) = -0,002149
a(8) = 0,002713
a(9) = 0,003454
a(10) = -0,001330
a(11) = -0,000093
a(12) = 0,000211
a(13) = 0,000428
a(14) = 0,000200
a(15) = -0,001189
a(16) = 0,000248
a(17) = -0,000887
a(18) = 0,000969
a(19) = 0,000919
a(20) = 0,000050
```

Рисунок 2 - Результат расчета коэффициента корреляции генераторам Java при изменении τ . ($k = 1000$, $N = 1\,000\,000$)

```
Генератор: Splittable  
a(1) = -0,000558  
a(2) = 0,002518  
a(3) = 0,000063  
a(4) = -0,000432  
a(5) = 0,000311  
a(6) = -0,001339  
a(7) = -0,001338  
a(8) = -0,001685  
a(9) = -0,000192  
a(10) = 0,000173  
a(11) = 0,000501  
a(12) = -0,000003  
a(13) = -0,000018  
a(14) = -0,002090  
a(15) = 0,000002  
a(16) = -0,000566  
a(17) = 0,000253  
a(18) = -0,000729  
a(19) = -0,000272  
a(20) = 0,001278
```

Рисунок 3 - Результат расчета коэффициента корреляции генераторам Splittable при изменении τ . ($k = 1000$, $N = 1\,000\,000$)

ЗАКЛЮЧЕНИЕ

В рамках лабораторной работы были проведены эксперименты с генераторами случайных чисел языков Kotlin и Java. С помощью данных генераторов был произведен анализ равномерности распределения по критерию Пирса и автокорреляции.

По результатам экспериментов можно сделать вывод о том, что гипотезы о равновероятном распределении в генераторах случайных чисел принимается, так как $\chi^2_{\text{эксп}} < \chi^2_{\text{таб}}$. Если бы $\chi^2_{\text{эксп}}$ значение попало в критическую область (была бы равна или больше чем $\chi^2_{\text{таб}}$), то гипотеза была бы отклонена. Следовательно, для всех трех различных генераторов гипотеза о равномерном распределении принимается.

По результатам эксперимента расчета критерия Пирса имеется следующая зависимость: чем больше отклонение, тем менее распределенным становится распределение. Также значение критерия Пирсона, не превышало критического значения, следовательно, гипотезу о равномерном распределении нельзя отвергнуть.

В каждом из наших проведенных экспериментов автокорреляционная функция при изменении параметра τ (смещение в последовательности от 1 до половины наших интервалов) приближена к нулю, что говорит нам об очень слабой силе корреляции. Она показывает зависимости между данными крайне мала и также можно отметить, что числа генерируются случайным образом.

ЛИСТИНГ ПРОГРАММЫ

```
/**
 * Перечисление генераторов случайных чисел
 */
enum class RandomMethod {
    KOTLIN, JAVA, SPLITTABLE;

    private val javaRandom: Random by lazy { Random() }
    private val splittableRandom: SplittableRandom by lazy { SplittableRandom() }

    fun generate() = when (this) {
        KOTLIN -> kotlin.random.Random.nextDouble()
        JAVA -> javaRandom.nextDouble()
        SPLITTABLE -> splittableRandom.nextDouble()
    }
}

fun generateGraph(k: Int, N: Int, randomMethod: RandomMethod) {
    check(N / k > 5) { "Недостаточное количество генерируемых чисел!" }
    var graph = mutableMapOf<Int, Int>()
    repeat(N) {
        val interval = (randomMethod.generate() / (1.0 / k)).toInt() + 1
        graph[interval] = graph[interval]?.plus(1) ?: 1
    }
    println("\nПопадания:\n")
    println(graph.values.toList().toColumn())
    graph = graph.toSortedMap()
    val x2 = calculateX2(graph, k, N)
    println("\nX2:\n")
    println(x2)
}
```

```

/**
 * Расчет критерия Пирса
 */
private fun calculateX2(nList: Map<Int, Int>, k: Int, n: Int) =
nList.map { (_, ni) ->
    val pi = 1.0 / k
    (ni - pi * n).pow(2.0) / (pi * n)
}.sum()

/**
 * Класс расчета автокорреляции
 */
class ACov(N: Int, randomMethod: RandomMethod) {
    private val list = mutableListOf<Double>()
    private val xAverage: Double
    private val s2: Double

    init {
        var xSum = 0.0
        var x2sum = 0.0
        repeat(N) {
            val random = randomMethod.generate()
            list.add(random)
            x2sum += random * random
            xSum += random
        }
        xAverage = xSum / N
        s2 = (x2sum / N - xAverage.pow(2.0))
    }

    fun calculateByTao(tao: Int) = (0 until list.size - tao).map { i ->
        (list[i] - xAverage) * (list[i + tao] - xAverage)
    }.sum() / ((list.size - tao) * s2)
}

```

```

companion object {
    fun calculateCov(N: Int, randomMethod: RandomMethod) {
        val aCov = ACov(N, randomMethod)
        (1..500).map { tao ->
            val a = aCov.calculateByTao(tao)
            println(a.double())
        }
    }
}

fun main() {
    generateGraph(k = 1000, N = 1_000_000, randomMethod = RandomMethod.KOTLIN)
    calculateCov(1_000, RandomMethod.KOTLIN)
}

```