Experiment No.: 6
Title: Implementation of Independence test

Batch: A2 **Roll No.:16010421073** **Experiment No.: 6**

Aim: To implement Autocorrelation test / Runs test to perform Independence test of generated random numbers.

Resources needed: Turbo C / Java / python

Theory

Problem Statement:
Write function in C / C++ / java / python or macros in MS-excel to implement Autocorrelation
/ Runs test.

Concepts:

Random Numbers generated using a known process or algorithm is called Pseudo random Number. The random numbers generates must possess the property of :

1. Uniformity
2. Independence

Tests for Independence:

These tests are done to check the independence of sequence of random numbers.

1) Runs Test

This test analyses an orderly grouping of numbers in a sequence to test the hypothesis of independence. A Run is defined as a succession of similar events preceded and followed by a different. The length of the run is the number of events that occur in the run.

In all cases, actual values are compared with expected values using chi square test.

The Runs test used re:

i) Runs Up and Down
ii) Runs above and below the mean
iii) Runs test for testing length of runs

Runs Up and Down:

In a sequence of numbers, if a number is followed by a larger number, this is an upward run. Likewise, a number followed by a smaller number is a downstream run. The numbers

are given + and – depending on whether they are followed by larger or smaller number. The last number is followed by no event. Eg. 10 numbers there will be 9 +or -. If the numbers are truly random, one would expect to find a certain numbers of runs up and down.

In a sequence of N numbers, a is the total no of runs, the mean and variance is given by the following equation

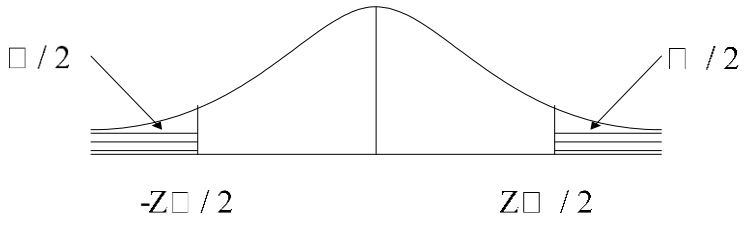
$$\mu = \frac{(2N-1)}{3} \qquad \sigma^2 = \frac{(16N-29)}{90}$$

For N > 20, the distribution of "a" is approximated by a normal distribution, N(0,1). This approximation can be used to test the independence of numbers from a generator. Finally, the standardised normal test statistics ,Zo is developed andcompared with critical value

$$Z_0 = (a - \mu) / \sigma$$

Where a is total no of runs.

Acceptance region for hypothesis of independence $-Z_{\alpha/2} \leq Z_0 \leq Z_{\alpha/2}$



2) Auto correlation Test: The test for auto correlation is concerned with dependence between numbers in a sequence. The test computes auto correlation between every m numbers starting with the ith number. Thus autocorrelation limit between following numberswould be of interest.

Ri , Ri+m , Ri+2m , Ri+(M+1)m

where M is the largest integer such that $i+(M+1)m \leq N$ where N is total number of values in the sequence.

Since the nonzero autocorrelation implies a lack of independence, the following test is appropriate:

H_0 : $\rho_{im} = 0$, if numbers are independent

H_1 : $\rho_{im} \neq 0$, if numbers are dependent

For large values of M, the distribution of the estimator of ρim, denoted $\hat{\rho}_{im}$ is approximately normal,if the values Ri , Ri+m , Ri+2m , Ri+(M+1)m are uncorrelated.

The test statistics is

$$Z_0 = \frac{\hat{\rho}_{im}}{\hat{\sigma}_{\hat{\rho}_{im}}}$$

with a mean of 0 and variance of 1,under the assumption of independence , for large M.

If $-Z\alpha/2 \leq Z0 \leq Z\alpha/2$, H0 is not rejecte for the significance level α .

3) Gap Test: The gap test is used to determine the significance of the interval between reoccurrence of the same digit. A gap of length x occurs between reoccurrence of samedigit.

4) Poker Test: The poker test for independence is based on frequency with which certain digits are repeated in a series of numbers in each case a pair of like digits appear in the numbers that were generated. In 3 digit sample of numbers there are three possibilities which are as follows:

i) The individual numbers can all be different
ii) The individual numbers can all be same
iii) There can be one pair of like digits.

Procedure:
(Write the algorithm for the test to be implemented and follow the steps given below)Steps:

- Implement either Autocorrelation Test or Runs test using C / C++ / java or macros in MS-excel
- Generate 5 sample sets (Each set consisting of 100 random numbers) of Pseudo random numbers using Linear Congruential Method.
- Execute the test using all the five sample sets of random numbers as input andusing α=0.05.
- Draw conclusions on the acceptance or rejection of the null hypothesis of independence

Results: (Program printout with output)

Code for Runs test:

```
#include <bits/stdc++.h>
#include <cmath>
using namespace std;

void linearCongruentialMethod(int Xo, int m, int a, int c, vector<int>& randomNums, int
noOfRandomNums) {
    randomNums[0] = Xo;

    for (int i = 1; i < noOfRandomNums; i++) {
        randomNums[i] = ((randomNums[i - 1] * a) + c) % m;
    }
}

int main() {
    int Xo = 1;
    int m = 128;
    int a[5] = {5, 9, 13, 1, 5};
    int c[5] = {7, 3, 5, 15, 9};

    int noOfRandomNums = 100;
```

```
    for (int j = 0; j < 5; j++) {
        vector<int> randomNums(noOfRandomNums);

        linearCongruentialMethod(Xo, m, a[j], c[j], randomNums, noOfRandomNums);

        cout << "Random Numbers: " << endl;

        for (int i = 0; i < noOfRandomNums; i++) {
            cout << randomNums[i] << " ";
        }

        int run[100];

        for (int x = 0; x < 99; x++) {
            if (randomNums[x] < randomNums[x + 1]) {
                run[x] = 1;
            } else {
                run[x] = 0;
            }
        }

        cout << endl;

        int r = 1;
        int temp = run[0];

        for (int x = 1; x < 99; x++) {
            if (temp == run[x]) {
                //
            } else {
                r = r + 1;
                temp = run[x];
            }
        }

        cout << "Run value: " << r << endl;

        float u;
        float p;

        u = (2 * 100 - 1) / 3.0;
        p = (6 * 100 - 29) / 90.0;
        p = sqrt(p);

        float z = (r - u) / p;

        cout << "μ: " << u << endl << "σ: " << p << endl << "Zo: " << z << endl;

        if (z >= -1.92 && z <= 1.92) {
            cout << "The null hypothesis is accepted and the random numbers are valid";
        } else {
            cout << "The null hypothesis is rejected";
        }
```

```
    }
    return 0;
}
```

Output:

Sample set 1:

```
Random Numbers:
1 12 67 86 53 16 87 58 41 84 43 94 93 88 63 66 81 28 19 102 5 32 39
    74 121 100 123 110 45 104 15 82 33 44 99 118 85 48 119 90 73 116
    75 126 125 120 95 98 113 60 51 6 37 64 71 106 25 4 27 14 77 8 47
    114 65 76 3 22 117 80 23 122 105 20 107 30 29 24 127 2 17 92 83
    38 69 96 103 10 57 36 59 46 109 40 79 18 97 108 35 54
Run value: 63
μ: 66.3333
σ: 2.51882
Zo: -1.32337
The null hypothesis is accepted and the random numbers are valid
```

Sample set 2:

```
Random Numbers:
1 12 111 106 61 40 107 70 121 68 103 34 53 96 99 126 113 124 95 90 45
    24 91 54 105 52 87 18 37 80 83 110 97 108 79 74 29 8 75 38 89 36
    71 2 21 64 67 94 81 92 63 58 13 120 59 22 73 20 55 114 5 48 51 78
    65 76 47 42 125 104 43 6 57 4 39 98 117 32 35 62 49 60 31 26 109
    88 27 118 41 116 23 82 101 16 19 46 33 44 15 10
Run value: 64
μ: 66.3333
σ: 2.51882
Zo: -0.926361
The null hypothesis is accepted and the random numbers are valid
```

Sample set 3:

```
Random Numbers:
1 18 111 40 13 46 91 36 89 10 7 96 101 38 115 92 49 2 31 24 61 30 11
    20 9 122 55 80 21 22 35 76 97 114 79 8 109 14 59 4 57 106 103 64
    69 6 83 60 17 98 127 120 29 126 107 116 105 90 23 48 117 118 3 44
    65 82 47 104 77 110 27 100 25 74 71 32 37 102 51 28 113 66 95 88
    125 94 75 84 73 58 119 16 85 86 99 12 33 50 15 72
Run value: 67
μ: 66.3333
σ: 2.51882
Zo: 0.264673
The null hypothesis is accepted and the random numbers are valid
```

Sample set 4:

```
Random Numbers:
1 16 31 46 61 76 91 106 121 8 23 38 53 68 83 98 113 0 15 30 45 60 75
    90 105 120 7 22 37 52 67 82 97 112 127 14 29 44 59 74 89 104 119
    6 21 36 51 66 81 96 111 126 13 28 43 58 73 88 103 118 5 20 35 50
    65 80 95 110 125 12 27 42 57 72 87 102 117 4 19 34 49 64 79 94
    109 124 11 26 41 56 71 86 101 116 3 18 33 48 63 78
Run value: 23
μ: 66.3333
σ: 2.51882
Zo: -17.2038
The null hypothesis is rejected hence random numbers are not valid.
```

Sample set 5:

```
Random Numbers:
1 14 79 20 109 42 91 80 25 6 39 76 5 34 51 8 49 126 127 4 29 26 11 64
    73 118 87 60 53 18 99 120 97 110 47 116 77 10 59 48 121 102 7 44
    101 2 19 104 17 94 95 100 125 122 107 32 41 86 55 28 21 114 67 88
    65 78 15 84 45 106 27 16 89 70 103 12 69 98 115 72 113 62 63 68
    93 90 75 0 9 54 23 124 117 82 35 56 33 46 111 52
Run value: 62
μ: 66.3333
σ: 2.51882
Zo: -1.72038
The null hypothesis is accepted and the random numbers are valid
```

Questions:

1) **Give an example and interpret the need of Independence test.**

Ans: The need for an independence test arises when we want to determine if there is a significant relationship between two categorical variables. It helps us understand if changes in one variable are associated with changes in the other, or if they occur independently. This is important in various fields such as medicine, sociology, and market research to make informed decisions based on data analysis. For example, in a clinical trial, we might use an independence test to assess if a new drug is more effective than a standard treatment. Overall, independence tests are vital tools for analyzing relationships and making evidence-based conclusions.

2) **What is Type 1 and Type 2 error?**

Answer:

actually true.

Type I Error:

Definition: This error occurs when the null hypothesis is incorrectly rejected when it is
Interpretation: In the context of random number testing, a Type I error would mean concluding that a set of numbers is not random when, in fact, they are random.
Consequence: This can lead to the rejection of a genuinely random sequence, potentially causing undue concern or incorrect actions based on the false belief that the sequence is non-random.

Type II Error :

Definition: This error occurs when the null hypothesis is not rejected when it is actually false.
Interpretation: In the context of random number testing, a Type II error would mean failing to identify a non-random pattern in a sequence when, in fact, such a pattern exists.
Consequence: This could result in accepting a non-random sequence as random, potentially leading to erroneous conclusions or decisions based on the incorrect assumption of randomness.

3) What of the independence tests make use of Chi square test?

Answer: The Chi-square test is a statistical method used to determine whether there is a significant association between two categorical variables. It measures the difference between observed and expected frequencies of categorical data and assesses whether this difference is likely to have occurred by chance. The test is commonly used in various contexts, including analyzing survey data, clinical trials, and market research, to investigate relationships between variables. It provides a p-value, which indicates the probability of obtaining the observed results if there is no true association between the variables. If the p-value is below a predetermined significance level (usually 0.05), the association is considered statistically significant, and the null hypothesis of independence is rejected.

Outcomes: CO2: Generate pseudorandom numbers and perform empirical tests to measure the quality of a pseudo random number generator.

Conclusion: (Conclusion to be based on outcomes)
In this experiment, we implemented Runs test to perform Independence test ofgenerated random numbers.

Grade: AA / AB / BB / BC / CC / CD /D

Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

.

1. Jerry Banks, Jerry Banks, John Carson, Barry Nelson, and David M. Nicol; “Discrete EventSystem System Simulation”, Third Edition, Pearson Education
2. "Linear Congruential Generators" by Joe Bolte, Wolfram Demonstrations Project.
3. Severance, Frank (2001). System Modeling and Simulation. John Wiley & Sons, Ltd. p. 86. ISBN 0-471-49694-4.
4. "A collection of selected pseudorandom number generators with linear structures, K. Entacher, 1997". Retrieved 16 June 2012.
5. GNU Scientific Library: Other random number generators
6. Novice Forth library
7. Matsumoto, Makoto, and Takuji Nishimura (1998) ACM Transactions on Modeling and Computer Simulation
8. S.K. Park and K.W. Miller (1988). "Random Number Generators: Good Ones Are Hard To Find". Communications of the ACM31 (10): 1192–1201. doi:10.1145/63039.63042.
9. D. E. Knuth. *The Art of Computer Programming*, Volume 2: *Seminumerical Algorithms*, Third Edition. Addison-Wesley, 1997. ISBN 0-201-89684-2. Section 3.2.1: The Linear Congruential Method, pp. 10–26.
10. P. L'Ecuyer (1999). "Tables of Linear Congruential Generators of Different Sizes and Good Lattice Structure". *Mathematics of Computation*68 (225): 249–260. doi:10.1090/S0025-5718-99-00996-5.
11. Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007), "Section 7.1.1. Some History", *Numerical Recipes: The Art of Scientific Computing* (3rd ed.), New York: Cambridge University Press, ISBN 978-0-521-88068-8
12. Gentle, James E., (2003). *Random Number Generation and Monte Carlo Methods*, 2nd edition, Springer, ISBN 0-387-00178-6.
13. Joan Boyar (1989). "Inferring sequences produced by pseudo-random number generators". *Journal of the ACM*36 (1): 129–141. doi:10.1145/58562.59305. (in this paper, efficient algorithms are given for inferring sequences produced by certain pseudo-random number generator