

**Experiment No.: 6**

**Title: Implementation of Independence test**

**Batch: A3 Roll No.:1610421119 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

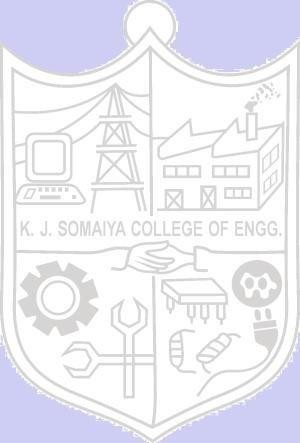
Problem Statement:

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:

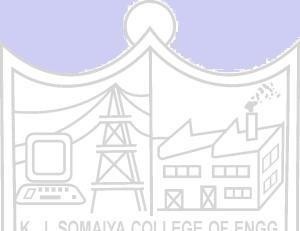
* 1. Runs Up and Down
  2. Runs above and below the mean
  3. Runs test for testing length of runs Runs Up and Down:

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

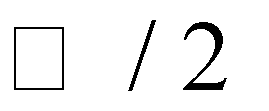
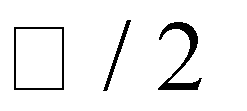


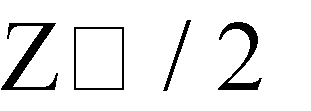
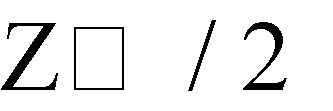
For N > 20, the distribution of “a” is approximated by a normal distribution, N(0,1). This approximation can be used to test the independence of numbers from a generator. Finally, the standardised normal test statistics ,Zo is developed and compared with critical value

Z 0 = (a - µ) / σ

Where a is total no of runs.

Acceptance region for hypothesis of independence -Za/2 ≤ Z0 ≤ Za/2



- 

1. 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 numbers would be of interest.

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

where M is the largest integer such that i+(M+1)m ≤ 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 :

*H*1 :

ρ*im* = 0,

ρ*im* ≠ 0,

if numbers are independent if numbers are dependent

For large values of M, the distribution of the estimator of ρim, denoted ρˆ*im*

normal,if the values Ri , Ri+m , Ri+2m , Ri+(M+1)m are uncorrelated. The test statistics is

is approximately

*Z* = ρˆ*im*

0 σˆ ρˆ

*im*

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

If -Zα/2 ≤ Z0 ≤ Zα/2 , H0 is not rejecte for the significance level α .

1. 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 same digit.
2. 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:
   1. The individual numbers can all be different
   2. The individual numbers can all be same
   3. 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 and using α=0.05.
* Draw conclusions on the acceptance or rejection of the null hypothesis of independence

Result :

**#include <iostream>**

**#include <cstdlib>**

**#include <ctime>**

**#include <cmath>**

**using namespace std;**

**void generateRandomNumbers(int N, int numbers[]) {**

**srand(time(0));**

**for (int i = 0; i < N; ++i) {**

**numbers[i] = rand() % 100 + 1;**

**}**

**}**

**int runsTest(int N, int numbers[]) {**

**int runs = 0;**

**bool isUpward = false;**

**bool isDownward = false;**

**for (int i = 0; i < N - 1; ++i) {**

**if (numbers[i] < numbers[i + 1]) {**

**if (!isUpward) {**

**runs++;**

**isUpward = true;**

**isDownward = false;**

**}**

**} else if (numbers[i] > numbers[i + 1]) {**

**if (!isDownward) {**

**runs++;**

**isUpward = false;**

**isDownward = true;**

**}**

**}**

**}**

**return runs;**

**}**

**int main() {**

**int N;**

**cout << "Enter the number of random numbers to generate: ";**

**cin >> N;**

**int numbers[N];**

**generateRandomNumbers(N, numbers);**

**cout << "Generated random numbers: ";**

**for (int i = 0; i < N; ++i) {**

**cout << numbers[i] << " ";**

**}**

**cout <<endl;**

**int runs = runsTest(N, numbers);**

**cout << "Total runs: " << runs <<endl;**

**double mu = (2 \* N - 1) / 3.0;**

**double sigma\_sq = (16 \* N - 29) / 90.0;**

**double Z\_0 = (runs - mu) / sqrt(sigma\_sq);**

**cout << "Z\_0 value: " << Z\_0 << endl;**

**double Za\_by\_2 = 1.96;**

**if (Z\_0 >= -Za\_by\_2 && Z\_0 <= Za\_by\_2)**

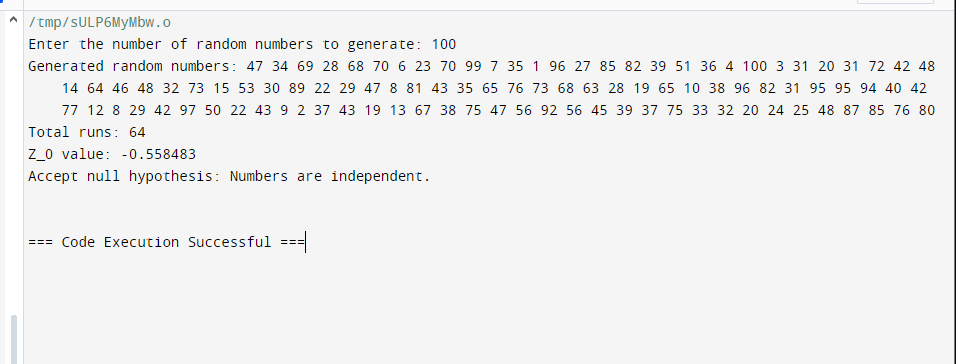
**cout << "Accept null hypothesis: Numbers are independent." << endl;**

**else**

**cout << "Reject null hypothesis: Numbers are not independent." << endl;**

**return 0;**

**}**



Questions:

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

Imagine a fast-food restaurant that wants to understand if there's a relationship between customer age groups (teens, adults, seniors) and their preferred meal options (burgers, fries, salads). They collect data on customer orders. An independence test helps determine if these two variables, age group and meal choice, are independent.

If they are independent: Age group has no bearing on meal preference. People of all ages order burgers, fries, and salads at similar rates.

If they are not independent: There's a relationship. Maybe teens tend to order more burgers and fries, while adults favor salads

1. What is Type 1 and Type 2 error?

**Type 1 Error (False Positive):** This occurs when we reject a true null hypothesis. In simpler terms, we conclude there's a relationship between variables when there actually isn't.

**Type 2 Error (False Negative):** This happens when we fail to reject a false null hypothesis. We miss a real relationship between variables.

1. What of the independence tests make use of Chi square test

The Chi-Square test of independence is a common statistical test used to assess if two categorical variables are related. It compares the observed frequencies (how often things occur in the data) with the expected frequencies (how often we would expect them to occur if the variables were independent).

A significant Chi-square test result (low p-value) suggests the observed data deviates from what we'd expect if the variables were independent, indicating a relationship.

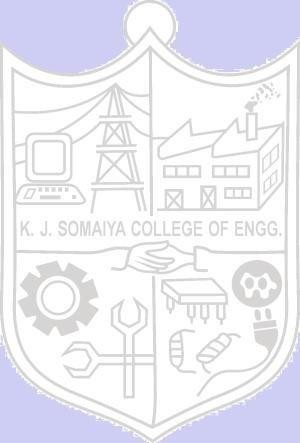
A non-significant result (high p-value) implies the observed data aligns with what we'd expect if the variables were independent, suggesting no relationship.

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"](http://demonstrations.wolfram.com/LinearCongruentialGenerators/) by Joe Bolte, [Wolfram Demonstrations Project.](http://en.wikipedia.org/wiki/Wolfram_Demonstrations_Project)
3. Severance, Frank (2001). System Modeling and Simulation. John Wiley & Sons, Ltd. p. 86. [ISBN 0-471-49694-4.](http://en.wikipedia.org/wiki/International_Standard_Book_Number)
4. ["A collection of selected pseudorandom number generators with linear structures, K.](http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.53.3686&rep=rep1&type=pdf) [Entacher, 1997".](http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.53.3686&rep=rep1&type=pdf) Retrieved 16 June 2012.
5. [GNU Scientific Library: Other random number generators](http://www.gnu.org/software/gsl/manual/html_node/Other-random-number-generators.html)
6. [Novice Forth library](http://www.forth.org/novice.html)
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](http://portal.acm.org/citation.cfm?id=63042) [Hard To Find". Communications of the ACM31](http://portal.acm.org/citation.cfm?id=63042) (10): 1192–1201. [doi:10.1145/63039.63042.](http://en.wikipedia.org/wiki/Digital_object_identifier)
9. [D. E. Knuth.](http://en.wikipedia.org/wiki/Donald_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](http://citeseer.ist.psu.edu/132363.html) [Good Lattice Structure". *Mathematics of Computation*68](http://citeseer.ist.psu.edu/132363.html) (225): 249–260. [doi:10.1090/S0025-5718-99-00996-5.](http://en.wikipedia.org/wiki/Digital_object_identifier)
11. Press, WH; Teukolsky, SA; Vetterling, WT; Flannery, BP (2007), ["Section 7.1.1.](http://apps.nrbook.com/empanel/index.html#pg%3D343) [Some History",](http://apps.nrbook.com/empanel/index.html#pg%3D343) *Numerical Recipes: The Art of Scientific Computing* (3rd ed.), New York: Cambridge University Press[, ISBN 978-0-521-88068-8](http://en.wikipedia.org/wiki/International_Standard_Book_Number)
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](http://portal.acm.org/citation.cfm?id=59305&dl=ACM&coll=portal) [generators". *Journal of the ACM*36](http://portal.acm.org/citation.cfm?id=59305&dl=ACM&coll=portal) (1): 129–141. [doi:10.1145/58562.59305.](http://en.wikipedia.org/wiki/Digital_object_identifier) (in this paper, efficient algorithms are given for inferring sequences produced by certain pseudo-random number generator