Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Кафедра цифрових технологій в енергетиці

ЗВІТ   
з виконання лабораторної роботи №2  
з дисципліни «Математичні методи в психології»

«Статистичні міри шкали порядку»

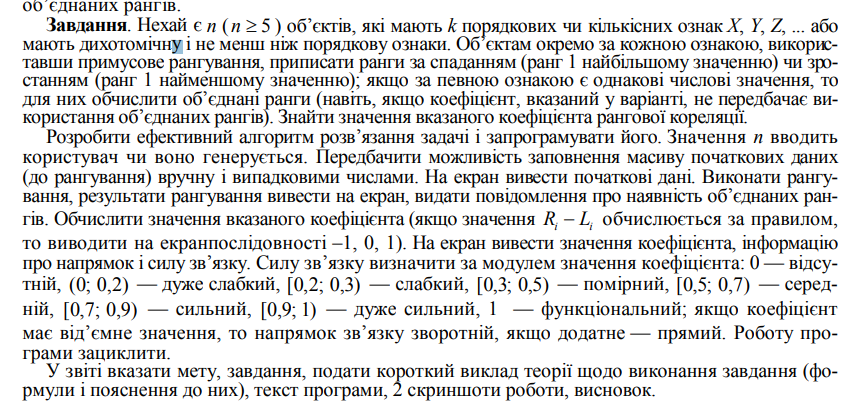
Варіант 8

Виконав: студент групи ТР-12 Руденко В.І

**Мета:**

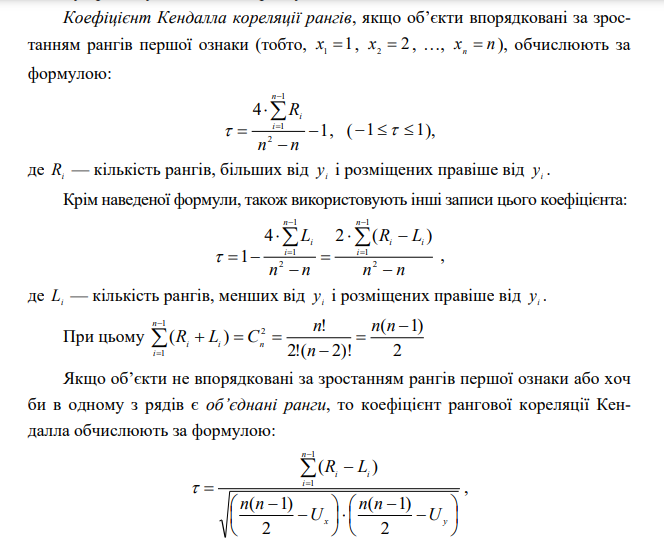
Навчитися розраховувати коефіцієнти рангової кореляції для звичайних і об’єднаних рангів.

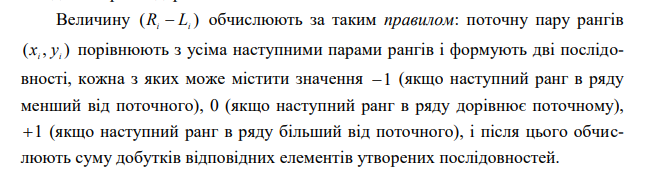
**Завдання:**



1. Кожен об’єкт має 2 ознаки. Прорангувати значення ознак за зростанням. Обчислити значення коефіцієнта рангової кореляції Кендалла для звичайних рангів за формулою з величиною Ri  Li , обчисленою за правилом.

Теорія:





Хід Роботи:

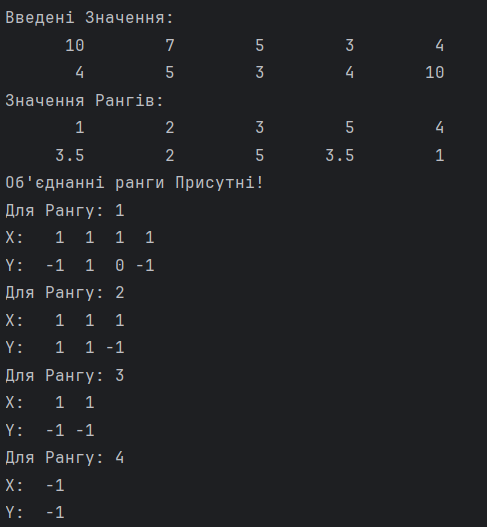


Table 1 Генерація Значень

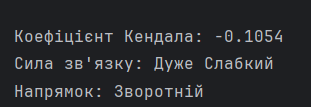


Table 2 Пошук значення коефіцієнту

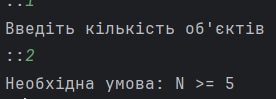


Table 3 Перевірка на кількість об'єктів

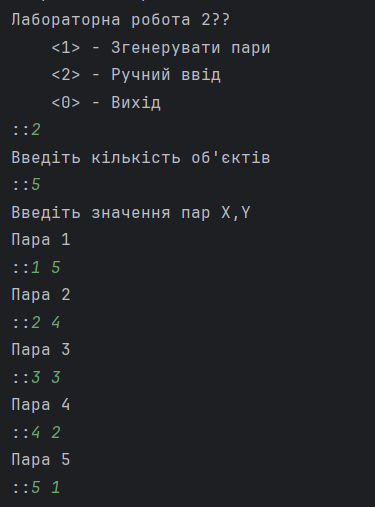


Table 4 Приклад ручного вводу

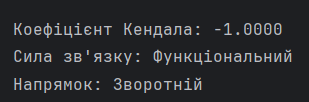


Table 5 Результат опрацювання

**Висновок:**

У результаті виконання лабораторної роботи ми навчилися розраховувати коефіцієнти рангової кореляції для звичайних і об'єднаних рангів. Дослідження проводилося на основі двох ознак кожного об'єкта.

Спочатку були проранжовані значення ознак за зростанням, використовуючи вбудовані функції сортування та присвоєння рангів. Після цього було обчислено величинRi−Li для кожної пари рангів.

За отриманими результатами був обчислений коефіцієнт рангової кореляції Кендалла для звичайних рангів за відповідною формулою

Код Програми:  
*package LR2;  
  
import java.util.Random;  
import java.util.Scanner;  
  
public class Main  
{  
 enum UIState {  
 Exit,  
 RandomGeneration,  
 ManualInput,  
 InputWait;  
  
 public static UIState getByValue(int value) {  
 switch (value) {  
 case 0:  
 return Exit;  
 case 1:  
 return RandomGeneration;  
 case 2:  
 return ManualInput;  
 default:  
 return InputWait;  
 }  
 }  
 }  
  
 public static void main(String[] args)  
 {  
 System.out.println("<><><><>");  
 ConsoleUI();  
 }  
  
 private static void ConsoleUI()  
 {  
 Scanner scan = new Scanner(System.in);  
 UIState CurrentProgramPart = UIState.InputWait;  
 while(true)  
 {  
 System.out.print("Лабораторна робота 2??\n\t<1> - Згенерувати пари\n\t<2> - Ручний ввід\n\t<0> - Вихід\n::");  
 CurrentProgramPart = UIState.getByValue(scan.nextInt()) ;  
 switch (CurrentProgramPart)  
 {  
 case UIState.Exit:  
 return;  
 case UIState.InputWait:  
 System.out.println("Невіриний ввід....");  
 break;  
 default:  
 MainRun(CurrentProgramPart,scan);  
 break;  
 }  
  
 }  
 }  
  
 private static void MainRun(UIState currentProgramPart, Scanner scan) {  
 double Array[][], RankArray[][];  
 int N = 0;  
 System.out.print("Введіть кількість об'єктів\n::");  
 while (true) {  
 N = scan.nextInt();  
 if (N >= 5) {  
 break;  
 }  
 System.out.print("Необхідна умова: N >= 5\n::");  
 }  
 Array = new double[2][N];  
 RankArray = new double[2][N];  
 for (int i = 0; i < 2; i++)  
 {  
 for(int j = 0; j < N; j++)  
 {  
 RankArray[i][j] = 0;  
 }  
 }  
 switch (currentProgramPart) {  
 case UIState.RandomGeneration:  
 System.out.print("Введення Верхньої межі ознаки Х[1;N]\n::");  
 int XRang[] = new int[]{1, scan.nextInt()};  
 System.out.print("Введення Верхньої межі ознаки Y[1;N]\n::");  
 int YRang[] = new int[]{1, scan.nextInt()};  
 Random rand = new Random();  
 for (int i = 0; i < N; i++) {  
 Array[0][i] = rand.nextInt(XRang[1] - XRang[0] + 1) + XRang[0];  
 Array[1][i] = rand.nextInt(YRang[1] - YRang[0] + 1) + YRang[0];  
 }  
 break;  
 case UIState.ManualInput:  
 System.out.println("Введіть значення пар X,Y");  
 for (int i = 0; i < N; i++) {  
 System.out.printf("Пара %d\n::", i+1);  
 Array[0][i] = scan.nextDouble();  
 Array[1][i] = scan.nextDouble();  
 }  
 break;  
 default:  
 return;  
 }  
  
 //Output-BasicDisplay/  
 System.out.println("Введені Значення:");  
 for (int i = 0; i < 2; i++)  
 {  
 for(int j = 0; j < N; j++)  
 {  
 if (Array[i][j] == Math.floor(Array[i][j])) {  
 System.out.printf("%8.0f ",Array[i][j]);  
 } else {  
 System.out.printf("%8.1f ",Array[i][j]);  
 }  
 }  
 System.out.println();  
 }  
 //////////////////////  
 int  
 PairArray[][] = new int[2][N],  
 PairCounter = 0,  
 PairLastPosition = 0;  
  
 //Rank-Calc///////////  
  
 boolean UnionRank = false, localUnionRank = false;  
 for (int i = 0; i < 2; i++)  
 {  
 int LowestPosition = 0;  
 int RankScore = 1;  
 for(int j = 0; j < N; j++)  
 {  
 if(RankArray[i][LowestPosition] != 0)  
 {  
 boolean EndPoint = false;  
 for(int k = 0; k < N; k++)  
 {  
 if(RankArray[i][k] == 0)  
 {  
 LowestPosition = k;  
 EndPoint = true;  
 break;  
 }  
 }  
 if(!EndPoint)  
 {  
 break;  
 }  
 }  
 for(int k = 0; k < N; k++)  
 {  
 if(RankArray[i][k] != 0)  
 {  
 continue;  
 }  
 if(Array[i][k] > Array[i][LowestPosition])  
 {  
 LowestPosition = k;  
 }  
 }  
 for(int k = 0; k < N; k++)  
 {  
 if (Array[i][k] == Array[i][LowestPosition] && k != LowestPosition)  
 {  
 UnionRank = true;  
 localUnionRank = true;  
 break;  
 }  
 }  
 if(localUnionRank) {  
 PairCounter = 0;  
 for (int k = 0; k < N; k++)  
 {  
 if (Array[i][k] == Array[i][LowestPosition])  
 {  
 RankArray[i][k] = (double) (RankScore + 0.5);  
 PairCounter++;  
 }  
 }  
 RankScore++;  
 PairArray[i][PairLastPosition++] = PairCounter;  
 localUnionRank = false;  
 }  
 else  
 {  
 RankArray[i][LowestPosition] = (double) RankScore;  
 }  
 RankScore++;  
 }  
 }  
 //////////////////////  
 //Output-RankDisplay//  
 System.out.println("Значення Рангів:");  
 for (int i = 0; i < 2; i++)  
 {  
 for(int j = 0; j < N; j++)  
 {  
 if (RankArray[i][j] == Math.floor(RankArray[i][j])) {  
 System.out.printf("%8.0f ",RankArray[i][j]);  
 } else {  
 System.out.printf("%8.1f ",RankArray[i][j]);  
 }  
 }  
 System.out.println();  
 }  
 if(UnionRank)  
 {  
 System.out.println("Об'єднанні ранги Присутні!");  
 }  
 else  
 {  
 System.out.println("Об'єднанні ранги Відсутні!");  
 }  
 //////////////////////  
  
 //(Ri-Li)/////////////  
 int currentRiLiSum = 0;  
 for (int i = 0; i < N; i++) {  
 int currentRiLiArray[][] = new int[2][N];  
 if(N-1 != i)  
 {  
 System.out.printf("Для Рангу: %d\n", i + 1);  
 }  
 for (int k = 0; k < 2; k++) {  
 if(N-1 != i)  
 {  
 if (k == 0) {  
 System.out.print("X: ");  
 } else {  
 System.out.print("Y: ");  
 }  
 }  
 for (int j = i + 1; j < N; j++) {  
 if (RankArray[k][i] < RankArray[k][j]) {  
 currentRiLiArray[k][j] = 1;  
 } else if (RankArray[k][i] > RankArray[k][j]) {  
 currentRiLiArray[k][j] = -1;  
 } else {  
 currentRiLiArray[k][j] = 0;  
 }  
 System.out.printf("%3.0f", (double) currentRiLiArray[k][j]);  
 }  
 System.out.println();  
 }  
  
 for (int j = i + 1; j < N; j++) {  
 currentRiLiSum += currentRiLiArray[0][j] \* currentRiLiArray[1][j];  
 }  
 }  
 //////////////////////  
 //Kendal-SMT//////////  
 double KendalValue = 0;  
 if(!UnionRank)  
 {  
 KendalValue = (double) (2 \* currentRiLiSum) /(N\*(N-1));  
 }  
 else  
 {  
 double Ux=0,Uy=0;  
 for(int i=0;i<N;i++)  
 {  
 Ux+=(PairArray[0][i]\*(PairArray[0][i]-1));  
 Uy+=(PairArray[1][i]\*(PairArray[1][i]-1));  
 }  
 Ux/=2;  
 Uy/=2;  
 double Divider = Math.sqrt(((double) (N \* (N - 1)) /2-Ux)\*((double) (N \* (N - 1)) /2-Uy));  
 KendalValue = currentRiLiSum/Divider;  
 }  
  
 System.out.printf("Коефіцієнт Кендала: %.4f\n",KendalValue);  
 String Result = "";  
 double TempValue = Math.abs(KendalValue);  
 if(TempValue == 0)  
 {Result = "Відсутній";}  
 else if (TempValue > 0 && TempValue<0.2)  
 {Result = "Дуже Слабкий";}  
 else if (TempValue>=0.2 && TempValue <0.3)  
 {Result = "Слабкий";}  
 else if (TempValue>=0.3 && TempValue <0.5)  
 {Result = "Помірний";}  
 else if (TempValue>=0.5 && TempValue <0.7)  
 {Result = "Середній";}  
 else if (TempValue>=0.7 && TempValue <0.9)  
 {Result = "Сильний";}  
 else if (TempValue>=0.9 && TempValue <1)  
 {Result = "Дуже Сильний";}  
 else  
 {Result = "Функціональний";}  
 System.out.printf("Сила зв'язку: "+Result+"\n");  
 System.out.println("Напрямок: "+(KendalValue>=0?"Прямий":"Зворотній"));  
 //////////////////////  
 }*  
}