МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Інститут ІКНІ
Кафедра ПЗ

ЗВІТ

До лабораторної роботи № 2

На тему: “Інтервальні статистичні ряди, їх геометричне зображення та основні числові параметри“

З дисципліни: “Статистичні методи аналізу диних”

Лектор:
професор каф. ПЗ
Гавриш В. І.
Виконав:
ст. групи ПЗ-26
Закаляк Р. А.
Прийняла:
асистент каф. ПЗ
Джумеля Е. А.
«__» _____ 2022 р.
Σ= ____ .

Львів – 2022

Тема: Інтервальні статистичні ряди, їх геометричне зображення та основні числові параметри.

Мета роботи: із використанням експериментальних даних сформувати інтервальний статистичний ряд за частотами та відносними частотами, визначити його основні числові параметри, та виконати геометричне відображення.

Індивідуальне завдання:

Створити програмні засоби, які дають змогу формувати інтервальний статистичний ряд за частотами та відносними частотами і визначати його основні числові параметри, а саме: середнє статистичне, моду, медіану, розмах, дисперсію, середнє квадратичне відхилення, виправлену дисперсію, виправлене середнє квадратичне відхилення, варіацію, початкові та центральні моменти певного порядку, асиметрію та ексцес, а також геометрично відображати гістограму і кумулятивну криву за частотами та відносними частотами і емпіричну функцію розподілу.

Експериментальні дані:

0.38	0.37	0.38	0.36	0.40	0.36	0.48
0.14	0.28	0.31	0.57	0.65	0.78	0.70
0.52	0.19	0.11	0.00	0.57	0.42	0.25

Хід роботи

Розроблено обчислювальну програму мовою програмування Java з використанням JavaFX у середовищі Eclipse, яка складається з трьох файлів: Lab2.java, IntervalStatisticSeriese.java., IntervalStatisticModel.java. Наведено дані файли, що містить всі функції, які використано для визначення числових параметрів дискретного статистичного ряду та його геометричного відображення.

IntervalStatisticSeriese.java

```
import java.util.List;

public class IntervalStatistic {
	static double sum(List<Double> arr) {
```

```
		double sum=0;
		for(double n:arr) {
			sum+=n;
		}
		return sum;
	}
	static double averStat(List<Double> arr) {
		double aver=sum(arr)/arr.size();
		return aver;
	}
	static double mode(List<Double> arr) {
		Double maxN=1.0, maxEl=-1.0;
		int n=0;
		for(int i=0; i<arr.size();i++) {
			for(int j=0; j<arr.size();j++) {
				if(arr.get(j)==arr.get(i)) {
					n++;
				}
			}
			if(n>maxN) {
				maxN=(double)n;
				maxEl=arr.get(i);
			}
			n=0;
		}
		return maxEl;
	}
	static double intervalMode(double[][] arr) {
		int interval=0;
		double max=0, mode;
		for(int i=0; i<arr.length; i++) {
			if(arr[i][2]>max) {
				max=arr[i][2];
				interval = i;
			}

		}

		mode = arr[interval][0]+(((arr[interval][2]-arr[interval-1][2])/(2*arr[interval][2]-arr[interval-1][2]-arr[interval+1][2]))*(arr[interval][1]-arr[interval][0]));
		return mode;
	}
	static double median(double[][] arr, double[][] accumulativeArr) {
		double median=0;
		int interval = (int)arr.length/2;
		median = arr[interval][0]+(arr[interval][1]-arr[interval][0])*((accumulativeArr[accumulativeArr.length-1][2]/2-accumulativeArr[interval-1][2])/(arr[interval][2]));
	//	median = arr[interval][0]+(arr[interval][1]-arr[interval][0])*((0.5-accumulativeArr[interval-1][2])/(accumulativeArr[interval][2]-accumulativeArr[interval-1][2]));

		return median;
	}
	static double range(double[][] arr) {
		double range=0;
		range = arr[arr.length-1][1]- arr[0][0];
		return range;
	}
	static double dispersia(List<Double> arr) {
		double aver = averStat(arr);
		double dispersia=0;
		for(double n:arr) {
			dispersia+=Math.pow(n-aver, 2);
		}
		dispersia/=arr.size();
		return dispersia;
	}
	static double standardError(List<Double> arr) {
		double standardError=Math.sqrt(dispersia(arr));
		return standardError;
	}
	static double vyprDispersia(List<Double> arr) {
		double vyprDispersia = dispersia(arr)*arr.size()/(arr.size()-1);
		return vyprDispersia;
	}
```

```
	static double vyprStandardError(List<Double> arr) {
		return Math.sqrt(vyprDispersia(arr));
	}
	static double variation(List<Double> arr) {
		return standardError(arr)/averStat(arr);
	}
	static double startMoment(List<Double> arr, int order) {
		double startMoment=0;
		for(var i=0; i<arr.size(); i++){
			startMoment+=Math.pow(arr.get(i),order);
			 }
		startMoment/=arr.size();
		return startMoment;
	}
	static double centralMoment(List<Double> arr, int order) {
		double centralMoment=0;
		double aver = averStat(arr);
		for(var i=0; i<arr.size(); i++){
			centralMoment+=Math.pow(arr.get(i)-aver,order);
			 }
		centralMoment/=arr.size();
		return centralMoment;
	}
	static double assymetry(List<Double> arr){
		double assymetry =
centralMoment(arr,3)/Math.pow(standardError(arr),3);
		if(assymetry<0) {
			System.out.println("Крива розподілу є зсунутою праворуч");
		}else {
			System.out.println("Крива розподілу є зсунутою ліворуч");
		}
		 return assymetry;
	}
	static double excess(List<Double> arr){
		double excess =
centralMoment(arr,4)/Math.pow(standardError(arr),4)-3;

		 return excess;
	}
}


}
```

Lab2.java

```
//import java.awt.Label;
import javafx.scene.control.ScrollPane;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Scanner;

import javafx.geometry.Insets;
import javafx.scene.control.Label;
import javafx.scene.control.Button;
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Node;
import javafx.scene.Scene;
import javafx.scene.chart.BarChart;
import javafx.scene.chart.CategoryAxis;
import javafx.scene.chart.LineChart;
import javafx.scene.chart.NumberAxis;
import javafx.scene.chart.XYChart;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

public class Lab2_Main extends Application {


//		static  ArrayList<Double>  arr  =new  ArrayList<>(Arrays.asList(0.38,
0.34, 0.37, 0.56, 0.63, 0.03, 0.96, 0.54,0.24, 0.19, 0.12, 0.00,
```

```
        //                              0.26, 0.34, 0.57, 0.72, 0.77, 0.85, 0.64 ,0.45, 0.24,
0.99, 0.59, 0.43,
        //                              0.23, 0.12, 0.24, 0.36, 0.52, 0.56, 0.68 ,0.62 ,0.52,
0.20, 0.24, 0.19));
                        static  ArrayList<Double>  arr  =new  ArrayList<>(Arrays.asList(0.38,
0.37, 0.38, 0.36, 0.40, 0.36, 0.48,
                                        0.14, 0.28, 0.31, 0.57, 0.65, 0.78, 0.70,
                                        0.52, 0.19, 0.11, 0.00, 0.57, 0.42, 0.25));

        //      static  ArrayList<Double>  arr  =new  ArrayList<>(Arrays.asList(0.35,  0.38,
0.42, 0.47, 0.91, 0.89, 0.59, 0.38, 0.27, 0.19, 0.02, 0.14, 0.28, 0.31, 0.57
        //                                                                      , 0.65, 0.78,
0.70, 0.52, 0.19, 0.11, 0.00, 0.25, 0.32, 0.36, 0.44, 0.65, 0.88, 0.57, 0.42, 0.25, 0.00
        //                                                                    , 0.52));

                        static   HashMap <Double,Double> frequencyRow = new  HashMap<>();
                        static   HashMap <Double,Double> relFrequencyRow = new  HashMap<>();
                        static        IntervalStaticticModel        model        =        new
IntervalStaticticModel(arr);
                        public static void main(String[] args) {
                                //IntervalStaticticModel          model          =          new
IntervalStaticticModel(arr);

                                for(Double n : arr) {
                                        if(frequencyRow.containsKey(n)) {

frequencyRow.replace(n,frequencyRow.get(n)+1);
                                        }else {
                                                frequencyRow.put(n, 1.0);
                                        }
                                }
                                for(Double n : arr) {
                                        if(relFrequencyRow.containsKey(n)) {

relFrequencyRow.replace(n,relFrequencyRow.get(n)+1);
                                        }else {
                                                relFrequencyRow.put(n, 1.0);
                                        }
                                }
                                for(int i=0; i<relFrequencyRow.size();i++) {
                                        relFrequencyRow.replace((Double)
relFrequencyRow.keySet().toArray()[i],
relFrequencyRow.get(relFrequencyRow.keySet().toArray()[i])/arr.size());

                                }
                                launch(args);
                        }

                        @Override
                        public void start(Stage primaryStage) {

                            arr.sort(null);
                     ArrayList<Double> set = new ArrayList<>();
                                boolean setContains=false;
                                for(double n : arr) {
                                        for(double unic: set) {
                                                if(unic==n)
                                                        setContains=true;
                                        }
                                        if(!setContains) {
                                                set.add(n);
                                        }
                                }

                        //polihon
chastot-----------------------------------------------------------------------
        //              NumberAxis xAxis = new NumberAxis(0,arr.get(arr.size()-1),0.05);
        //              xAxis.setLabel("Значення");
        //
        //              NumberAxis yAxis = new NumberAxis(0,3,1);
        //              yAxis.setLabel("Частота");
                         final CategoryAxis xAxis = new CategoryAxis();
                          final NumberAxis yAxis = new NumberAxis();
                        final BarChart<String,Number> barChart = new BarChart(xAxis,yAxis);
                         barChart.setCategoryGap(0.5);
                         barChart.setBarGap(0.5);
                         XYChart.Series series = new XYChart.Series();
```

```
                        series.setName("Гістограма за частотами");
                        for(int i=0;i<model.getIntervals().length;i++)
                                series.getData().add(new   XYChart.Data(String.format("%.3f",+
model.getIntervals()[i][0])+"-"+String.format("%.3f",          model.getIntervals()[i][1]),
model.getIntervals()[i][2]));
                        xAxis.setLabel("Інтервал");
                        yAxis.setLabel("Частота");
                        barChart.getData().addAll(series);

                        //polihon
chastot-------------------------------------------------------------------
                        //polihon                                                    vidnosnyh
chastot-----------------------------------------------------
                        final CategoryAxis xAxis1 = new CategoryAxis();
                         final NumberAxis yAxis1 = new NumberAxis();
                        final       BarChart<String,Number>       barChart1       =       new
BarChart(xAxis1,yAxis1);
                        barChart1.setCategoryGap(0.5);
                        barChart1.setBarGap(0.5);
                        XYChart.Series series1 = new XYChart.Series();
                        series1.setName("Гістограма за відносними частотами");
                        for(int i=0;i<model.getIntervals().length;i++)
                                series1.getData().add(new  XYChart.Data(String.format("%.3f",+
model.getIntervals()[i][0])+"-"+String.format("%.3f",          model.getIntervals()[i][1]),
model.getIntervals()[i][2]/arr.size()));
                        xAxis1.setLabel("Інтервал");
                        yAxis1.setLabel("Відносна частота");
                        barChart1.getData().addAll(series1);

                        //polihon                                                      vidnosnyh
chastot-----------------------------------------------------------
                        //komuliatyvna          kryva          vydnosnykh          nakopychenyh
chastot---------------------------------

                        double        [][]        accumulatedRelFrequency        =        new
double[model.getDiscretArr().length][2];
                        for(int i=0; i<model.getDiscretArr().length;i++) {
                                accumulatedRelFrequency[i][0]=model.getDiscretArr()[i][0];
                                if(i!=0)
                                        accumulatedRelFrequency[i][1]=
accumulatedRelFrequency[i-1][1]+model.getDiscretArr()[i][1]/arr.size();
                                else
                                        accumulatedRelFrequency[i][1]=
model.getDiscretArr()[i][1]/arr.size();
                        }

                        NumberAxis xAxis2 = new NumberAxis(0,arr.get(arr.size()-1),0.05);
                        xAxis2.setLabel("Значення");

                        NumberAxis yAxis2 = new NumberAxis(0,1,0.01);
                        yAxis2.setLabel("Накопичена відносна частота");

                        LineChart lineChart2 = new LineChart(xAxis2,yAxis2);
                        XYChart.Series series2 = new XYChart.Series();

                        series2.setName("Комулятивна крива відносних частот");

                        for(int i=0; i<accumulatedRelFrequency.length;i++) {
                                series2.getData().add(new
XYChart.Data(accumulatedRelFrequency[i][0],accumulatedRelFrequency[i][1]));
                        }
                        lineChart2.getData().add(series2);
                        //komuliatyvna          kryva          vydnosnykh          nakopychenyh
chastot---------------------------------
                        //komuliatyvna                    kryva                    nakopychenyh
chastot--------------------------------------------

                        double         [][]         accumulatedFrequency         =         new
double[model.getDiscretArr().length][2];
                        for(int i=0; i<model.getDiscretArr().length;i++) {
                                accumulatedFrequency[i][0]=model.getDiscretArr()[i][0];
                                if(i!=0)
                                        accumulatedFrequency[i][1]=
accumulatedFrequency[i-1][1]+model.getDiscretArr()[i][1];
                                else
                                        accumulatedFrequency[i][1]=
model.getDiscretArr()[i][1];
```

```
                        }


                        NumberAxis xAxis3 = new NumberAxis(0,model.getMax(),0.1);
                        xAxis3.setLabel("Значення");

                        NumberAxis yAxis3 = new NumberAxis(0,30,1);
                        yAxis3.setLabel("Накопичена частота");

                        LineChart lineChart3 = new LineChart(xAxis3,yAxis3);
                        XYChart.Series series3 = new XYChart.Series();

                        series3.setName("Комулятивна крива частот");

                        for(int i=0; i<accumulatedFrequency.length;i++) {
                                series3.getData().add(new
XYChart.Data(accumulatedFrequency[i][0],accumulatedFrequency[i][1]));
                        }
                        lineChart3.getData().add(series3);



                        //komuliatyvna                  kryva                   nakopychenyh
chastot-----------------------------------------------
                        //empirychna                                                funkciya
rozpodilu---------------------------------------------------------

                        double           [][]           empFunction           =           new
double[(model.getDiscretArr().length*2)][2];
                        for(int i=0, j=0; j<model.getDiscretArr().length;j++) {
                                if(i==0) {
                                        empFunction[i][0]=model.getDiscretArr()[j][0];
                                        empFunction[i][1]=      accumulatedRelFrequency[j][1];
//relFrequencyRow.get(set.get(j));
                                        i++;
                                }
                                else{
                                        empFunction[i][0]=model.getDiscretArr()[j][0]-0.001;
                                        empFunction[i][1]=  accumulatedRelFrequency[j-1][1];
                                        i++;
                                        empFunction[i][0]=model.getDiscretArr()[j][0]+0.001;
                                        empFunction[i][1]= accumulatedRelFrequency[j][1];
                                        i++;
                                }
                        }
                                empFunction[model.getDiscretArr().length*2-1][0]=
model.getDiscretArr()[model.getDiscretArr().length-1][0]+0.1;
                                empFunction[model.getDiscretArr().length*2-1][1]=
accumulatedRelFrequency[model.getDiscretArr().length-1][1];



                        NumberAxis                xAxis4                =                new
NumberAxis(empFunction[0][0],arr.get(arr.size()-1)+0.2,0.05);
                        xAxis4.setLabel("Значення");

                        NumberAxis yAxis4 = new NumberAxis(0,1.1,0.05);
                        yAxis4.setLabel("Накопичена відносна частота");

                        LineChart lineChart4 = new LineChart(xAxis4,yAxis4);
                        XYChart.Series series4 = new XYChart.Series();

                        series4.setName("Емпірична функція розподілу");

                        for(int i=0; i<empFunction.length;i++) {
                                series4.getData().add(new
XYChart.Data(empFunction[i][0],empFunction[i][1]));
                        }
                        lineChart4.getData().add(series4);

                        //empirychna                                                funkciya
rozpodilu---------------------------------------------------
                        GridPane gridPane = new GridPane();
                        GridPane gridPaneOutput = new GridPane();

                        gridPaneOutput.add( new Label("Середнє статистичне :"), 0, 0,1,1);
```

```
                    gridPaneOutput.add(                                                 new
Label(Double.toString(IntervalStatistic.averStat(model.getDiscretArrTotal()))), 1, 0,1,1);
                    gridPaneOutput.add( new Label("Мода :"), 0, 1,1,1);
                    gridPaneOutput.add(                                                 new
Label((Double.toString(IntervalStatistic.intervalMode(model.getIntervals())))), 1, 1,1,1);
                    gridPaneOutput.add( new Label("Медіана :"), 0, 2,1,1);
                    gridPaneOutput.add(                                                 new
Label(Double.toString(IntervalStatistic.median(model.getIntervals(),model.accumulatedFreque
ncy))), 1, 2,1,1);
                    gridPaneOutput.add( new Label("Розмах :"), 0, 3,1,1);
                    gridPaneOutput.add(                                                 new
Label(Double.toString(IntervalStatistic.range(model.getIntervals()))), 1, 3,1,1);
                    gridPaneOutput.add( new Label("Дисперсія :"), 0, 4,1,1);
                    gridPaneOutput.add(                                                 new
Label(Double.toString(IntervalStatistic.dispersia(model.getDiscretArrTotal()))), 1, 4,1,1);
                    gridPaneOutput.add( new Label("Середнє квадратичне відхилення :"), 0,
5,1,1);
                    gridPaneOutput.add(                                                 new
Label(Double.toString(IntervalStatistic.standardError(model.getDiscretArrTotal()))),     1,
5,1,1);
                    gridPaneOutput.add( new Label("Виправлена дисперсія :"), 0, 6,1,1);
                    gridPaneOutput.add(                                                 new
Label(Double.toString(IntervalStatistic.vyprDispersia(model.getDiscretArrTotal()))),     1,
6,1,1);
                    gridPaneOutput.add( new Label("Варіація :"), 0, 7,1,1);
                    gridPaneOutput.add(                                                 new
Label(Double.toString(IntervalStatistic.variation(model.getDiscretArrTotal()))), 1, 7,1,1);
                            System.out.println("Введіть     порядок     початкового     та
центрального моменту:");
                            Scanner sc=new Scanner(System.in);
                            int p =sc.nextInt();
                    gridPaneOutput.add(        new        Label("Початковий        момент
"+Integer.toString(p)+" порядку:"), 0, 8,1,1);
                    gridPaneOutput.add(                                                 new
Label(Double.toString(IntervalStatistic.startMoment(model.getDiscretArrTotal(),p))),     1,
8,1,1);
                    gridPaneOutput.add(        new        Label("Центральний        момент
"+Integer.toString(p)+" порядку:"), 0, 9,1,1);
                    gridPaneOutput.add(                                                 new
Label(Double.toString(IntervalStatistic.centralMoment(model.getDiscretArrTotal(),p))),   1,
9,1,1);
                    gridPaneOutput.add( new Label("Асиметрія :"), 0, 10,1,1);
                    gridPaneOutput.add(                                                 new
Label(Double.toString(IntervalStatistic.assymetry(model.getDiscretArrTotal()))),         1,
10,1,1);
                    gridPaneOutput.add( new Label("Ексцес :"), 0, 11,1,1);
                    gridPaneOutput.add(                                                 new
Label(Double.toString(IntervalStatistic.excess(model.getDiscretArrTotal()))), 1, 11,1,1);
                    gridPaneOutput.add( new Label("Інтервальний ряд :"), 0, 12,2,1);
                    Label intervals = new Label();
                    for(int i=0; i<model.getIntervals().length;i++) {

intervals.setText(intervals.getText()+"["+String.format("%.3f",+
model.getIntervals()[i][0])+"-"+String.format("%.3f",
model.getIntervals()[i][1])+"],"+'\t');
                    }
                    Label avers = new Label("");
                    for(int i=0; i<model.getDiscretArr().length;i++) {

avers.setText(avers.getText()+String.format("%.3f",model.getDiscretArr()[i][0])+'\t'+'\t');
                    }
                    Label ns = new Label("");
                    for(int i=0; i<model.getIntervals().length;i++) {

ns.setText(ns.getText()+String.format("%.3f",model.getIntervals()[i][2])+'\t'+'\t');
                    }
                    gridPaneOutput.add( new Label("Інтервали"), 0, 13,1,1);
                    gridPaneOutput.add( new Label("Z(i)"), 0, 14,1,1);
                    gridPaneOutput.add( new Label("M(i)"), 0, 15,1,1);
                    gridPaneOutput.add( intervals, 1, 13,2,1);
                    gridPaneOutput.add( avers, 1, 14,2,1);
                    gridPaneOutput.add( ns, 1, 15,2,1);

                    gridPane.add(gridPaneOutput, 0, 0, 1, 1);
                    gridPane.add(barChart, 1, 0, 1, 1);
                    gridPane.add(barChart1, 2, 0, 1, 1);
                    gridPane.add(lineChart3, 0, 1, 1, 1);
```

```
                    gridPane.add(lineChart2, 1, 1, 1, 1);
                    gridPane.add(lineChart4, 2, 1, 1, 1);
                    gridPane.setHgap(10);
                    gridPane.setVgap(10);
                    Group root = new Group();
                    root.getChildren().add(gridPane);
                    ScrollPane sp = new ScrollPane();
                    sp.setContent(root);

                    Scene scene = new Scene(sp,600,400);
                     primaryStage.setTitle("Lab1!");
                     primaryStage.setScene(scene);
                     primaryStage.show();
                }
```

IntervalStatisticModel.java

```
        import java.util.ArrayList;
        import java.util.Arrays;

        public class IntervalStaticticModel {
        //      private ArrayList<ArrayList<Double>> arr;
                private double[][] intervals;
                private double intervalWidth =0;
                private int r;
                private ArrayList<Double> discretArrTotal = new ArrayList<>();
                private double[][]discretArr;
                private double max,min;
                double [][] accumulatedFrequency;
                public IntervalStaticticModel(ArrayList<Double> arr) {
                        int arrN = arr.size();
                        ArrayList<Double> tempArr= new ArrayList<>();
                        for(Double n : arr) {
                                if(!tempArr.contains(n)) {
                                        tempArr.add(n);
                                }
                        }
                        arr.sort(null);
                         min = arr.get(0);
                         max = arr.get(arrN-1);
                        r=(int) Math.floor(1+3.322*Math.log10(arr.size()));
                        System.out.println(r);
                        intervalWidth = (max-min)/r;
                        intervals = new double[r][3];
                        for(int i=0; i<r;i++) {
                                intervals[i][0]=min+intervalWidth*i;
                                intervals[i][1]=min+intervalWidth*(i+1);
                                intervals[i][2]=0;
                                //System.out.println(intervals[i][0]+" "+intervals[i][1]);
                        }
                        for(Double n: arr) {
                                for(int i=0; i<r;i++) {

if(((n>intervals[i][0])||(i==0&&n>=intervals[i][0]))&&n<=intervals[i][1]) {
                                                intervals[i][2]++;
                                        }
                                }
                        }
                        for(int i=0; i<r;i++) {
                                System.out.println(intervals[i][0]+"       "+intervals[i][1]+"
"+intervals[i][2]);
                        }
                        discretArr = new double[r][2];
                        for(int i=0; i<r;i++) {
                                discretArr[i][0]=(intervals[i][0]+intervals[i][1])/2;
                                discretArr[i][1]=intervals[i][2];
                                for(int j=0; j<discretArr[i][1];j++) {
                                        discretArrTotal.add(discretArr[i][0]);
                                }
                                System.out.println(discretArr[i][0]+" "+discretArr[i][1]);
                        }
                        for(Double d:discretArrTotal) {
                                System.out.print(d+" ");
```

```
                        }
                        accumulatedFrequency = new double[intervals.length][3];
                for(int i=0; i<intervals.length;i++) {
                        accumulatedFrequency[i][0]=intervals[i][0];
                        accumulatedFrequency[i][1]=intervals[i][1];
                        if(i!=0)
                                accumulatedFrequency[i][2]=
accumulatedFrequency[i-1][2]+this.getDiscretArr()[i][1];
                        else
                                accumulatedFrequency[i][2]= this.getDiscretArr()[i][1];
                }
                }
                public double[][] getIntervals() {
                        return intervals;
                }
                public void setIntervals(double[][] intervals) {
                        this.intervals = intervals;
                }
                public double getIntervalWidth() {
                        return intervalWidth;
                }
                public void setIntervalWidth(double intervalWidth) {
                        this.intervalWidth = intervalWidth;
                }
                public int getR() {
                        return r;
                }
                public void setR(int r) {
                        this.r = r;
                }
                public ArrayList<Double> getDiscretArrTotal() {
                        return discretArrTotal;
                }
                public void setDiscretArrTotal(ArrayList<Double> discretArrTotal) {
                        this.discretArrTotal = discretArrTotal;
                }
                public double[][] getDiscretArr() {
                        return discretArr;
                }
                public void setDiscretArr(double[][] discretArr) {
                        this.discretArr = discretArr;
                }
                public double getMax() {
                        return max;
                }
                public void setMax(double max) {
                        this.max = max;
                }
                public double getMin() {
                        return min;
                }
                public void setMin(double min) {
                        this.min = min;
                }
        }
```

Середнє статистичне :	0.39000000000000007				
Мода :	0.39				
Медіана :	0.39				
Розмах :	0.78				
Дисперсія :	0.03708342857142856				
Середнє квадратичне відхилення :	0.19257058075269068				
Виправлена дисперсія :	0.03893759999999999				
Варіація :	0.493770719878694				
Початковий момент 2 порядку:	0.1891834285714286				
Центральний момент 2 порядку:	0.03708342857142856				
Асиметрія :	-1.3418350438370155E-15				
Ексцес :	-0.8671875000000004				
Інтервальний ряд :					
Інтервали	[0,000-0,156],	[0,156-0,312],	[0,312-0,468],	[0,468-0,624],	[0,624-0,780],
Z(i)	0,078	0,234	0,390	0,546	0,702
M(i)	3,000	4,000	7,000	4,000	3,000

Рис.1 Результати розрахунків

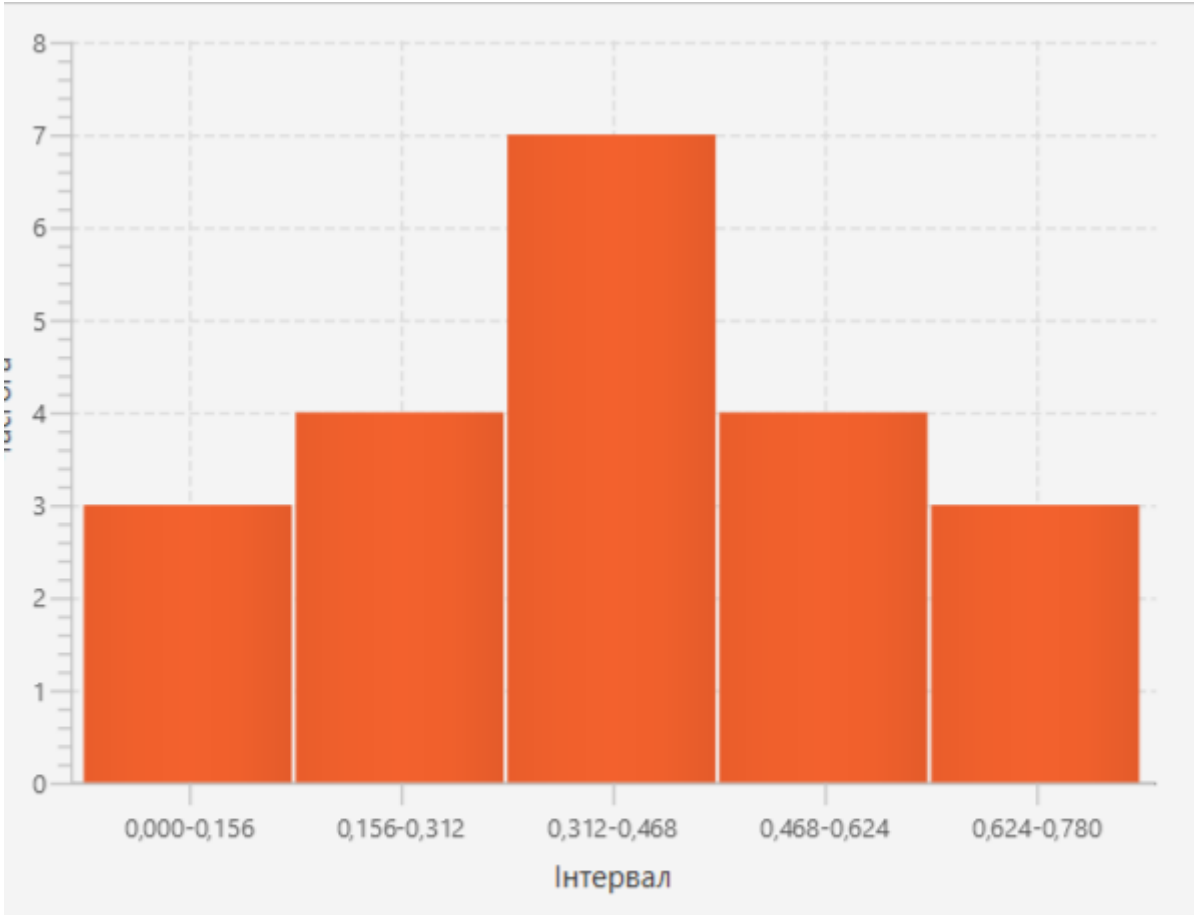

Рис.2 Гістограма за частотами

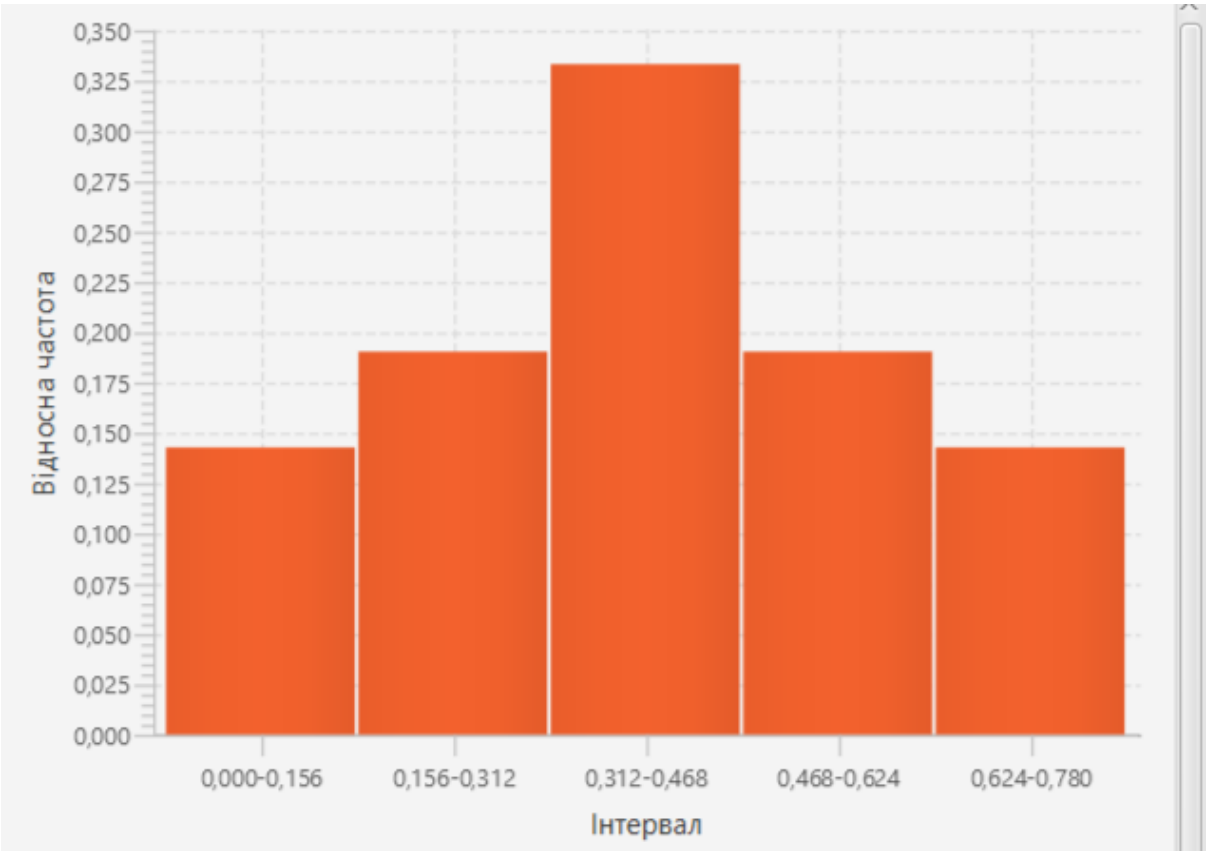

Рис.3 Гістограма за відносними частотами

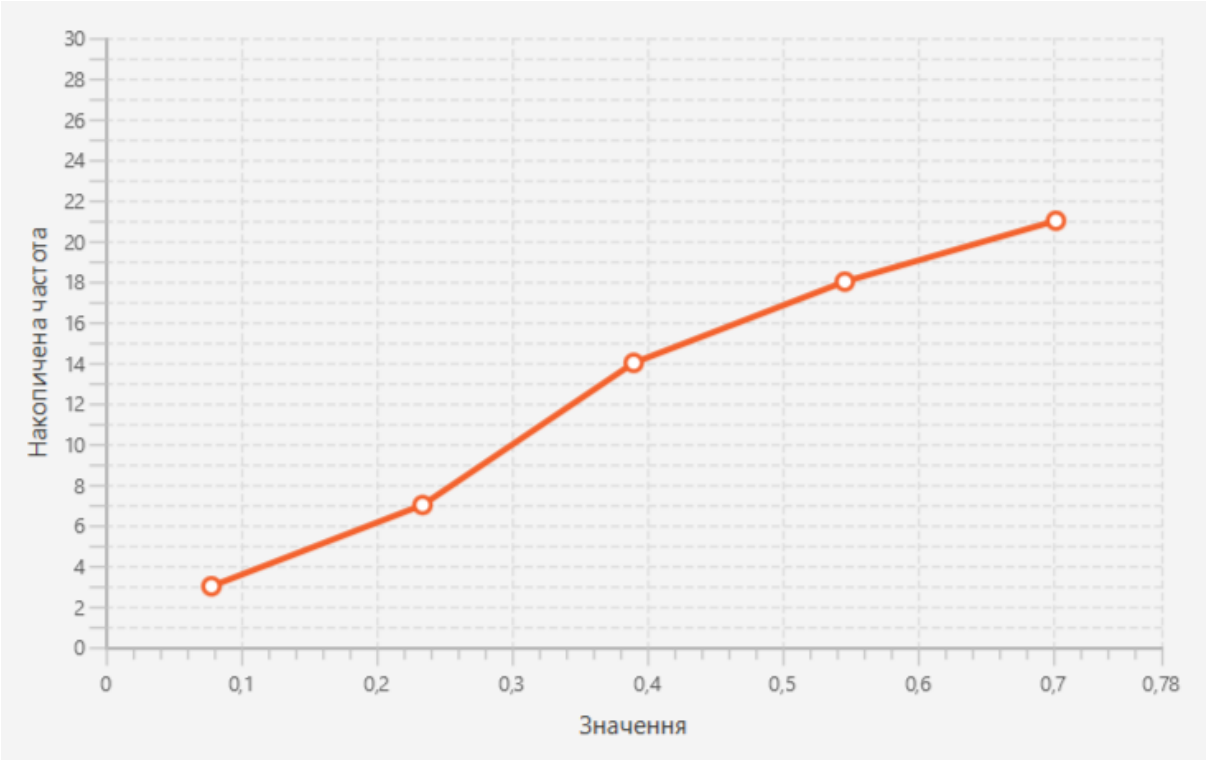

Рис.4 Комулятивна крива частот

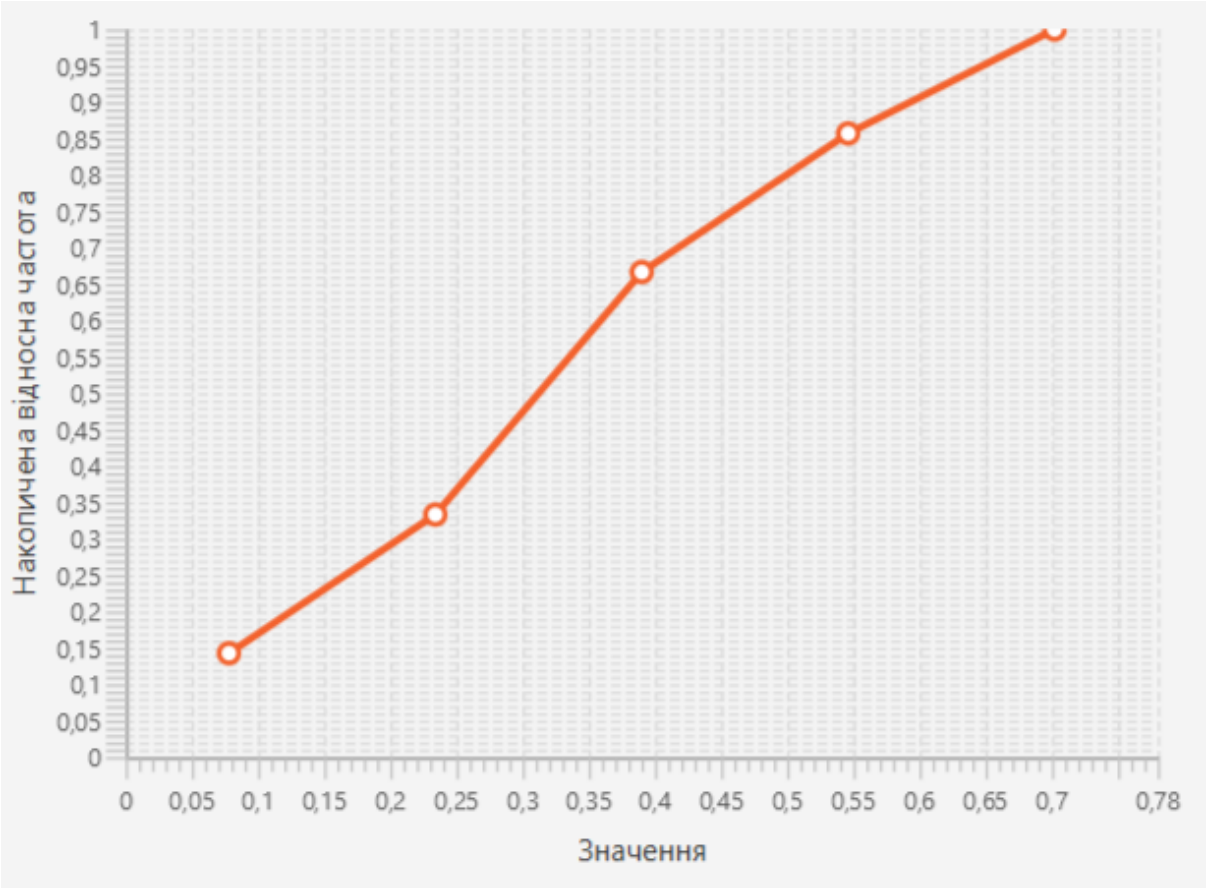

Рис.5 Кумулятивна крива відносних частот

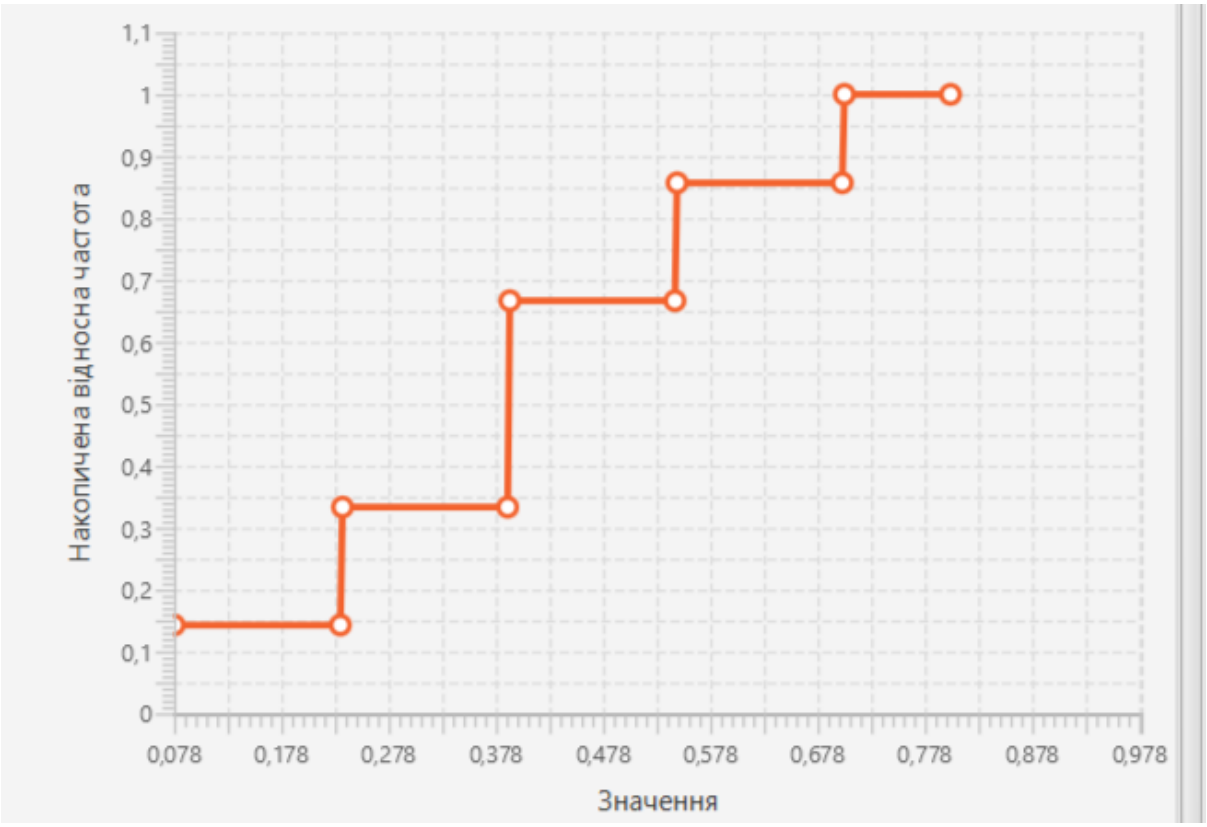

Рис.6 Емпірична функція розподілу

Висновки

У результаті виконання лабораторної роботи розроблено обчислювальний алгоритм для опрацювання інтервального статистичного ряду, реалізований мовою програмування Javay середовищі Eclipse. Результати досліджень свідчать, що крива розподілу зсунута праворуч відносно кривої нормального розподілу (асиметрія є від’ємною) і вона є низьковершинною (ексцес є від’ємним).