Sarah Kirby

Module 2 Assignment

**Program output:**

Average degree: 9.594501718213055

Histogram saved as histogram.png

Probability function saved as probabilityFunction.pn g

P(degree > mean): 0.4192439862542955

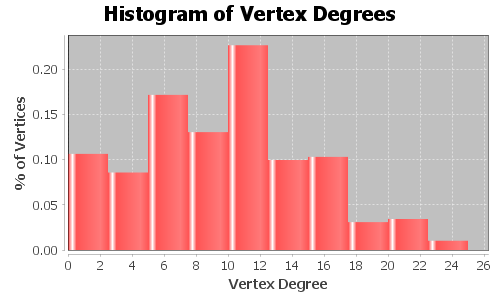


Figure histogram.png

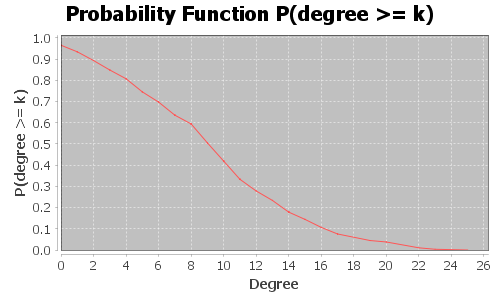


Figure probabilityFunction.png

**Program:**

package assignments;  
  
import graph.GraphUtils;  
import org.apache.commons.math3.random.EmpiricalDistribution;  
import org.apache.tinkerpop.gremlin.structure.Vertex;  
import org.apache.tinkerpop.gremlin.tinkergraph.structure.TinkerGraph;  
import org.jfree.chart.ChartFactory;  
import org.jfree.chart.ChartUtilities;  
import org.jfree.chart.JFreeChart;  
import org.jfree.chart.plot.PlotOrientation;  
import org.jfree.data.statistics.HistogramDataset;  
import org.jfree.data.statistics.HistogramType;  
import org.jfree.data.xy.XYSeries;  
import org.jfree.data.xy.XYSeriesCollection;  
  
import java.awt.\*;  
import java.io.File;  
import java.io.IOException;  
import java.util.Iterator;  
  
public class Module2 {  
  
 private final TinkerGraph graph;  
 private String GRAPH\_FILE = "C:\\Users\\Sarah\\Documents\\JHU EP\\GraphAnalytics\\students.graphml";  
 private final String DEGREE = "degree";  
  
 public Module2() {  
 graph = GraphUtils.*readGraphML*(GRAPH\_FILE);  
  
 double[] degreeData = getDegreeData();  
 EmpiricalDistribution distribution = new EmpiricalDistribution();  
 distribution.load(degreeData);  
  
 double mean = distribution.getNumericalMean();  
 System.*out*.println("Average degree: " + mean);  
  
 createHistogram(degreeData);  
 plotProbabilityFunction(distribution);  
  
 System.*out*.println("P(degree > mean): " +  
 ( 1 - distribution.cumulativeProbability(Math.*round*(mean))) );  
  
 }  
  
 private void plotProbabilityFunction(EmpiricalDistribution dist){  
  
 //create dataset  
 XYSeriesCollection dataset = new XYSeriesCollection();  
 XYSeries series = new XYSeries("P(degree >= k)");  
 for(int i = 0; i <= 100; i++){  
 double x = dist.getNextValue();  
 series.add(x, 1 - dist.cumulativeProbability(x));  
 }  
 dataset.addSeries(series);  
  
 JFreeChart chart = ChartFactory.*createXYLineChart*(  
 "Probability Function P(degree >= k)", "Degree", "P(degree >= k)",  
 dataset, PlotOrientation.*VERTICAL*, false, false, false);  
  
 try {  
 File file = new File("probabilityFunction.png");  
 ChartUtilities.*saveChartAsPNG*(file, chart, 500, 300);  
  
 if(Desktop.*isDesktopSupported*()){  
 Desktop desktop = Desktop.*getDesktop*();  
 if(file.exists()) desktop.open(file);  
 }  
  
 System.*out*.println("Probability function saved as probabilityFunction.png");  
 }  
 catch(IOException e){  
 System.*out*.println("Unable to save chart");  
 }  
 }  
  
 private void createHistogram(double[] data){  
 int binCount = 10;  
 HistogramDataset dataset = new HistogramDataset();  
 dataset.setType(HistogramType.*RELATIVE\_FREQUENCY*);  
 dataset.addSeries("Histogram", data, binCount);  
  
 JFreeChart chart = ChartFactory.*createHistogram*(  
 "Histogram of Vertex Degrees", "Vertex Degree", "% of Vertices",  
 dataset, PlotOrientation.*VERTICAL*, false, false, false);  
 try {  
 File file = new File("histogram.png");  
 ChartUtilities.*saveChartAsPNG*(file, chart, 500, 300);  
  
 if(Desktop.*isDesktopSupported*()){  
 Desktop desktop = Desktop.*getDesktop*();  
 if(file.exists()) desktop.open(file);  
 }  
  
 System.*out*.println("Histogram saved as histogram.png");  
 }  
 catch(IOException e){  
 System.*out*.println("Unable to save chart");  
 }  
 }  
  
 private double[] getDegreeData()  
 {  
 double[] data = new double[countVertices()];  
 graph.variables().get(DEGREE);  
 Iterator<Vertex> it = graph.vertices();  
  
 int i = 0;  
 while(it.hasNext())  
 {  
 Vertex v = it.next();  
 data[i] = new Double((long)v.property(DEGREE).value()).doubleValue();  
 i++;  
 }  
 return data;  
 }  
  
 private int countVertices()  
 {  
 int count = 0;  
 Iterator<Vertex> it = graph.vertices();  
 while(it.hasNext())  
 {  
 Vertex v = it.next();  
 count++;  
 }  
 return count;  
 }  
  
 public static void main(String[] args)  
 {  
 new Module2();  
 }  
}