Team#: 02
School Name: Albuquerque Academy
Area of Science: Environmental Science and health
Project Title: Exploring Pollution Borne Illness

Exploring Pollution Borne Illness

Interim Report By:
Calvin Stewart
Wesley Catbagan
Colin Finnegan
Suraj Kholwadwala
Raffy Schleder

Executive Summary

We have determined that the best way to stop or at least decrease the amount of pollution related diseases is to spread awareness about it through computational analysis. We believe that the model of New York would be valuable because it would show the real-world risk in polluting New York. Because you can model the city you would be able to figure out what sections of the city are at greater risk of succumbing to certain illnesses, and with that knowledge you would be able to take precautionary measures that could save lives. This project is a continuation of our project last year, and with additional time we are implementing a few new things. Among these are the creation of a heatmap to allow for easier interpretation of data, as well as a template system in which you can implement certain geographic features like wind in order to allow our code to be used in any city. To show that the template will work, we are extending our code to the southwest's larger cities

Definition of Problem

Our team has chosen to work with the epidemiology of pollution borne illness across the country. In recent years, with growing pollution, the rise of these diseases can be tied directly back to certain pollutants present in the environment. By using multiple studies we will be able to create a template that could be used by civil engineers to show which disease are being caused by what pollution and from where. We will then use this template to create models of a few cities around the United States as a proof of concept. We will also work to create a friendly user environment and a heat map for easy interfacing.

Plan to Solve This Problem

We have determined that the best way us to solve this problem is by working from the cases of disease backwards to the origin of pollution. This will allow us to see which outbreaks are caused by which areas of pollution. Our given information will be the type of disease, location of the outbreak, wind and terrain of the surrounding area, and origins of all pollutants. Using public data we will be able to fill out all of these which will allow us to determine our dependent variable which is the specific origin of the particulate matter responsible for the outbreak of disease. We will then model the area in question, weather, and pollutant creators. We will then overlay a heat map of the pollution overtime. This will then allow us to effectively look back and see the origin of the pollutants that caused this to occur. We will use an algorithm of our own creation to calculate the movement of particulate matter through the atmosphere. This then shows us the origin of the pollutants that cause outbreaks of disease.

Team Roles

Calvin Stewart:

Calvin Stewart is the team coordinator as well the pollution researcher. Calvin is primarily focused on coordinating and organizing the team as well as writing the reports. He is taking a backseat role in the creation of the core of the coding, but will still be heavily involved with the write up of the explanation of said code.

Colin Finnegan:

Colin Finnegan is the disease data/researcher and code editor. He has found numerous primary and secondary sources that show clearly the correlation between diseases and pollutants. He will be helping Raffy with the editing and debugging of the codes, as well as serving as the primary data analyst.

Wesley Catbagan:

Wesley Catbagan is a writer and the head of design. He works with Calvin on writing the reports and brainstorming for the project. Also, he will help Colin in finding and analyzing the data as well as work on directing the work on the posterboard.

Raffy Schleder:

Raffy is the primary coder and will be, aside from working some on the initial brainstorming of the project, exclusively working on the production of the code. He will be assisted by some other team members, but he will bear the brunt of the coding effort and the creation of the heat map.

Suraj Kholwadwala:

Suraj is an assistant coder to raffy and is really a jack of all trades in our group. He will help Raffy with coding on the heat map because he has taken AP Java more recently than Raffy, and has slightly more experience with graphics. Although he might not take as heavy a role in writing as some of the other group members, his ideas are important for the brainstorming process as well as the analysis of data.

Anticipated Results

We anticipate a detailed and representative heatmap proving our hypothesis and demonstrating that there is a correlation between pollution and disease. Also we expect that our template program will work, and will be useful for civil engineers in determining the best places to emphasize pollution control.

Progress

We are on the brink of success, our code is near perfect, we simply need to add a heat map, and debug, and we will be finished with our program. We have blocked out 6 more meetings in order to finish all of our code, presentations, and our final poster. We intend to continue working together as we see that our great teamwork makes us more productive when are in our group.

This is our code so far:

```
public class Arrays
{
	static int size = 25; //35000

	public void NewYork(int Array[][])
	{
		for (int i = 0; i <size; i ++)
		{
			for (int j = 0; j <size;j ++)
			{
				Array[i][j]= map(i,j);
```

```
                }
            }
        }
        private int map(int i, int j)
        {
            if (i<.3*size&&j>.3*size)
                return 9;
            else if (i>.3*size&&i<.5*size&&j<.1*size)
                return 9;
            else if (i>.6*size&&j>.8*size)
                return 9;
            else if (i>.7*size&&j<.3*size)
                return 9;
            else
                return 0;
        }
}
public class Calculate
{
        double pi = 3.14159;
        public void InMCM(int Array[][], int num, int type)
        {
            int length = Array.length;
            for (int i = 0; i< num; i++)
            {
                int x = (int) ((length-1)*Math.random());
                int y = (int) ((length-1)*Math.random());
                if (Array[x][y]!=0)
                {
                    x = (int) ((length-1)*Math.random());
                    y = (int) ((length-1)*Math.random());
                }
                Array[x][y]=type;
            }
        }
        private int InArray(int length)
        {
            int a = (int) ((length/2+1)*Math.random());
            int b = (int) ((length/2+1)*Math.random());
            return a+b;
        }
        public double OutMCM(int Array[][],int ArrayAfter[][])
        {
            for (int i = 0; i < Array.length; i++)
            {
                for (int j = 0; j < Array.length; j++)
                {
                    if (Array[i][j]!=10 && Array[i][j]!=0)
                    {
                        int day = (int) (366*Math.random());
                        int x=0;
                        int y=0;
                        for (int a = day; a<day+randTime(); a++)
                        {
                            double speed = speed(a);
                            double direction = direction(a);
```

```
                                    int tempx = (int)
(speed*Math.acos(direction));
                                    int tempy = (int)
(speed*Math.asin(direction));
                                    x = tempx + x;
                                    y = tempy + y;
                              }
                              ArrayAfter[i+x][j+y]=
Array[i][j];
                        }
                  }
            }
            return 0;

      }
      public int randTime()
      {
            double height = 100*Math.random();//11001
            double FallSpeed = Math.random();
            double time = height/FallSpeed;
            time = time/60;
            time = time/24;
            return (int)time;
      }
      public double speed(int x)
      {
      double speed = (Math.sin(x*pi/120))*(.5*Math.sin(x*pi/2)+1);
      speed = speed*(1609/60);
      return speed/50;//(for mini model)
      }
      public double direction(int x)
      {
      double direction = (6.5*Math.sin(x*2*pi/5))*(Math.PI/8);
      return direction;
      }

}
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.GraphicsConfiguration;
import java.awt.GraphicsDevice;
import java.awt.GraphicsEnvironment;
import java.awt.LinearGradientPaint;
import java.awt.MultipleGradientPaint;
import java.awt.Point;
import java.awt.RadialGradientPaint;
import java.awt.Transparency;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.image.BufferedImage;
import java.awt.image.ByteLookupTable;
import java.awt.image.LookupOp;
import java.awt.image.LookupTable;
import java.awt.image.Raster;
import java.io.IOException;
```

```
import javax.imageio.ImageIO;
import javax.swing.JFrame;
import javax.swing.JPanel;
import org.jdesktop.swingx.graphics.BlendComposite;
public class Heatmap extends JPanel implements MouseListener {
 private static final long serialVersionUID = -2105845119293049049L;
 private final BufferedImage backgroundImage;
 private final BufferedImage dotImage = createFadedCircleImage(96);
 private BufferedImage monochromeImage;
 private BufferedImage heatmapImage;
 private LookupOp colorOp;
 public Heatmap(BufferedImage backgroundImage) {
      this.backgroundImage = backgroundImage;
      int width = backgroundImage.getWidth();
      int height = backgroundImage.getHeight();
      final BufferedImage colorImage =
      createGradientImage(new Dimension(64, 1), Color.WHITE, Color.RED,
Color.YELLOW,
            Color.GREEN.darker(), Color.CYAN, Color.BLUE, new Color(0, 0,
0x33));
      final LookupTable colorTable = createColorLookupTable(colorImage, .5f);
      colorOp = new LookupOp(colorTable, null);
      monochromeImage = createCompatibleTranslucentImage(width, height);
      Graphics g = monochromeImage.getGraphics();
      g.setColor(Color.WHITE);
      g.fillRect(0, 0, width, height);
      setPreferredSize(new Dimension(width, height));
      addMouseListener(this);
 }
 public BufferedImage colorize(LookupOp colorOp) {
      return colorOp.filter(monochromeImage, null);
 }
 public BufferedImage colorize(LookupTable colorTable) {
      return colorize(new LookupOp(colorTable, null));
 }
 public void paintComponent(Graphics g) {
      super.paintComponent(g);
      heatmapImage = colorize(colorOp);
      g.drawImage(backgroundImage, 0, 0, this);
      g.drawImage(heatmapImage, 0, 0, this);
 }
 public void mouseClicked(MouseEvent e) {
      addDotImage(e.getPoint(), .75f);
      repaint();
 }
 private void addDotImage(Point p, float alpha) {
      int circleRadius = dotImage.getWidth() / 2;
      Graphics2D g = (Graphics2D) monochromeImage.getGraphics();
      g.setComposite(BlendComposite.Multiply.derive(alpha));
      g.drawImage(dotImage, null, p.x - circleRadius, p.y - circleRadius);
 }
 public void mousePressed(MouseEvent e) {
 }
 public void mouseReleased(MouseEvent e) {
 }
 public void mouseEntered(MouseEvent e) {
 }
```

```
 public void mouseExited(MouseEvent e) {
 }
 public static LookupTable createColorLookupTable(BufferedImage im, float
alpha) {
      int tableSize = 256;
      Raster imageRaster = im.getData();
      double sampleStep = 1d * im.getWidth() / tableSize;   byte[][]
colorTable = new byte[4][tableSize];
      int[] pixel = new int[1];
      Color c;
      for (int i = 0; i < tableSize; ++i) {
      imageRaster.getDataElements((int) (i * sampleStep), 0, pixel);
      c = new Color(pixel[0]);
      colorTable[0][i] = (byte) c.getRed();
      colorTable[1][i] = (byte) c.getGreen();
      colorTable[2][i] = (byte) c.getBlue();
      colorTable[3][i] = (byte) (Math.max(0, Math.min(1, alpha)) * 0xff);
      }
      LookupTable lookupTable = new ByteLookupTable(0, colorTable);
      return lookupTable;
 }
 public static BufferedImage createGradientImage(Dimension size, Color...
colors) {
      float[] fractions = new float[colors.length];
      float step = 1f / colors.length;
      for (int i = 0; i < colors.length; i++) {
      fractions[i] = i * step;
      }
      LinearGradientPaint gradient =
      new LinearGradientPaint(0, 0, size.width, 1, fractions, colors,
            MultipleGradientPaint.CycleMethod.REPEAT);
      BufferedImage im = createCompatibleTranslucentImage(size.width,
size.height);
      Graphics2D g = im.createGraphics();
      g.setPaint(gradient);
      g.fillRect(0, 0, size.width, size.height);
      g.dispose();
      return im;
 }
 public static BufferedImage createCompatibleTranslucentImage(int width, int
height) {
      GraphicsEnvironment env =
GraphicsEnvironment.getLocalGraphicsEnvironment();
      GraphicsDevice dev = env.getDefaultScreenDevice();
      GraphicsConfiguration conf = dev.getDefaultConfiguration();
      return conf.createCompatibleImage(width, height,
Transparency.TRANSLUCENT);
 }
 public static BufferedImage createFadedCircleImage(int size) {
      float radius = size / 2f;
      RadialGradientPaint gradient =
      new RadialGradientPaint(radius, radius, radius, new float[] {0f, 1f},
new Color[] {
            Color.BLACK, new Color(0xffffffff, true)});
      BufferedImage im = createCompatibleTranslucentImage(size, size);
      Graphics2D g = (Graphics2D) im.getGraphics();
      g.setPaint(gradient);
```

```
        g.fillRect(0, 0, size, size);
        g.dispose();
        return im;
    }
    public static void main(String... args) throws IOException {
        BufferedImage backgroundImage =
ImageIO.read(Heatmap.class.getResource("map.png"));
        JPanel comp = new Heatmap(backgroundImage);
        JFrame frame = new JFrame("Heatmap");
        frame.add(comp);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setResizable(false);
        frame.pack();
        frame.setVisible(true);
    }
}
```


Acknowledgment

James Mims: head of the computer science department at Academy

1. http://circ.ahajournals.org/content/109/1/71.short (An entry by the American heart Association)
2. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1240667/ (A dissertation by the enviromental health prospective on the correlation between pollution and disease)
3. http://www.atsjournals.org/doi/abs/10.1164/rccm.200701-036OC (A Scholarly Article that correlates car emmisions with disease)
4. http://www.atsjournals.org/doi/abs/10.1164/ajrccm/151.3_Pt_1.669#.Vmd3__krKhc (An article working with the mortality rates in correlation to pollution)
5. http://www.nytimes.com/2013/04/02/world/asia/air-pollution-linked-to-1-2-million-deaths-in-china.html?_r=0

http://www.supercomputingchallenge.org/15-16/final-reports/
http://www.eoearth.org/view/article/150296/
http://www.nyc.gov/html/doh/downloads/pdf/eode/eode-air-quality-impact.pdf