50 years (1951-2000) of Mumbai Weather data

50 years of weather data recorded at Santa Cruz, Mumbai is used to create thisvisualization.

Under the Guidance of Prof. Venkatesh Rajamanickam

Baisampayan Saha
136130004
IDC, IIT Bombay

A
Data Art
Project

Acknowledgement

I would like to thank my guide ***Prof. Venkatesh Rajamanickam*** for proposing this topic as it was challenging as well as stimulating. I would like to thank him for supporting me and having patience in me as I was doing my first data art project.

I would like to thank ***Rasagy Sharma*** in helping me out in bits and pieces in coding when I would get struck.

I would also like to thank ***Gauri*** and ***Kavita*** for their invaluable suggestions. And lastly I would like to thank all my friends who helped me out by giving suggestions in improving the project at various phases of the project.

Avowal

I asseverate, that the present work has been produced without the help of third party and only with the denoted references. All parts that have been taken from sources are indicated. This work has never been presented to an examination board.

Name:

Roll Number:

Date:

Signature:

Approval

Design Research Seminar

"50 years of Mumbai Weather" - By Baisampayan Saha

M.Des, Industrial Design Batch 2013-15, is approved as a partial fulfillment of requirement of Post Graduate degree in Industrial Design.

Prof. Venkatesh Rajamanickam

Abstract

When a large data set is presented in front of us, it does not make any sense to us until and unless we start doing number crunching. If the data is represented in some form of charts or diagrams than probably we can figure out something. Though the project "50 years of Mumbai weather" is not about data visualization, but it deals with the notion of data art. The weather data is taken up and woven into a fabric of a narration that for each viewer would give an understanding of its own. Each viewer could come to an altogether different conclusion of the diagrams shown in front of them and their conclusions be equally correct as the underlying principles of generating each diagram from the data set are kept exactly the same.

An attempt is made to generate such diagrams from the weather data of Mumbai for each month of the year. When someone would view all the diagrams together a form of a story would emerge out from the data embedded inside the diagrams.

Index

List of Figures

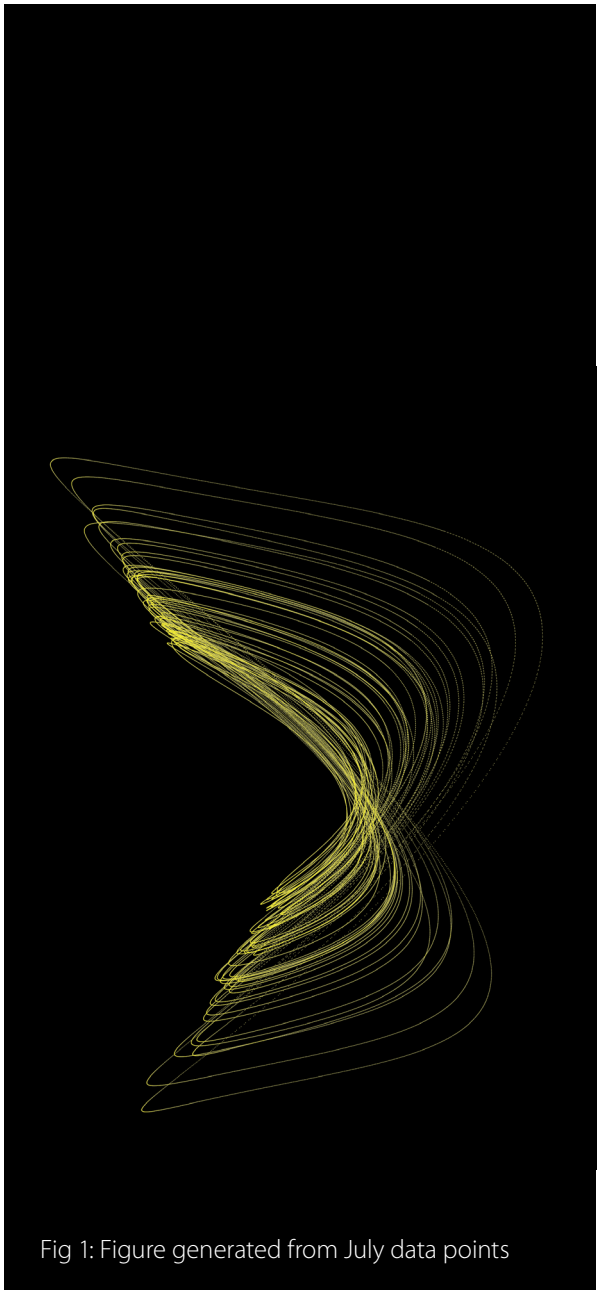


Fig 1: Figure generated from July data points

Introduction

50 years of data of Mumbai weather is first taken from the website of Indian Meteorological Department website ***(http://www.imd.gov.in/section/nhac/mean/Mumbai.htm)*** The data is sorted out corresponding to each month of the year, i.e. from January to December.

For depicting the data into various diagrams or charts, an open-source software application called ***Processing*** is used. Processing is a Java based programming application, which was developed mainly for creating data visualizations and data art and targeted from designers to engineers to graphic artists alike.

The data from the website is sorted in the form of maximum average temperature, minimum average temperature and average rainfall for each both for a period from 1951 - 2000. The three data points are taken as coordinates of a point in 3D space. All these points generated by these coordinate points are plotted in a 3D space and these plotted points are revolved in X-axis and Y-axis of the 3D space and their rotating paths are traced to finally generate the diagrams for each month.

Fig 1 is an example of diagram generated from July data set.

Literature Review

Data art was a completely new domain of work for me. There are various ways of generating data art. Many open-source tools can be found for generating it. Some needs programming knowledge and some does not required any programming knowledge at all. So, the first thing that I did after beginning the project is to search a tool that is flexible enough to create generative data art and also the learning curve of it is pretty easy. I found out there are various such tools for it. Most of them are free and open-source. Few examples are Processing, Java-script, NodeBox, vvvv, OpenFrameworks, etc. I choose Processing as the amount of on-line learning resources of it are too exhaustive and also i found it little it easier to grasp then other tools of the trade. So, I started looking into examples given into processing website. This helped a lot in understanding the basics of it. Then I got a book from Prof. Venkatesh called "Getting started with processing by Casey Reas and Ben Fry". This actually helped me in understanding the tool a lot better. After reading the book, I started looking into on-line video tutorials that were there in https://www.youtube.com, https://vimeo.com and various other online video sites. The url link from vimeo website is "http://vimeo.com/album/2573675". This has an complete tutorial of the basic understanding of processing and also has some advanced stuff in it. The website "http://funprogramming.org" also provided to see how other artist used processing as a tool

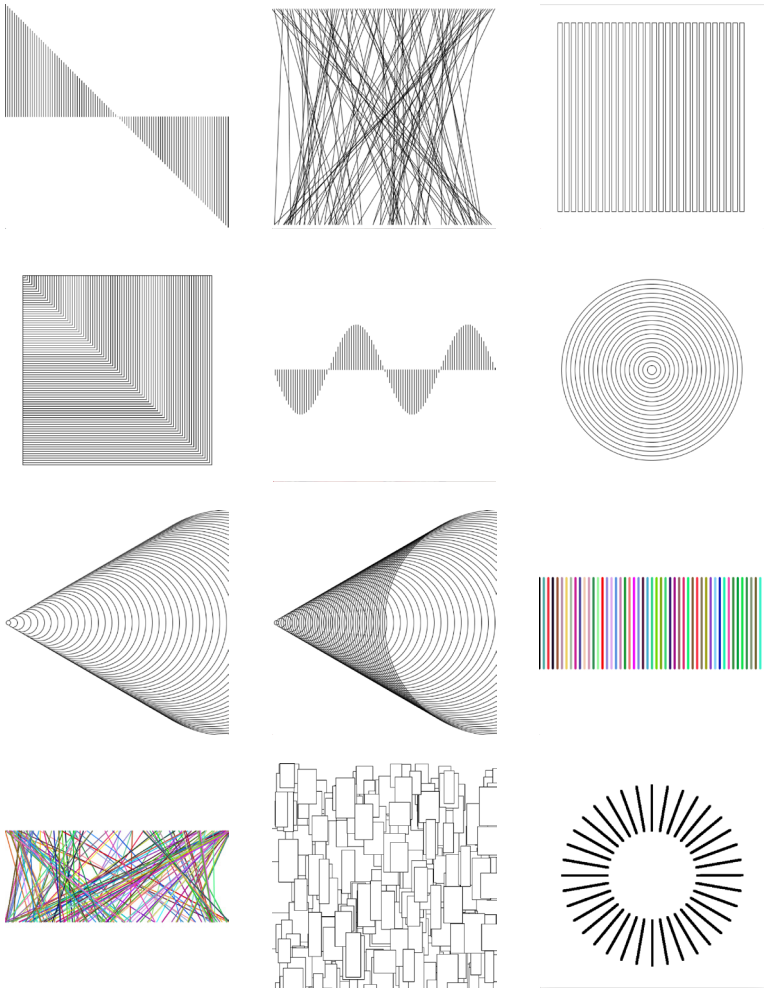


Fig 2: Examples of generative art using processing from various books

for generating art. Then the website "http://fyprocessing.tumblr.com/page/12 "opened up a new dimension for me that showed how processing can be used to do some of the most mind-boggling data art and animations.

I got hold of a book called "Algorithms for Visual Design - Using Processing language by Kostas Terzidis" and another book called "Generatice Art" that I had found in the website "http://www.scribd.com/doc/239849789/Generative-Art". I started practicing the examples that were given in the book and started to solve the exercise problems given after the end of each chapter.

Fig 2 is an example of few of the solved exercise problems.

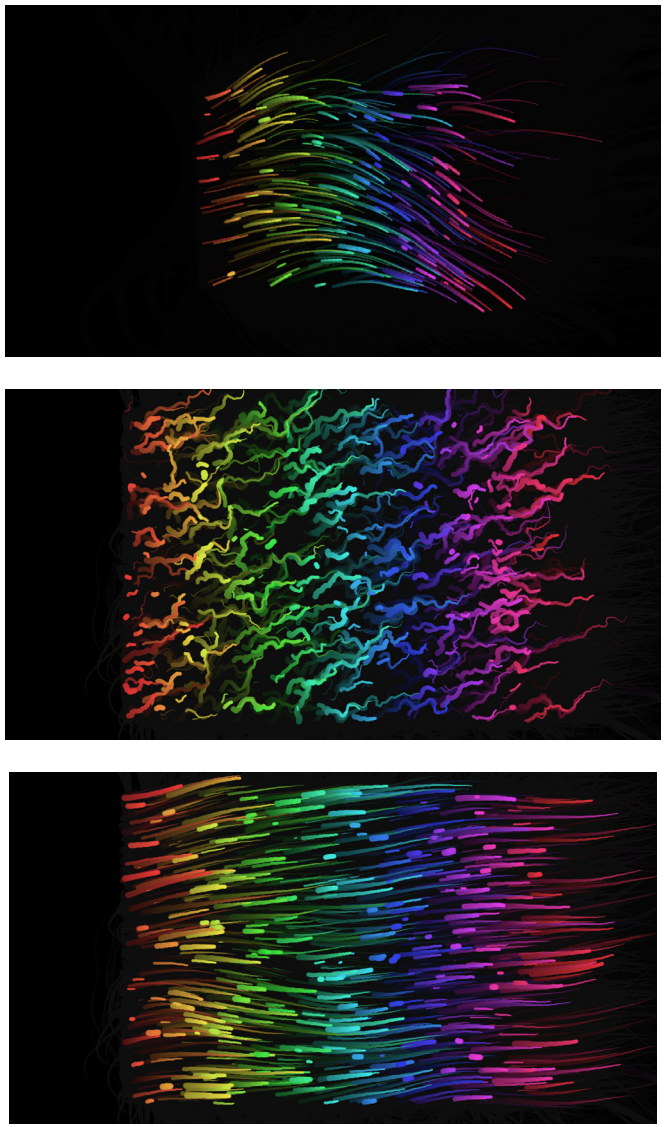


Fig 3: Creation of Noise field by Perlin Noise by Felix Turner (airtightinteractive.com). The bottom two pictures are modifications of his code.

After spending some considerate in learning the basics of processing, I tried to see if i can modify codes of existing generative art that can are found in the web and are given the permission to modify the codes given the actual artist contribution is also acknowledged. In one such example was the generative art shown in Fig 3. This is an example of a noise field created by using Perlin Noise by artist Felix Turner. I tried to modify the code and make the size and shape of the noise field more random than what he had created. The original one was much smaller in size and the speed of the particles were much faster then the modified one. The last two pictures are the modified versions of the code, where I had increased the size of the canvas, increased the randomness of the particles and also increased the size of the particles.

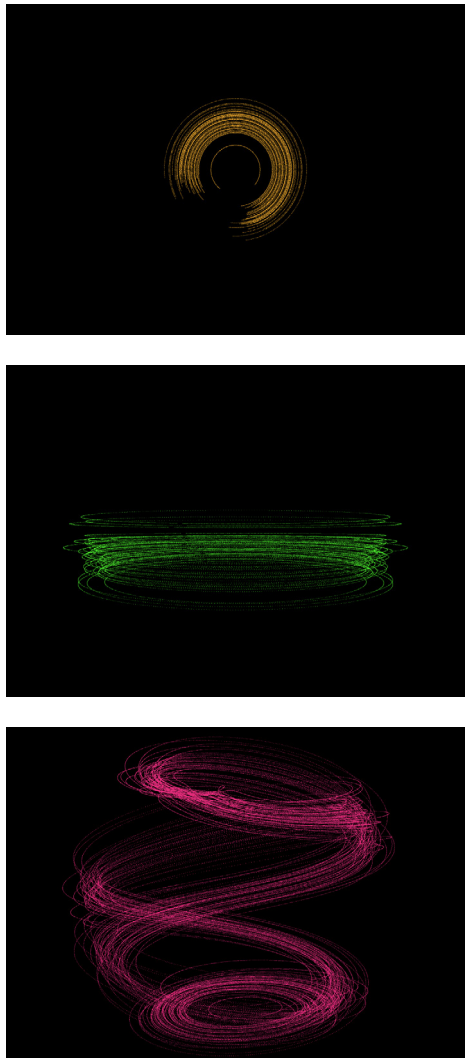


Fig 4: First attempt of plotting the weather data

50 years of Mumbai Weather

After getting a fair bit of confidence in Processing, I started to look into the data that we have got from Indian Meteorological Department website. The data was of 50 years of information about Mumbai weather collected at Santa Cruz weather station. The data comprised of mainly 3 variables: average maximum temperature, average minimum temperature and average rainfall. After thinking for sometime how can these data be presented into a form of generative art, one idea struck in. As there are three variables, these can be taken as a coordinates for a point in 3d space. So if we plot these coordinates we would get points that would correspond to each months data for a period of fifty years. Now if we can rotate them in an axis, then these points would generate patterns that would be different for each month and also the nature of the patterns would be governed by the same programming logic and the data set collected from the website. So we would get a correct scientific model of weather data for 50 years that can be represented by generative art. Fig 4 shows some of the first attempt in plotting the weather data.

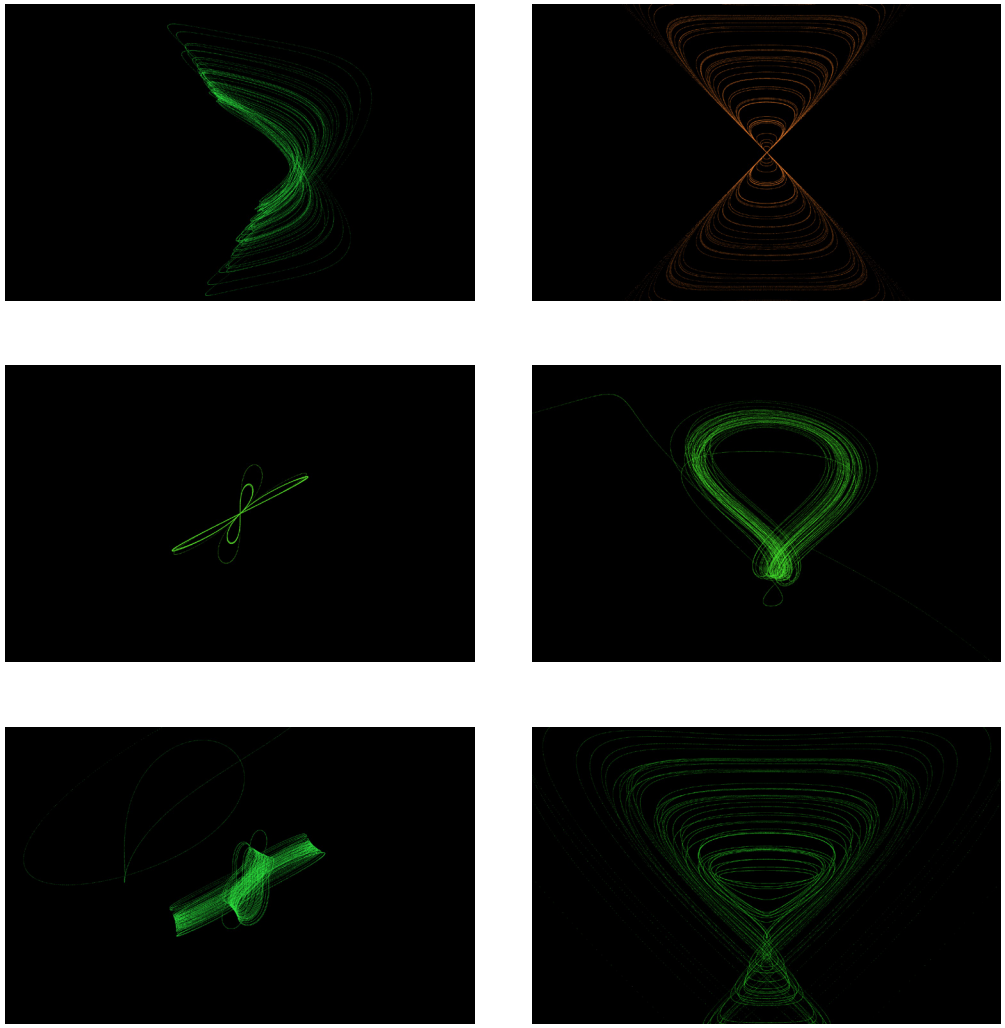


Fig 5: Modifying the code further to include trigonometric functions to ahieve more forms

After plotting the points in the 3D space, the points were rotated in X, Y, Z axis at different times to see what sort of patterns they create. The pattern is mainly created as the path of rotation is traced. So every time a point moves it leaves behind a trail of the path it is moving, thus creating a pattern. The rotation of the points are sometimes put on one single axis or sometimes clubbed together to either 2 axis or all the 3 axis.

All possible formats of generating the patterns were checked. Some patterns were created where the rotation of the particles around an axis would depend on a sine or cosine function. The patterns thus created would be very different than normal rotation of these points in the axis. Fig 5 shows some of the patterns created by using trigonometric functions as well as multi axis rotation.

So finally after many hit and run trials so see which combination of codes would generate the best patterns, I decided upon not to include any trigonometric functions but to simply rotate the point in X-axis and Y-axis at the same time. So this resulted in the final output of the pattern.

But when the pattern were showed to people in general, they were not being able to underdtand or decifer anything. So I thought another piece of information about the pattern should be included. So for this, I created a graph, that had all the three variables in it, i.e. average maximum temperature, average minimum temperature and average rainfall. All the variables are put in one single graph and their maximum and minimum values were mapped to the width and height of the canvas of the graph. These created a graph where the nature of the temperature and rainfall for a certain month can be easily seen. When this graph was juxtaposed with the generated pattern, the pattern made much more sense now and also when all the patterns and the graphs were shown, a story of Mumbai weather seemed to come out. The codes of both the pattern and the graph can be found in github website "https://github.com/baisampayans/consolidated_v2_alternate_colors" with the data set used to create it.

The codes for both the programs can be found in the next page. Some of the codes are hashed out as comments. These hashed out codes can be used to modify the codes again.

```
import processing.pdf.*;

//import peasy.*;

//PeasyCam jCam;

Table table_jan;
Table table_feb;
Table table_mar;
Table table_apr;
Table table_may;
Table table_jun;
Table table_jul;
Table table_aug;
Table table_sep;
Table table_oct;
Table table_nov;
Table table_dec;


int amount = 50;
```

```
float[] x_jan = new float[amount];
float[] y_jan = new float[amount];
float[] z_jan = new float[amount];
float[] n_jan = new float[amount];

float[] x_feb = new float[amount];
float[] y_feb = new float[amount];
float[] z_feb = new float[amount];
float[] n_feb = new float[amount];

float[] x_apr = new float[amount];
float[] y_apr = new float[amount];
float[] z_apr = new float[amount];
float[] n_apr = new float[amount];

float[] x_mar = new float[amount];
float[] y_mar = new float[amount];
float[] z_mar = new float[amount];
float[] n_mar = new float[amount];

float[] x_may = new float[amount];
```

```
float[] y_may = new float[amount];
float[] z_may = new float[amount];
float[] n_may = new float[amount];

float[] x_jun = new float[amount];
float[] y_jun = new float[amount];
float[] z_jun = new float[amount];
float[] n_jun = new float[amount];

float[] x_jul = new float[amount];
float[] y_jul = new float[amount];
float[] z_jul = new float[amount];
float[] n_jul = new float[amount];

float[] x_aug = new float[amount];
float[] y_aug = new float[amount];
float[] z_aug = new float[amount];
float[] n_aug = new float[amount];

float[] x_sep = new float[amount];
float[] y_sep = new float[amount];
```

```
float[] z_sep = new float[amount];
float[] n_sep = new float[amount];

float[] x_oct = new float[amount];
float[] y_oct = new float[amount];
float[] z_oct = new float[amount];
float[] n_oct = new float[amount];

float[] x_nov = new float[amount];
float[] y_nov = new float[amount];
float[] z_nov = new float[amount];
float[] n_nov = new float[amount];

float[] x_dec = new float[amount];
float[] y_dec = new float[amount];
float[] z_dec = new float[amount];
float[] n_dec = new float[amount];

int currentMonth=0;

int stageX = 400;
```

```
int stageY = 300;
int stageZ = 150;

PFont font1;
PFont font2;


void setup() {
 size(1900, 1200, OPENGL);

 beginRaw(PDF, "x-###.pdf");

 //*
 //font1 = loadFont("HelveticaNeue-Light-48.vlw");
// font2 = loadFont("HelveticaNeue-Bold-48.vlw"); //*/

 font1 = createFont("HelveticaNeue-Bold", 48);
 font2 = createFont("HelveticaNeue-Bold", 48);

 // for camera control with mouse
 //jCam = new PeasyCam(this, 500);
```

```
int a = int (random(0, width));
int b = int (random(0, height));
//int c = int (random(255));
float sColor = map(a, 0, width, 0, 100);

//colorMode(HSB, 100);
background(0);
noFill();
//stroke(sColor,80,90);
strokeWeight(1);
smooth();
//  frameRate(14);


stroke(255, 203, 30);
line(300, (height/2-90), 300, (height/2+80));


loadData();
writeText();
```

```
 // for randomly creating points in 3D space
  //  for(int i = 0; i<amount; i++) {
  //    x[i] = int(random(-540, 540));
  //    y[i] = int(random(-390, 390));
  //    z[i] = int(random(-300, 300));
}



void loadData() {
  // Load CSV file into a Table object
  // "header" option indicates the file has a header row
  table_jan = loadTable("points_jan.csv", "header");

  // You can access iterate over all the rows in a table
  int rowCount = 0;
  for (TableRow row : table_jan.rows ()) {
    // You can access the fields via their column name (or index)
    x_jan[rowCount] = row.getFloat("Max_temp");
    y_jan[rowCount] = row.getFloat("Min_temp");
    z_jan[rowCount] = row.getFloat("Rainfall");
```

```
n_jan[rowCount] = row.getFloat("Year_Jan");
  rowCount++;
 }

 table_feb = loadTable("points_feb.csv", "header");

 // You can access iterate over all the rows in a table
 rowCount = 0;
 for (TableRow row : table_feb.rows ()) {
  // You can access the fields via their column name (or index)
  x_feb[rowCount] = row.getFloat("Max_temp");
  y_feb[rowCount] = row.getFloat("Min_temp");
  z_feb[rowCount] = row.getFloat("Rainfall");
  n_feb[rowCount] = row.getFloat("Year_Feb");
  rowCount++;
 }
 table_mar = loadTable("points_march.csv", "header");

 // You can access iterate over all the rows in a table
 rowCount = 0;
 for (TableRow row : table_mar.rows ()) {
```

```
// You can access the fields via their column name (or index)
  x_mar[rowCount] = row.getFloat("Max_temp");
  y_mar[rowCount] = row.getFloat("Min_temp");
  z_mar[rowCount] = row.getFloat("Rainfall");
  n_mar[rowCount] = row.getFloat("Year_March");
  rowCount++;
}
table_apr = loadTable("points_april.csv", "header");

// You can access iterate over all the rows in a table
rowCount = 0;
for (TableRow row : table_apr.rows ()) {
  // You can access the fields via their column name (or index)
  x_apr[rowCount] = row.getFloat("Max_temp");
  y_apr[rowCount] = row.getFloat("Min_temp");
  z_apr[rowCount] = row.getFloat("Rainfall");
  n_apr[rowCount] = row.getFloat("Year_April");
  rowCount++;
}
table_may = loadTable("points_may.csv", "header");
```

```
// You can access iterate over all the rows in a table
 rowCount = 0;
 for (TableRow row : table_may.rows ()) {
  // You can access the fields via their column name (or index)
  x_may[rowCount] = row.getFloat("Max_temp");
  y_may[rowCount] = row.getFloat("Min_temp");
  z_may[rowCount] = row.getFloat("Rainfall");
  n_may[rowCount] = row.getFloat("Year_May");
  rowCount++;
 }
 table_jun = loadTable("points_june.csv", "header");

 // You can access iterate over all the rows in a table
 rowCount = 0;
 for (TableRow row : table_jun.rows ()) {
  // You can access the fields via their column name (or index)
  x_jun[rowCount] = row.getFloat("Max_temp");
  y_jun[rowCount] = row.getFloat("Min_temp");
  z_jun[rowCount] = row.getFloat("Rainfall");
  n_jun[rowCount] = row.getFloat("Year_June");
  rowCount++;
```

```
}
 table_jul = loadTable("points_july.csv", "header");

 // You can access iterate over all the rows in a table
 rowCount = 0;
 for (TableRow row : table_jul.rows ()) {
  // You can access the fields via their column name (or index)
  x_jul[rowCount] = row.getFloat("Max_temp");
  y_jul[rowCount] = row.getFloat("Min_temp");
  z_jul[rowCount] = row.getFloat("Rainfall");
  n_jul[rowCount] = row.getFloat("Year_July");
  rowCount++;
 }
 table_aug = loadTable("points_aug.csv", "header");

 // You can access iterate over all the rows in a table
 rowCount = 0;
 for (TableRow row : table_aug.rows ()) {
  // You can access the fields via their column name (or index)
  x_aug[rowCount] = row.getFloat("Max_temp");
  y_aug[rowCount] = row.getFloat("Min_temp");
```

```
z_aug[rowCount] = row.getFloat("Rainfall");
  n_aug[rowCount] = row.getFloat("Year_August");
  rowCount++;
}
table_sep = loadTable("points_sep.csv", "header");

// You can access iterate over all the rows in a table
rowCount = 0;
for (TableRow row : table_sep.rows ()) {
  // You can access the fields via their column name (or index)
  x_sep[rowCount] = row.getFloat("Max_temp");
  y_sep[rowCount] = row.getFloat("Min_temp");
  z_sep[rowCount] = row.getFloat("Rainfall");
  n_sep[rowCount] = row.getFloat("Year_September");
  rowCount++;
}
table_oct = loadTable("points_oct.csv", "header");

// You can access iterate over all the rows in a table
rowCount = 0;
for (TableRow row : table_oct.rows ()) {
```

```
// You can access the fields via their column name (or index)
  x_oct[rowCount] = row.getFloat("Max_temp");
  y_oct[rowCount] = row.getFloat("Min_temp");
  z_oct[rowCount] = row.getFloat("Rainfall");
  n_oct[rowCount] = row.getFloat("Year_October");
  rowCount++;
}
table_nov = loadTable("points_nov.csv", "header");

// You can access iterate over all the rows in a table
rowCount = 0;
for (TableRow row : table_nov.rows ()) {
  // You can access the fields via their column name (or index)
  x_nov[rowCount] = row.getFloat("Max_temp");
  y_nov[rowCount] = row.getFloat("Min_temp");
  z_nov[rowCount] = row.getFloat("Rainfall");
  n_nov[rowCount] = row.getFloat("Year_November");
  rowCount++;
}
table_dec = loadTable("points_dec.csv", "header");
```

```
  // You can access iterate over all the rows in a table
  rowCount = 0;
  for (TableRow row : table_dec.rows ()) {
    // You can access the fields via their column name (or index)
    x_dec[rowCount] = row.getFloat("Max_temp");
    y_dec[rowCount] = row.getFloat("Min_temp");
    z_dec[rowCount] = row.getFloat("Rainfall");
    n_dec[rowCount] = row.getFloat("Year_December");
    rowCount++;
  }
  //  println();

  //*/
}



void draw() {

  //  smooth();
```

```
//background(0);

 //  int a = int (random(0, width));
 //  int b = int (random(0, height));
 //  int c = int (random(255));
 //  float sColor = map(a, 0, width, 0, 100);
 //  stroke(sColor, 80, 90);

 //color(random(0,255), random(0,255), random(0,255));

 pushMatrix();
 translate(width/2, height/2);


 rotateY(frameCount/100.0);
 //rotateZ(frameCount /100.0);
 rotateX(frameCount/100.0);


 //box(300);
 for (int i = 0; i<amount; i++) {
```

```
 if (currentMonth==1) {

    point(map(x_jan[i], 0, 50, -stageX, stageX), map(y_jan[i], 0, 30, -stageY, stageY),
map(z_jan[i], 0, 700, -stageZ, stageZ));

  } else if (currentMonth==2) {

    point(map(x_feb[i], 0, 50, -stageX, stageX), map(y_feb[i], 0, 30, -stageY, stageY),
map(z_feb[i], 0, 700, -stageZ, stageZ));

  } else if (currentMonth==3) {

    point(map(x_mar[i], 0, 50, -stageX, stageX), map(y_mar[i], 0, 30, -stageY, stag-
eY), map(z_mar[i], 0, 700, -stageZ, stageZ));

  } else if (currentMonth==4) {

    point(map(x_apr[i], 0, 50, -stageX, stageX), map(y_apr[i], 0, 30, -stageY, stageY),
map(z_apr[i], 0, 700, -stageZ, stageZ));

  } else if (currentMonth==5) {

    point(map(x_may[i], 0, 50, -stageX, stageX), map(y_may[i], 0, 30, -stageY, stag-
eY), map(z_may[i], 0, 700, -stageZ, stageZ));

  } else if (currentMonth==6) {

    point(map(x_jun[i], 0, 50, -stageX, stageX), map(y_jun[i], 0, 30, -stageY, stageY),
map(z_jun[i], 0, 700, -stageZ, stageZ));

  } else if (currentMonth==7) {

    point(map(x_jul[i], 0, 50, -stageX, stageX), map(y_jul[i], 0, 30, -stageY, stageY),
map(z_jul[i], 0, 700, -stageZ, stageZ));

  } else if (currentMonth==8) {

    point(map(x_aug[i], 0, 50, -stageX, stageX), map(y_aug[i], 0, 30, -stageY, stag-
eY), map(z_aug[i], 0, 700, -stageZ, stageZ));
```

```
} else if (currentMonth==9) {

    point(map(x_sep[i], 0, 50, -stageX, stageX), map(y_sep[i], 0, 30, -stageY, stageY), map(z_sep[i], 0, 700, -stageZ, stageZ));

  } else if (currentMonth==10) {

    point(map(x_oct[i], 0, 50, -stageX, stageX), map(y_oct[i], 0, 30, -stageY, stageY), map(z_oct[i], 0, 700, -stageZ, stageZ));

  } else if (currentMonth==11) {

    point(map(x_nov[i], 0, 50, -stageX, stageX), map(y_nov[i], 0, 30, -stageY, stageY), map(z_nov[i], 0, 700, -stageZ, stageZ));

  } else if (currentMonth==12) {

    point(map(x_dec[i], 0, 50, -stageX, stageX), map(y_dec[i], 0, 30, -stageY, stageY), map(z_dec[i], 0, 700, -stageZ, stageZ));

  }

 }

 popMatrix();

}




void keyPressed() {


 //  if (key == 's') {

 //    saveFrame("saved-3/"+currentMonth+"-#.png");
```

```
// }

 writeText();

 if (key == '1') {
   stroke(39, 169, 225);
   currentMonth = 1;
 } else if (key == '2') {
   stroke(0, 167, 157);
   currentMonth = 2;
 } else if (key == '3') {
   stroke(0, 147, 68);
   currentMonth = 3;
 } else if (key == '4') {
   stroke(56, 180, 73);
   currentMonth = 4;
 } else if (key == '5') {
   stroke(140, 198, 62);
   currentMonth = 5;
 } else if (key == '6') {
   stroke(214, 223, 35);
```

```
vv currentMonth = 6;
  } else if (key == '7') {
    stroke(248, 237, 49);
    currentMonth = 7;
  } else if (key == '8') {
    stroke(251, 175, 63);
    currentMonth = 8;
  } else if (key == '9') {
    stroke(247, 147, 29);
    currentMonth = 9;
  } else if (key == 'a') {
    stroke(240, 90, 40);
    currentMonth = 10;
  } else if (key == 'b') {
    stroke(239, 64, 54);
    currentMonth = 11;
  } else if (key == 'c') {
    stroke(27, 117, 187);
    currentMonth = 12;
  }
}
```

```
void writeText() {


 if (key == '1' || key == '2' || key == '3' || key == '4' || key == '5' || key == '6' || key ==
'7' || key == '8' || key == '9' || key == 'a' || key == 'b' || key == 'c'  ) {
   background(0);
   stroke(255, 203, 30);
   line(300, (height/2-90), 300, (height/2+80));
 }
 // line ((width - 140), (height/2 + 18), (width - 20), (height/2 + 18));
 //*




 if (key == '1') {
   currentMonth = 1;
   textSize(24);
   text("January", (width-125), (height/2));
   textSize(60);
   text("1", (width-95), (height/2 + 82));
   line ((width - 140), (height/2 + 18), (width - 20), (height/2 + 18));
 } else if (key == '2') {
```

```
textSize(24);
    text("February", (width-130), (height/2));
    textSize(60);
    text("2", (width-95), (height/2 + 82));
    line ((width - 140), (height/2 + 18), (width - 20), (height/2 + 18));
  } else if (key == '3') {
    textSize(24);
    text("March", (width-115), (height/2));
    textSize(60);
    text("3", (width-95), (height/2 + 82));
    line ((width - 140), (height/2 + 18), (width - 20), (height/2 + 18));
  } else if (key == '4') {
    textSize(24);
    text("April", (width-105), (height/2));
    textSize(60);
    text("4", (width-95), (height/2 + 82));
    line ((width - 140), (height/2 + 18), (width - 20), (height/2 + 18));
  } else if (key == '5') {
    textSize(24);
    text("May", (width-105), (height/2));
    textSize(60);
```

```
text("5", (width-95), (height/2 + 82));
    line ((width - 140), (height/2 + 18), (width - 20), (height/2 + 18));
  } else if (key == '6') {
    textSize(24);
    text("June", (width-105), (height/2));
    textSize(60);
    text("6", (width-95), (height/2 + 82));
    line ((width - 140), (height/2 + 18), (width - 20), (height/2 + 18));
  } else if (key == '7') {
    textSize(24);
    text("July", (width-105), (height/2));
    textSize(60);
    text("7", (width-95), (height/2 + 82));
    line ((width - 140), (height/2 + 18), (width - 20), (height/2 + 18));
  } else if (key == '8') {
    textSize(24);
    text("August", (width-120), (height/2));
    textSize(60);
    text("8", (width-95), (height/2 + 82));
    line ((width - 140), (height/2 + 18), (width - 20), (height/2 + 18));
  } else if (key == '9') {
```

```
textSize(24);
    text("September", (width-140), (height/2));
    textSize(60);
    text("9", (width-95), (height/2 + 82));
    line ((width - 140), (height/2 + 18), (width - 20), (height/2 + 18));
  } else if (key == 'a') {
    textSize(24);
    text("October", (width-125), (height/2));
    textSize(60);
    text("10", (width-110), (height/2 + 82));
    line ((width - 140), (height/2 + 18), (width - 20), (height/2 + 18));
  } else if (key == 'b') {
    textSize(24);
    text("November", (width-138), (height/2));
    textSize(60);
    text("11", (width-105), (height/2 + 82));
    line ((width - 145), (height/2 + 18), (width - 20), (height/2 + 18));
  } else if (key == 'c') {
    textSize(24);
    text("December", (width-142), (height/2));
    textSize(60);
```

```
text("12", (width-110), (height/2 + 82));
  line ((width - 140), (height/2 + 18), (width - 20), (height/2 + 18));
 }



 textFont(font1, 18);
 // fill(#B5B5B1);
 fill(255);
 text("Press", 40, (height/2 - 80));
 textFont(font2, 18);
 fill(255, 203, 30);
 text("1 ", 95, (height/2 - 80));
 textFont (font1, 18);
 fill(255);
 text("to", 112, (height/2 - 80));
 textFont(font2, 18);
 fill(255, 203, 30);
 text("9", 138, (height/2 - 80));
 textFont(font1, 18);
 fill(255);
```

```
 text("and ", 158, (height/2 - 80));
  textFont(font2, 18);
  fill(255, 203, 30);
  text ("a, b, c", 198, (height/2 - 80));
  textFont(font1, 18);
  fill(255);
  text("to toggle between different ", 40, (height/2-40));
  text("months's", 40, (height/2));
  textFont(font2, 18);
  fill(255, 203, 30);
  text("visualisation", 130, (height/2));

  text("Click", 40, (height/2 + 80));
  fill(255);
  textFont(font1, 18);
  text("to save a", 90, (height/2 + 80) );
  textFont(font2, 18);
  fill(255, 203, 30);
  text ("frame", 170, (height/2 + 80));
  //*/
}
```

```
void mousePressed() {


 endRaw();


 //background(0);
 //saveFrame("saved-2/"+currentMonth+"-##.png");
 //  int a = int (random(0, width));
 //  int b = int (random(0, height));
 //  //int c = int (random(255));
 //  float sColor = map(a, 0, width, 0, 100);
 //  stroke(sColor, 80, 90);
}
```

Fig 6 at the next page is a screenshot of the application that is made by compiling the code written above.

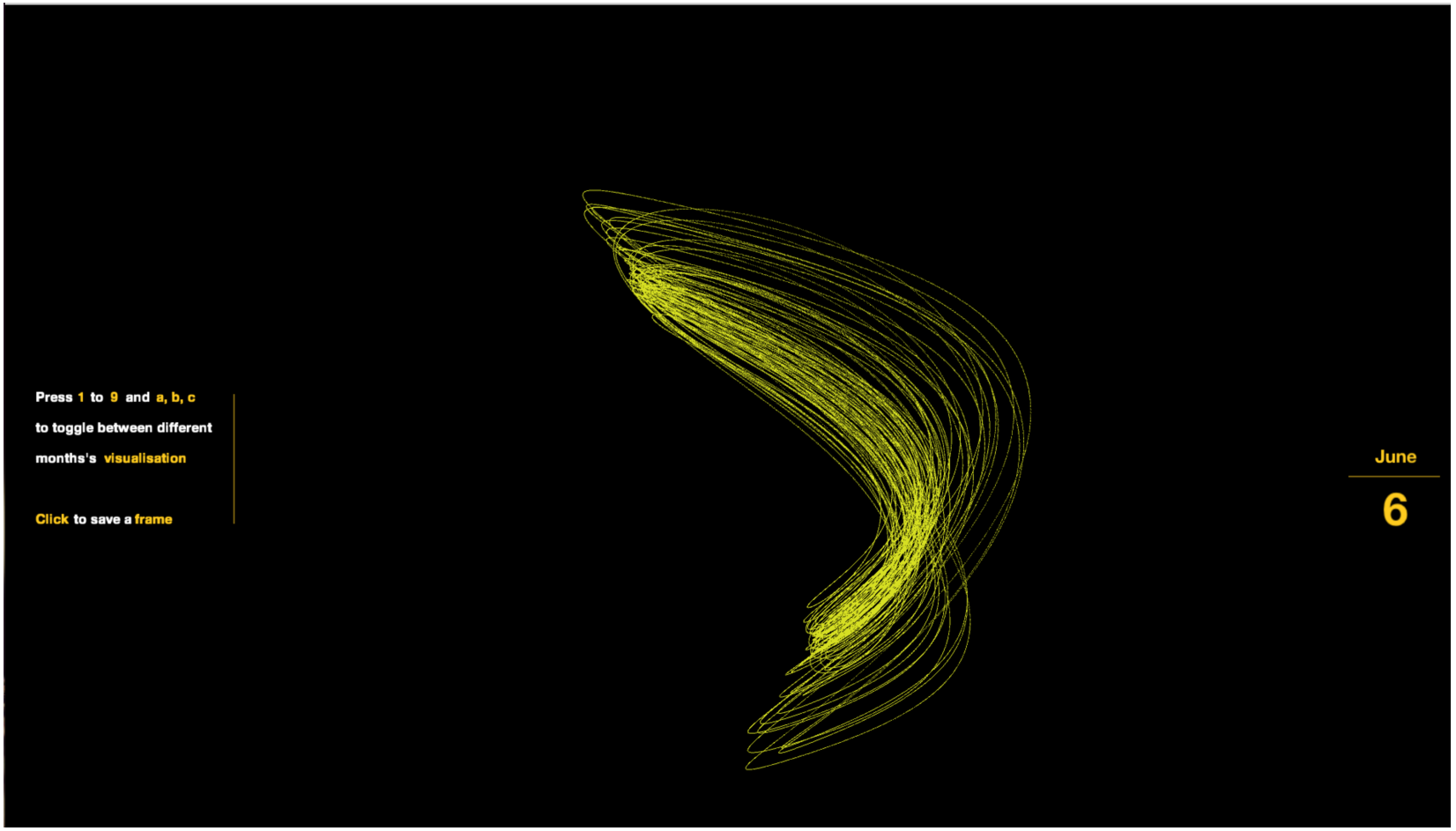


Fig 6: A screenshot of the application built in processing for generating the patterns

The code for generating the graph for each month is given below. The code below is for generating the graph of January month.

```
import processing.pdf.*;

Table table;

int amount = 51;

PVector[] positions_x = new PVector[amount];

PVector[] positions_y = new PVector[amount];

PVector[] positions_z = new PVector[amount];

float[] x = new float[amount];

float[] y = new float[amount];

float[] z = new float[amount];

String[] n = new String[amount];


void setup() {
  size(1080, 780, OPENGL);
  beginRaw(PDF, "x-###.pdf");
  background(0);
  noFill();
  strokeWeight(.5);
  smooth();


  loadData();
```

```
graphData();
}
void loadData() {
  // Load CSV file into a Table object
  // "header" option indicates the file has a header row
  table = loadTable("points_jan.csv", "header");

  // You can access iterate over all the rows in a table
  int rowCount = 0;
  for (TableRow row : table.rows ()) {
    // You can access the fields via their column name (or index)
    x[rowCount] = row.getFloat("Max_temp");
    y[rowCount] = row.getFloat("Min_temp");
    z[rowCount] = row.getFloat("Rainfall");
    n[rowCount] = row.getString("Year_Jan");
    rowCount++;
  }
}
void draw() {
  for (int i = 0; i<amount; i++) {
    fill(#FC4349);
```

```
noStroke();
   ellipse(positions_x[i].x, positions_x[i].y, 10, 10);

   fill(#314678);
   ellipse(positions_y[i].x, positions_y[i].y, 10, 10);
   fill(#D7DADB);
   ellipse(positions_z[i].x, positions_z[i].y, 10, 10);
   stroke(255, 255, 255, 2);
   line (positions_x[i].x, positions_x[i].y, positions_z[i].x, positions_z[i].y );
  }
}

void graphData() {

  float xMin = 0.0;
  float xMax = 35.0;
  float yMin = min(y);
  float yMax = max(y);
  float zMin = 0.0;
  float zMax = 1500.0;
```

```
//    println(xMin);
 //    println(xMax);
int margin = 50;
 int graphHeight = height - margin * 2;
 int xGutter = width - margin * 2;
 for ( int i = 0; i < amount; i++) {
  // mapping x, y, z according to the coordinates of the graph size
  float xMap = map (x[i], xMin, xMax, 0, graphHeight);
  float xYPos = height - margin - xMap;
  float xXPos =  margin*i/2.7 + width/12;
  float yMap = map (y[i], xMin, xMax, 0, graphHeight);
  float yYPos = height - margin - yMap;
  float yXPos =  margin*i/2.7 + width/12;
  float zMap = map (z[i], zMin, zMax, 0, graphHeight);
  float zYPos = height - margin - zMap;
  float zXPos =  margin*i/2.7 + width/12;
  positions_x[i] = new PVector(xXPos, xYPos);
  positions_y[i] = new PVector(yXPos, yYPos);
  positions_z[i] = new PVector(zXPos, zYPos);
 }
}
```

```
void mousePressed() {


  endRaw();


 // background(0);
 //saveFrame("save1/line-####.png");
 //  int a = int (random(0, width));
 //  int b = int (random(0, height));
 //  //int c = int (random(255));
 //  float sColor = map(a, 0, width, 0, 100);
 //  stroke(sColor, 80, 90);
}
```

Fig 7 is a screenshot of the graph when generated by compiling the code in processing.

A list of compiled images are shown in the next few pages that has all the patterns and graphs grouped togeher.

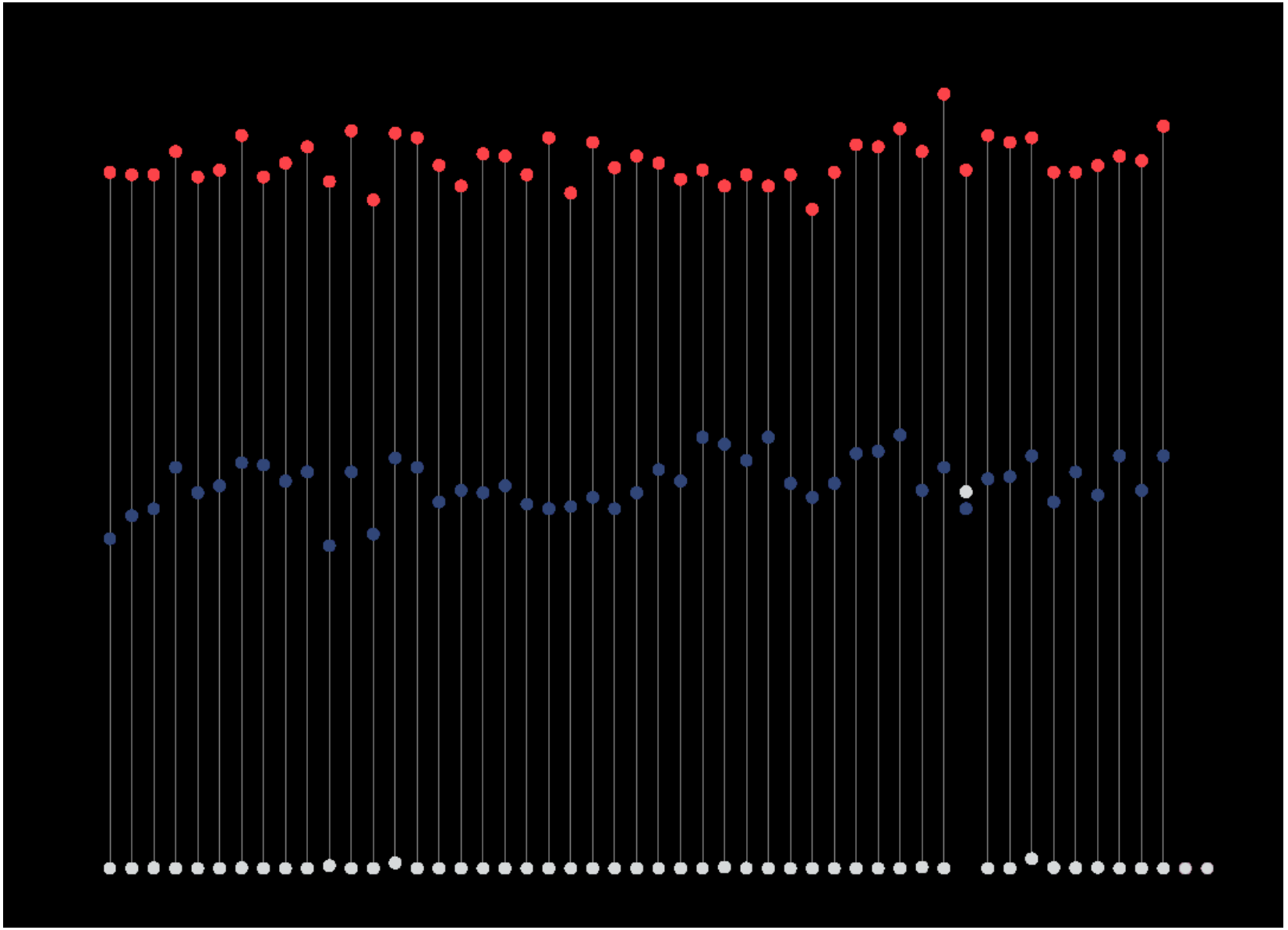


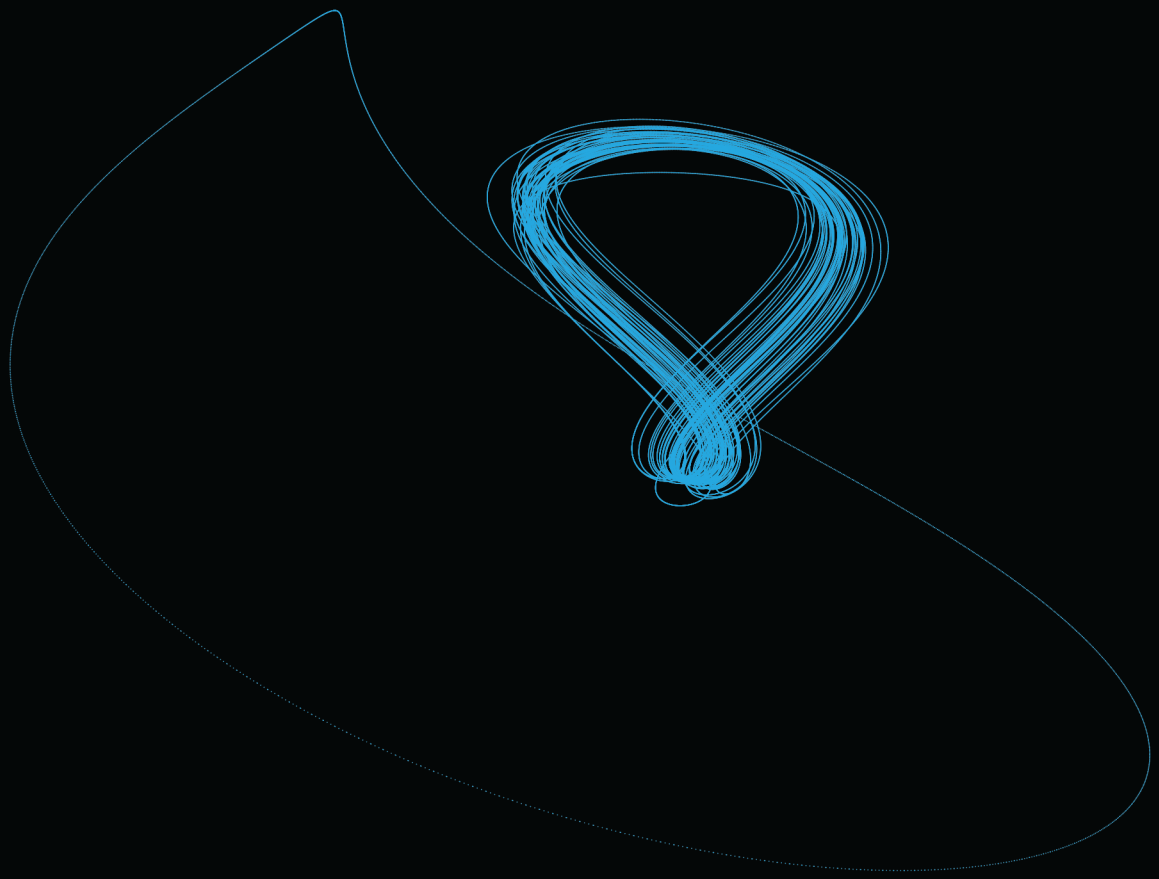
Fig 7: A screenshot of the graph generated in processing

January

Average maximum temperature, minimum temperature, and rainfall for the month of January from 1951 to 2000 is plotted as points rotating in a 3D space. The pattern is formed when the path of all the 50 revolving points are traced.

The eccentric path at the bottom depicts the unusual rainfall of 700mm in the year of 1991.

The small multiples at the bottom depicsts 50 years of average minimum, maximum temperature and rainfall for the corresponding 12 months.



Temperature Range
0 - 35 ° C

Rainfall Range
0 - 1500 mm

Max Temperature Min Temperature Rainfall

Fig 8: A pattern of the month of January along with the graphs of each month

A story

When all the patterns of each month are put together, they tell us a story. January beign one of the coldest month has the least height among the patterns and July being the hottest month has the highest one. So as temperature increases height of the pattern increases. January is a nearly perfect symmetrical pattern along the vertical axis. As rainfall comes into the picture, the axis of symmetry changes to the horizontal axis. This can be seen from the month of June to September, the months having most rainfall in 50 years time.

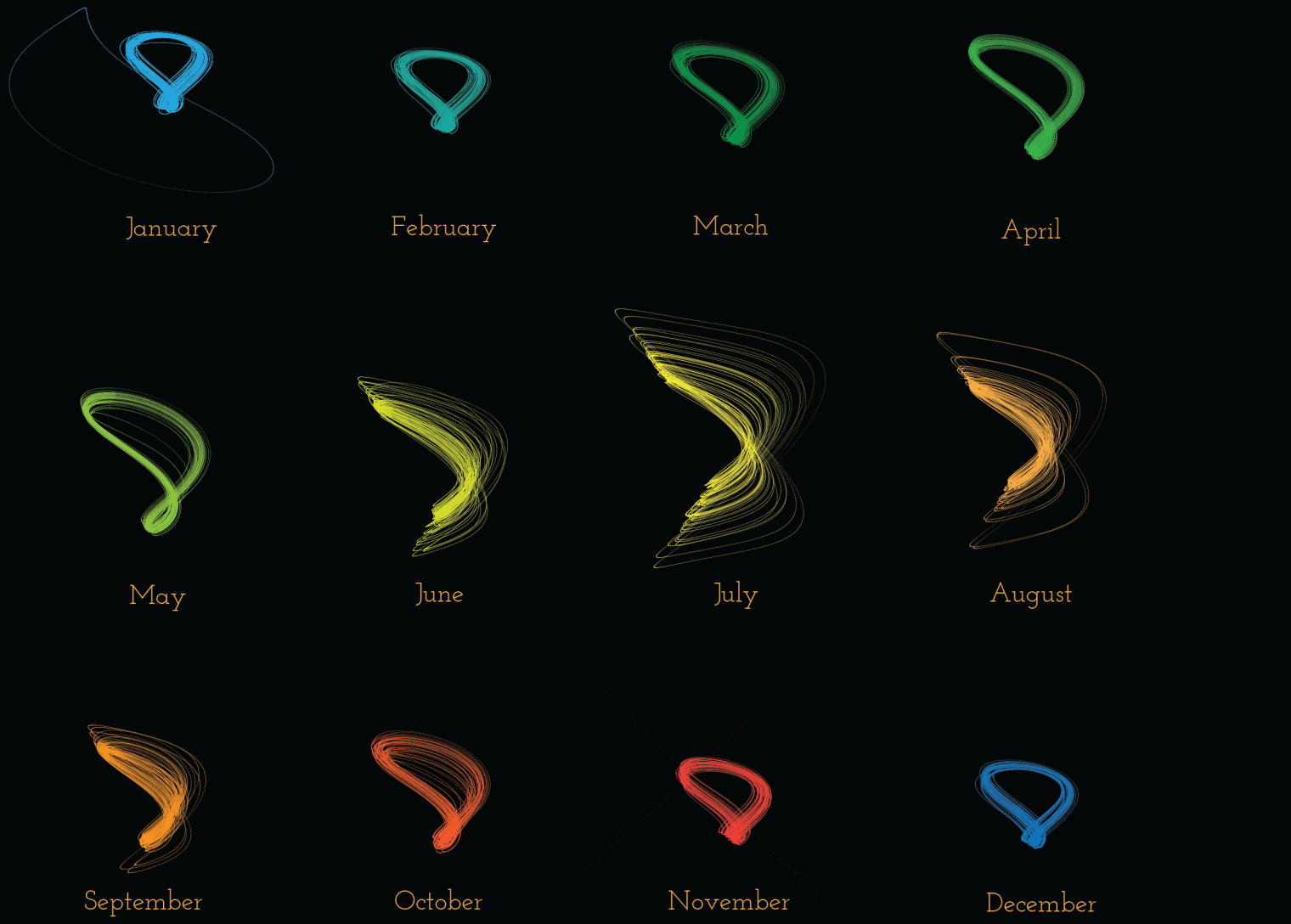


Fig 9: All the patterns are grouped together along with the graphs of each month

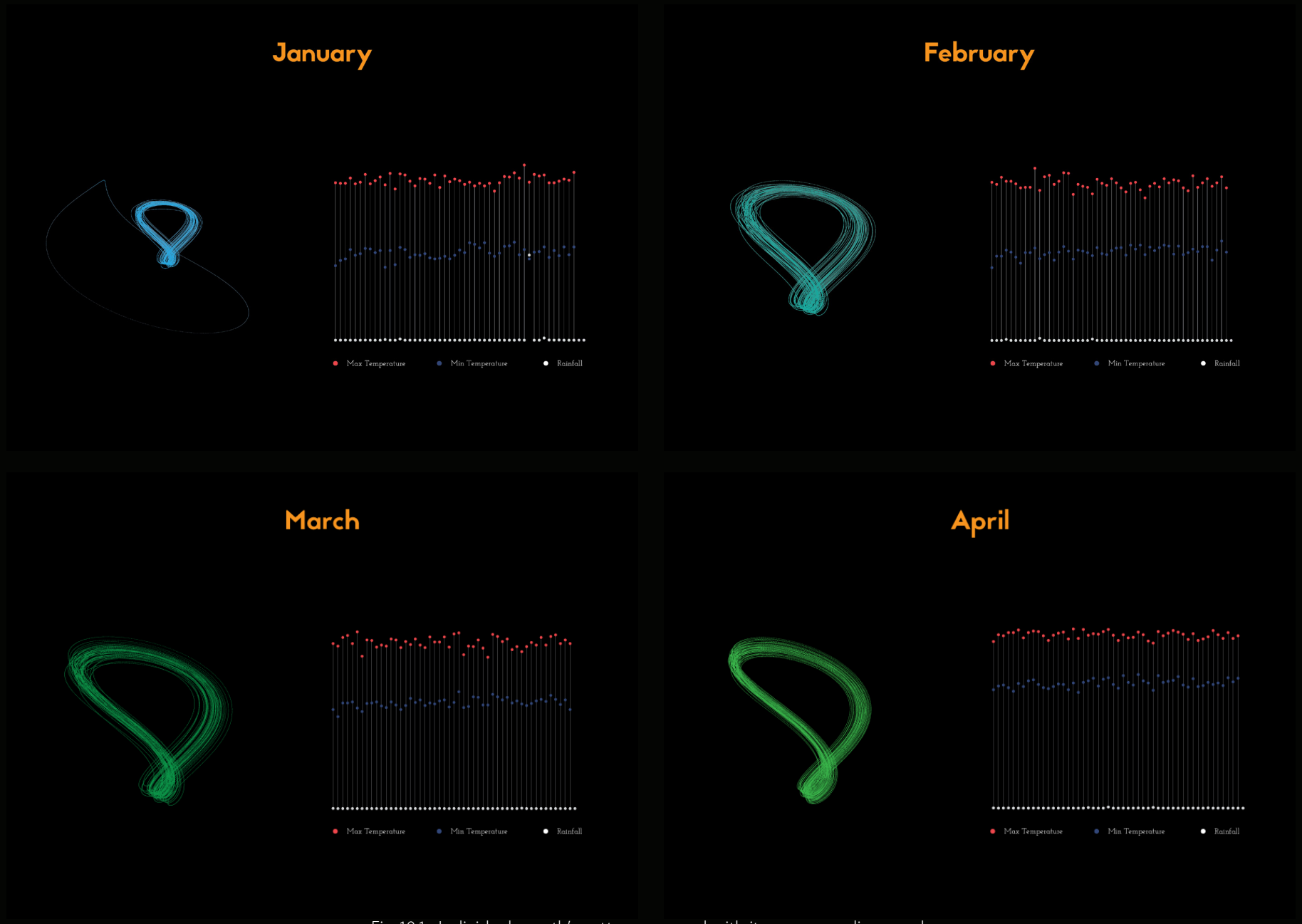


Fig 10.1 : Individual month's pattern grouped with its corresponding graph

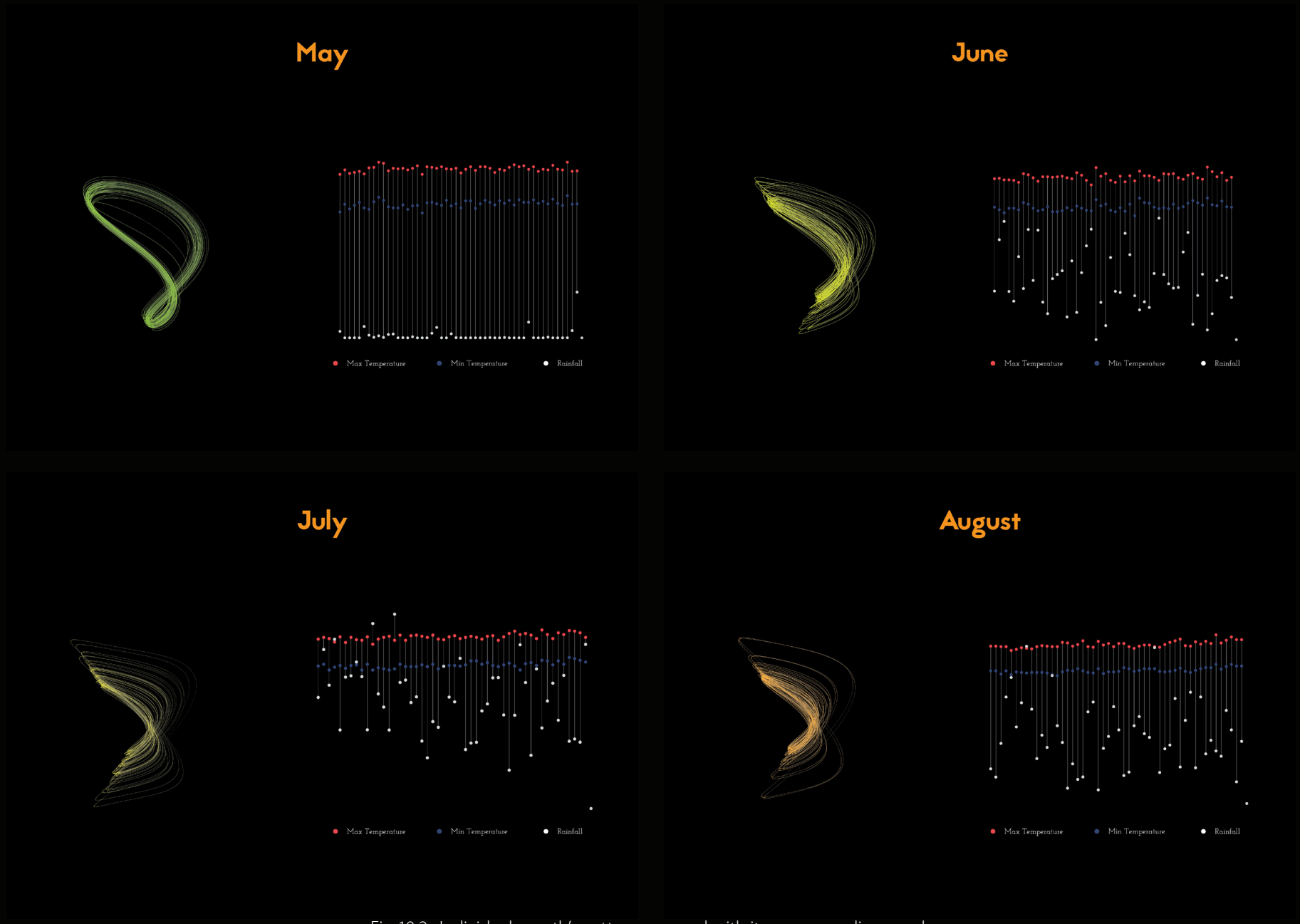


Fig 10.2 : Individual month's pattern grouped with its corresponding graph

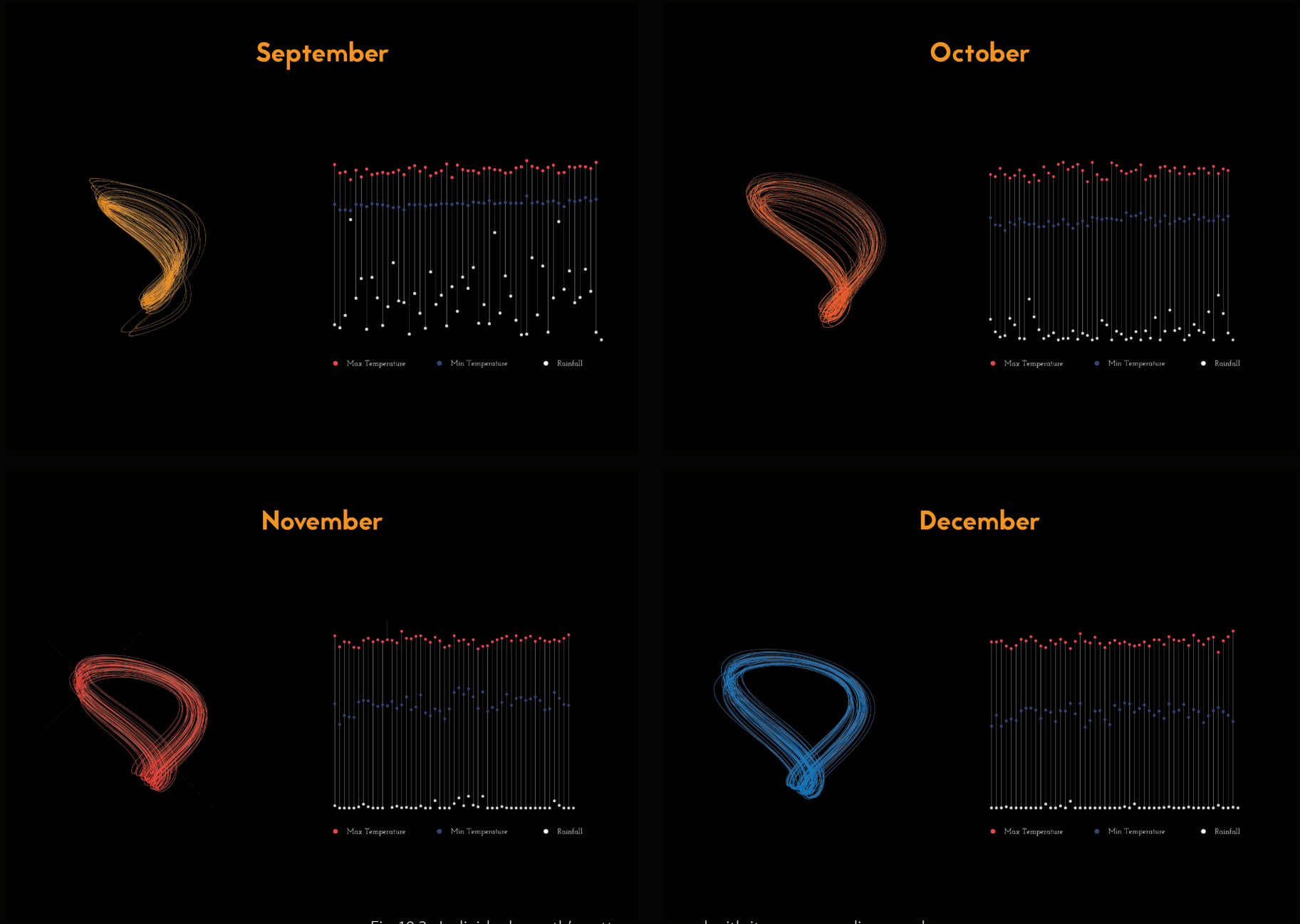


Fig 10.3 : Individual month's pattern grouped with its corresponding graph

Bibliography

[1] Creative code: Exploring Generative System to Create Art.

[2] CHIMERA.LABS.OREILLY.COM
Interactive Data Visualization for the Web
In-text: (Chimera.labs.oreilly.com, 2015)
Bibliography: Chimera.labs.oreilly.com, (2015). Interactive Data Visu
alization for the Web. [online] Available at: http://chimera.labs.oreilly.com/
books/1230000000345/index.html [Accessed 4 Dec. 2014].

[3] CODING TUTORIALS AND BOOKS
In-text: (Coding tutorials and books, 2015)
Bibliography: Coding tutorials and books. (2015). [online] Available at:
http://p5art.tumblr.com/tutorials [Accessed 10 Dec. 2014].

[4] FYPROCESSING.TUMBLR.COM
For Your Processing
In-text: (Fyprocessing.tumblr.com, 2015)
Bibliography: Fyprocessing.tumblr.com, (2015). For Your Processing. [on line] Available at: http://fyprocessing.tumblr.com/page/12 [Accessed 9 Nov. 2014].

[5] SCHUELLER, A.
Introduction to Processing: June 2013
In-text: (Schueller, 2013)
Bibliography: Schueller, A. (2013). Introduction to Processing: June 2013. [online] Introprocessing.blogspot.in. Available at: http://introprocessing.blogspot.in/2013_06_01_archive.html [Accessed 8 Dec. 2014].

[6] BOHNACKER, H., GROSS, B., LAUB, J. AND LAZZERONI, C.
Generative design
In-text: (Bohnacker et al., 2012)
Bibliography: Bohnacker, H., Gross, B., Laub, J. and Lazzeroni, C. (2012). Generative design. New York: Princeton Architectural Press.

[7] FRY, B.
Visualizing data
In-text: (Fry, 2008)
Bibliography: Fry, B. (2008). Visualizing data. Beijing: O'Reilly Media, Inc.

[8] GREENBERG, I., XU, D. AND KUMAR, D.
Processing
In-text: (Greenberg, Xu and Kumar, 2013)
Bibliography: Greenberg, I., Xu, D. and Kumar, D. (2013). Processing. Berkeley, Calif.: Friends of Ed.

[9] PEARSON, M.
Generative art
In-text: (Pearson, 2011)
Bibliography: Pearson, M. (2011). Generative art. Shelter Island, NY: Manning.

[10] REAS, C. AND FRY, B.
Processing
In-text: (Reas and Fry, 2007)
Bibliography: Reas, C. and Fry, B. (2007). Processing. Cambridge, Mass.: MIT Press.

[11] TERZIDIS, K.
Algorithms for visual design using the Processing language
In-text: (Terzidis, 2009)
Bibliography: Terzidis, K. (2009). Algorithms for visual design using the Processing language. Indianapolis, IN: Wiley Pub.