Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 Keller Hall
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 19-006

Open Science: GitHub Site Publication of Hotspot Algorithms

Wen Jing

May 3, 2019

Spring
2019

UNIVERSITY OF MINNESOTA
MS DATA SCIENCE CAPSTONE PROJECT

Open Science: GitHub Site Publication of Hotspot Algorithms

Supervisor: Prof. Shashi Shekhar, Prof. George Karypis, Prof. Eric Shook

Author: Wen Jing

CONTENT

INTRODUCTION

1.1 MOTIVATION: OPEN SCIENCE

Open science is transparent and accessible knowledge that is shared and developed through collaborative networks [1]. The possible subsections under open science are: open data, open source, open peer review, open access, open methodology, and open educational resources. Open science will promote science reproducibility. Thus, the lab result can be re-achieved by other researchers or users. Publication plus a linked and executable code and data can make the full replication. [2]

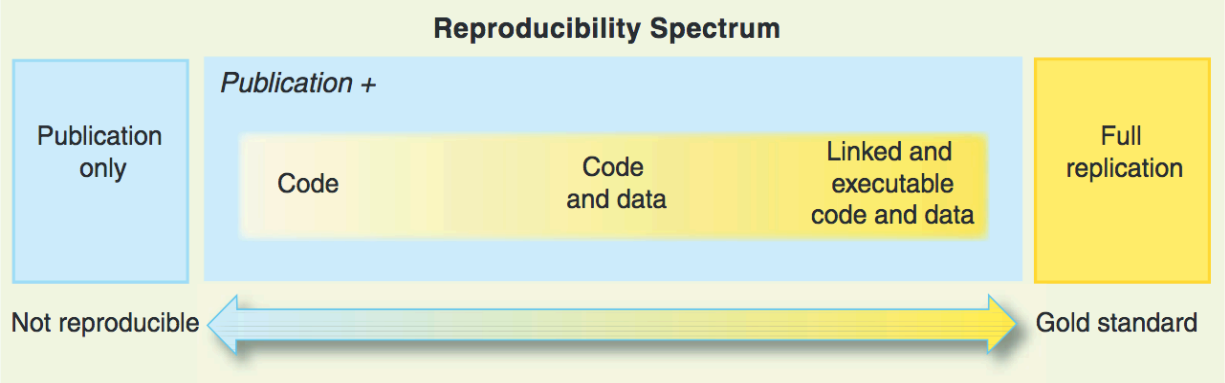


Figure 1 The spectrum of reproducibility.

It also helps distribute resource equally among different science groups, so labs in small institutions can have access to more resource for their own study and research. This report focuses on optimizing, modularizing and publishing lab code to eliminate gaps between lab achievements and users from different fields. This addresses the issues associated with open access, open methodology, and open source.

To publish lab code online is a part of democratizing science. It allows public to participate in science development. Public can help check the correctness of the lab result and accelerate the lab developing process.

The tool that is used to achieve the goal of open science in this project is GitHub. Microsoft completed the acquisition of GitHub in 2018 with $7.5 billion, which potentially setting GitHub as the trend in the social coding community. Thus, in the project, GitHub is chosen as a bridge between lab achievements and ordinary users.

1.2 PROBLEM STATEMENT

This project is aiming to eliminate the gap between lab achievements and users from different fields and to make contributions for open science. The process can be expressed as follows:

Given: Published paper and separated code used for testing in lab

Output: GitHub Page that

(1). Can be used by users without a strong computer science background.

(2). Contains algorithm explanation for academic users to use.

(3). Contains modularized code for users to replace subsections based on different goals

Constraints:

(1). Does not change logic

(2). Stay with the original language.

Social coding enables a different experience of software development as the activities and interests of one developer are easily advertised to other developers [3]. However, the algorithms and concepts ordinary developers use to develop new functions of their software were usually proved correct decades ago or even hundreds of years ago and are "common sense" nowadays in academic areas. The latest academic achievements usually do not have associated software. The code is usually for lab testing and only a few researchers who make the achievement are able to execute the code well. For ordinary users that may benefit from the new achievement, it may take years for them to finally use the code to get the desired results in their own projects. In other words, there is a gap between lab achievements and their potential audiences. In this report I use the GitHub as tool to try to fix this gap.

GitHub is mainly a computer code community for users to get open-source code, and also publish their own. It is arguably the largest social coding site, containing more than 31 million users and 3 million repositories. It offers the version control and source code management functionality of Git as well as allowing all users to add features to the existing code. Developers can also track all the changes that have been made by themselves or other contributors. Such a major change in collaborative software development makes an investigation of networking on social coding sites valuable [3].

1.3 CHALLENGES

Users often have different levels of computer science background knowledge: Researchers and ordinary users. Researchers may need the code to support a part of their academic paper. These people usually have strong computer science background knowledge. They need to fully understand the algorithm and all the conditions behind the code to meet their own research goals. The academic users may also have the requirement of using some specific methods in some of the classes to create their own function under one repository. But since our algorithm detects hotspots, lots of our ordinary users are from the GIS area, who may not have a strong computer science background knowledge. For those users, they may care more about getting the results of hotspot areas than knowing every detail in the algorithm that run behind it. It is hard to balance all the users' needs within one GitHub Repository.

The hotspots detection methods have lots of versions in the lab already. But it seems there is not an efficient way for researchers to share the lab code to ordinary users. All though the algorithms have been published in papers, the codes they use are only the lab testing versions, which are not used for publishing. If users want to use the code, they have to email the researcher for the testing code and the researcher has to explain everything about using the code to each user in different emails. It is a waste of time for the researcher to repeat the same work over and over again and it is also inconvenient for users to use.

Eliminating the gap between lab achievement and users is challenging due to different purpose on different sides. Testing code in the lab is usually not for other users to read. It contains lots of testing cases and comments for the developers only. Different test pieces are usually executed manually and with some variable changes in between. That would be too complicate for normal users to use. Each hotspots detection repository usually has 1000 to 2000 lines of code waiting to be reorganized. To meet academic users' requirements, the code has to be modularized and separated into pieces based on the logic, so that each piece can be replaced by users-defined-functions. To meet ordinary users' requirements, there should be a clear instruction that can be easily followed. The finally GitHub page has to consider about different users and meet all the users' needs.

1.4 Related Work

In the geographic information systems (GIS) area, detecting hotspots is a very common method. A hotspot can be defined as an area that has a higher concentration of events compared to the expected number given a random distribution of events. Hotspots detection methods are also wildly used in lots of other disciplines, such as public health, transportation, criminology, urban planning, etc. There are lots of different ways to detect hotspots in the lab based on different research goals. Geographically Robust Hotspot Detection (GRHD) finds hotspot areas where the concentration of points inside is significantly high [4]. Significant Linear Hotspot Discovery (SLHD) identifies routes with statistically significant concentrations of activities (e.g., crimes, accidents, etc.) [5], which detects linear shaped hotspots. Elliptical Hotspot Detection (EHD) finds elliptical hotspot areas where the concentration of activities inside is significantly higher than outside [6].

The tool that has been chosen to publish the code is GitHub. It features include: collaborative code review, branch comparison views, support for more than 200 languages and data formats and it can also be used as an integrated issue tracker. There are also some similar tools that can be an alternative of GitHub, such as Bitbucket, GitLab, Kiln, and Beanstalk. Bitbucket can do pull requests, code reviews, branch comparison and commit history and it supports Git and Mercurial VCS, while GitHub as the name indicates only supports Git. However, it has less functions than GitHub. GitLab aims for the entire software development, deployment and DevOps market. Kiln and Beanstalk are also similar to GitHub, but they are not free tools, thus, this makes them significantly less popular than GitHub.

1.5 Contribution

Three Java repositories on GitHub are introduced in this report. In total of 5000 lines of code have been optimized and the clear code is published on GitHub. For users that do not have strong computer science background knowledge, there are step-by-step README files in each repository, to guide users and make sure users can get the hotspots result on their own datasets using the latest algorithm from the lab.

For academic users, the explanation of concepts and algorithms behind each method is shown in detail on the GitHub wiki pages to provide full reference. For advanced users who would like to modify the code and add their own attributes to the existing approach, my GitHub

page can also meet their requirements. All the codes are reorganized and combined into reasonable sections, based on the logic in between classes. The relationship between all classes are demonstrated with Java class diagrams. Other developers can replace sections based on their own needs. By replacing sections, they can change the minor methods that help find hotspots as well as add the features they need to detect hotspots under different circumstances. The Branch on the GitHub page is used to display core methods that can be used under different hotspots shape discovery. The design of the GitHub page has been modified three time to meet all the users' requirements. The final version is used for all three hotspots detection methods that have been published on GitHub.

My project eliminates the gap between lab achievements and ordinary users. It delivers the optimized code directly to users. It also helps the research lab to get feedback and polish the algorithm to make more progresses.

1.6 Scope and Outline of the Report

This report focuses on achieving open science goals and eliminating gaps between lab achievements and different users. Clear step-by-step code is produced to guide users. Wiki pages in GitHub are used to demonstrate hard concepts behind the algorithms. Code is modularized in to replaceable pieces for advanced users to make their own contributions. The design of the GitHub page is more suitable for users' behaviors. Generalize the lab code into different programming languages is out of scope of this project. I only focus on the original programming language the code was written, in this project is Java.

Chapter 3 presents the details of the GitHub page design. Chapter 4, 5, and 6 provide three case studies using the Significant Linear Hotspot Discovery (SLHD) repository, Geographically Robust Hotspot Detection (GRHD) repository, and Elliptical Hotspot Detection (EHD) repository separately. Conclusions and future work are covered in Chapter 7.

2 GitHub Page Design

2.1 Functions of GitHub

SpatialUMN GitHub page is the place I publish everything. Figure. 1 shows all the tags in the main page. The overview page keeps tracking how many attributes have been made in the past. Each repository currently contains one method of detecting hotspots. Projects tag is used for maintainer to check and organize all the works on GitHub page. Stars, Followers and Following tags plays their roles in social code community. Since all the codes and instructions are published under Repositories tag, this report will focus on repository. In Figure. 2, we can see the popular repositories have already been listed. You can click on any of them and get in to the repository for detail. Or click the Repositories tag and choose the repository.

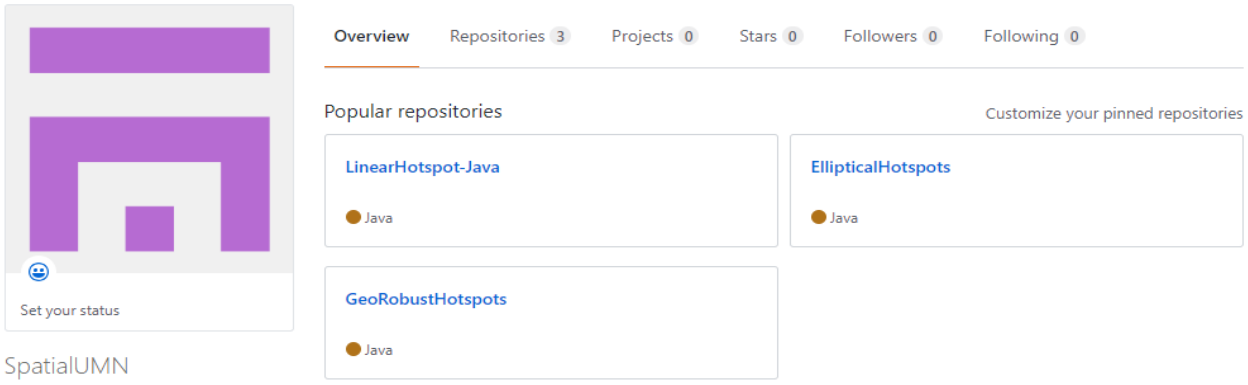


Figure 2 SpatialUMN GitHub Main Page

After getting into one repository, the Code tag is the default tag. The code files and a README file will show up on this page. The main functions covered in this report is shown in Figure 3.

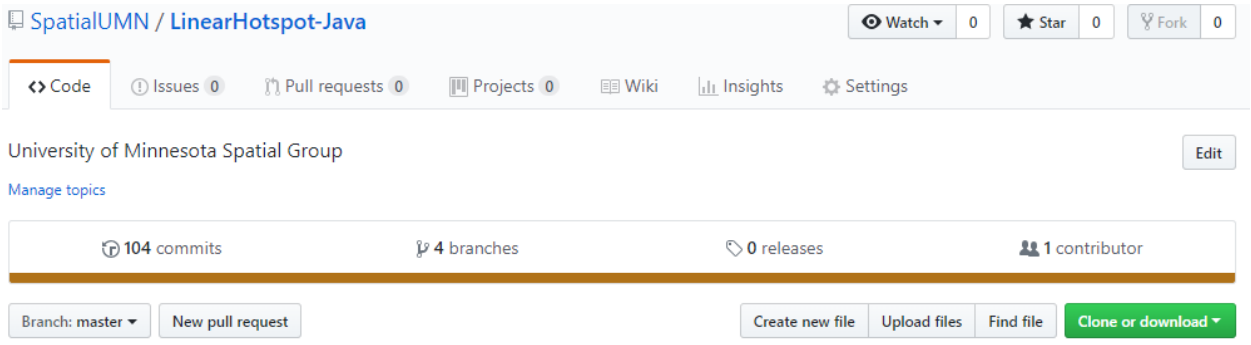


Figure 3 Repository Page

The Issues tag is where users report bugs and ask questions. Pull requests, Projects, and Insights help maintainer tracking the popularity and changes on the page as well as maintain the account. Wiki is an important tag in this project. It contains all the concepts explanations, class relationship diagrams, proof of algorithm correctness and all the details for academic users and advanced users as well as users that want to learn more about the project. The Branch drop list is used to switch between different branches. If one hotspot discovery method has several algorithms to achieve it, there are multiple branches to display those different algorithm functions.

2.2 CHANGE OF DESIGN

There were in total three versions of design for the GitHub page to meet different requirement of different users. The higher version is developed from the lower one. Thus, the code is published use the third version of design. This section also shows the first two version, to unfold all the process to the final version.

2.2.1 Version One

The first version (Figure 4 and Figure 5) has a lot of details in README file. It provides a brief introduction of linear hotspot as well as a link to the motivation of developing this algorithm. Followed by that is the problem statement, given the users a formal definition of the problem. Thus, users would know what to expect in both input and output parts. Academic users can find all the detail about this algorithm in the next part. It contains basic concepts definition and the algorithm explanation. Then the usage part is design for all users. Researchers who doesn't have strong computer science background can also follow the step-by-step instruction and get the linear hotspots result using their own dataset. The usage section includes two parts: Input data preparation and Run the algorithm. The detail will be covered in the case study in later chapters. Then the case study is for that specific algorithm. Bug Report section takes users to the issues page, where they can ask questions, report bugs, and answer other users' questions. The code explanation section contains a Java class diagram, which can help users understand the

structure of the algorithm as well as provide a reference for advance users who may want to add their own contributions to the code.

Version one only considered about researchers who are already familiar with academic paper. There was no outline at the beginning. The problem statement was displayed here for academic users to understand the algorithm. Ordinary users might easily lose interests at the beginning, thus this version of GitHub page might not attractive enough users.

In the first version, the algorithm that has multiple methods was not separated in different branch. There are controlled by a variable. Users have to manually set it every time. Since there is no outline to guide at the beginning, users have to scroll down to look up all the information they need. It is easy to miss any bullet point, and that makes the use experience less efficient.

README.md

Significant Linear Hotspot Discovery

Why Linear Hotspot

Linear Hotspot is a line shape area of significant activity. Our motivation of developing linear hotspot detection method is to identify accident prone road segment.

Problem Statement

Given:

1. A spatial network G = (N, E) with a set of geo-referenced activities A, each if which is associated with an edge.
2. A density ratio threshold, 0_λ
3. A p-value threshold, 0_p and the corresponding number of Monte Carlo simulations needed, m,

Find:
All shortest paths $r \in R$ with $\lambda_r \geq 0_\lambda$, p-value $\leq 0_p$

Objective:
Computational efficiency

Constraints:

1. $r_i \in R$ is not a sub-path of $r_j \in R$ for $\forall r_i, r_j \in R$ where $r_i \neq r_j$
2. $\forall r_i \in R$ is not shorter than a minimum distance (φ) threshold 0_φ
3. Correctness and completeness

Concept and algorithm

Basic Concepts can help better understanding the problem

For academic users, see here for algorithm explanation.

Usage

Input data format

Generate Information for Monte Carlo Simulation

- We need 2 input files: `Node` and `Edge`:
  Node is the road intersaction file that has three attributes: Node ID, Longitude and Latitude. Attributes are separated by `;`
  Edge is the road segment file that has four attributes: Edge ID, Start node id, End node id, and Distance between start node and end node (weight). Attributes are separated by `;`

Detect Linear Hotspots

- We need 3 input files: `Node`, `Edge` and `Activity`:
  The `Node` and `Edge` file formats are the same as the files mentioned in previous session with same names.
  Activity is the car accident file (or whatever interesting events in the research) that has five attributes: Activity ID, Longitude, Latitude, Edge start node id and Edge end node id. The edge here is the one that contains the activity. Attributes are separated by `;`
- The output file contains linear hotspot road segment:
  result has 4 attributes:
  Activity 1 id, Activity 2 id, Shortest path between two activities and Density ratio

Figure 4 First Version of GitHub Design – First half

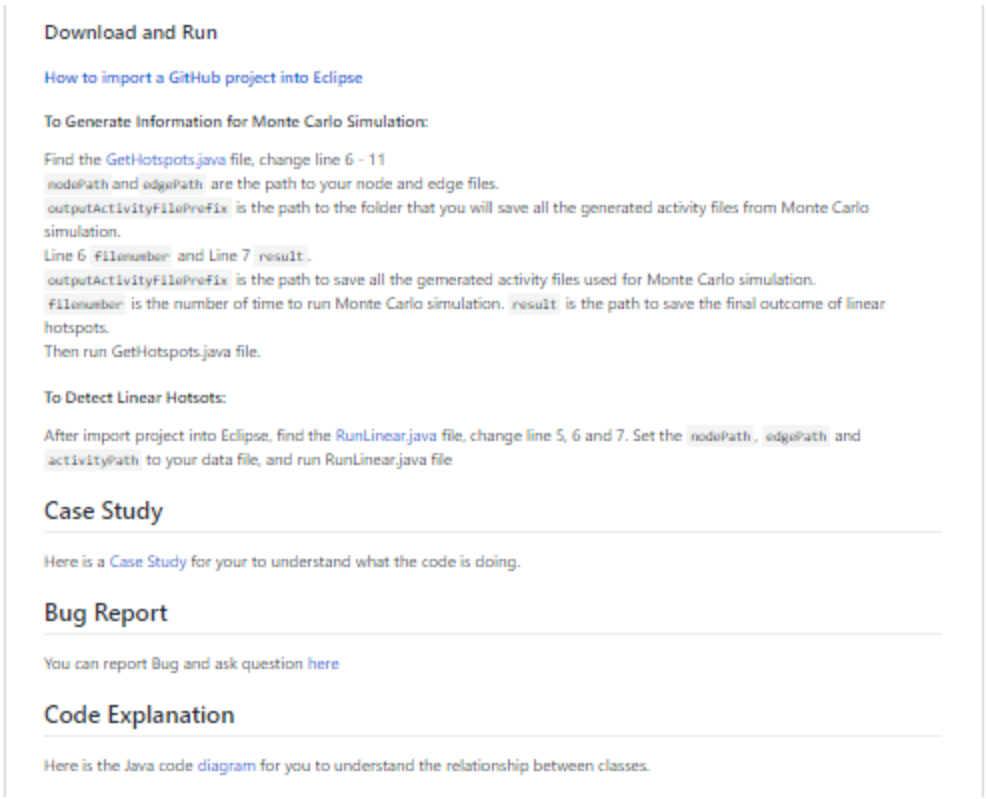


Figure 5 First Version of GitHub Design – Second half

2.2.2 Version Two

Compare with the first version, the second version is trying to consider more about ordinary users and extend the audience group of this GitHub page. The first change in version two is replacing the problem statement with a case study output visualized on road map (Figure 5). The problem statement was moved to wiki page. Academic users can find it through the link. The README file clearly showed users what could they expect from running this code. It is easier for users from all levels to quickly understand the method.

Another important change in the second version is the Use your own code part (Figure 7). The code is much cleaner in this version. It is modified and modularized for users to add their own code into the algorithm. To meet different research goals, researchers sometimes have to use a specific method in particular step. The code in version two is reorganized to help users achieve this goal. But this part is only used on complex algorithm that can be divided into small subproblems. Detail of how to use the users-defined code will be introduced in case study chapters.

For the algorithm that has more than one method to detect the final hotspots result, branch function is used in the second version to clearly display each method. However, the usage instructions are the same among all the branches. It repeats some unnecessary information, even though the final result of each method will not affect each other. The users still need to scroll down to look for the information, because of the lack of an outline.

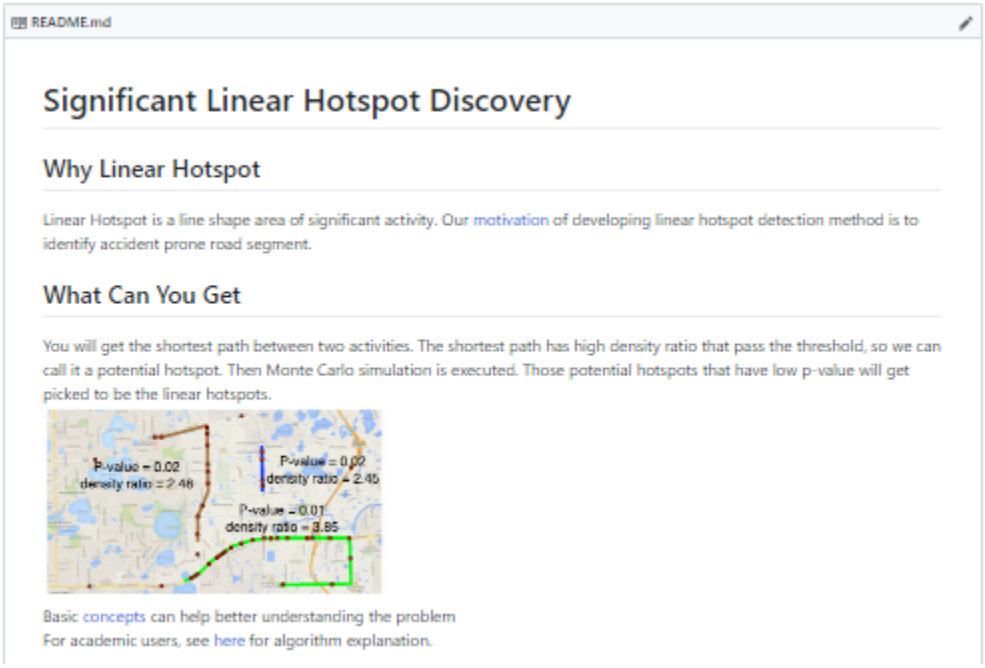


Figure 6 Second Version of GitHub Design – Change 1

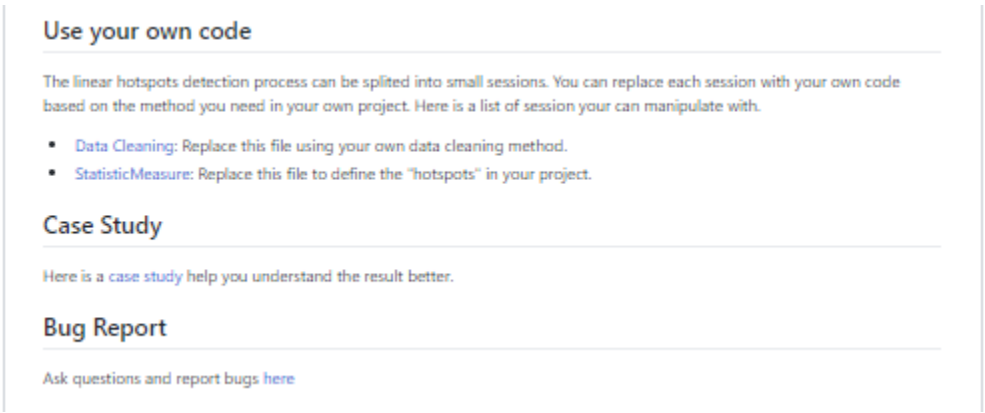


Figure 7 Second Version of GitHub Design – Change 2

2.2.3 Version Three

The third version is used to publish all the lab achievement on the GitHub page. It is improved based on the first two versions. The third version takes all level users' need into

consideration and focus more on detail. An outline (Figure 8) is added at the beginning to guide users. Also help them get a clear understanding of the structure in this page. All the bullet points have hyperlinks, so users do not get lost when jumping from page to page looking for information.

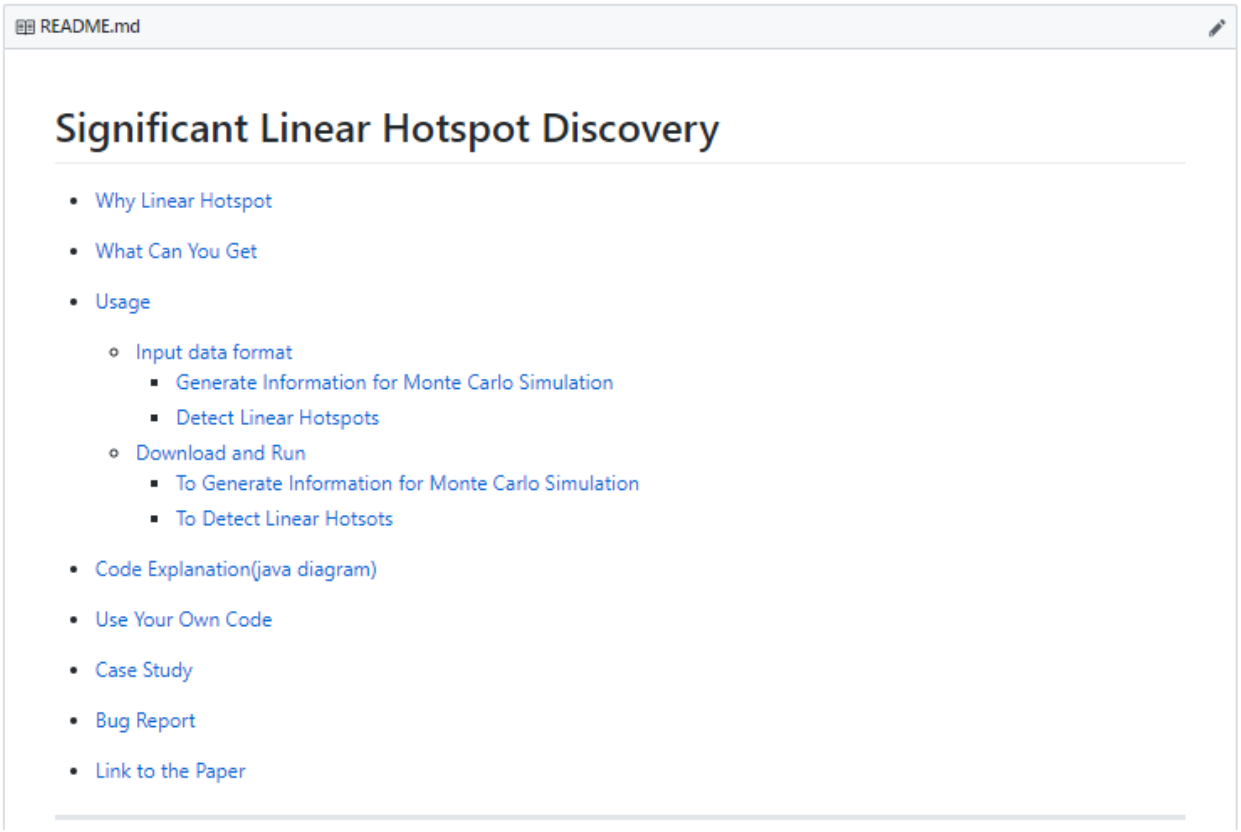


Figure 8 Third Version of GitHub - Outline

For the algorithm that has multiple ways to discover the hotspots, branches are created with clear instruction for each method. Users can switch branches as shown in Figure 9. No redundant code or instruction in each branch in this version, only the content that is relevant to the labelled method is shown on each branch. In this version, a link to the paper is also added at the end for academic users' reference.

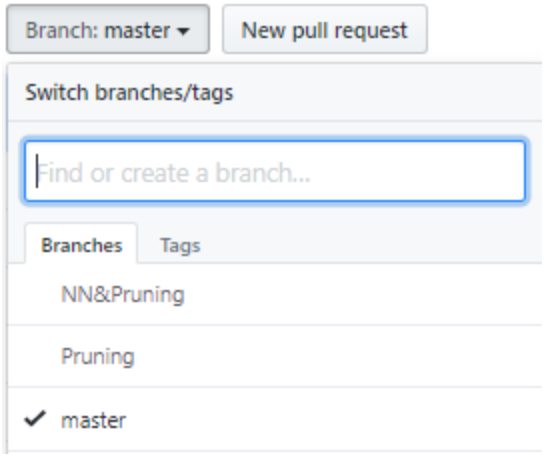


Figure 9 Third Version of GitHub – Branch Switch

3 CASE STUDY 1: ELLIPTICAL HOTSPOT DETECTION (EHD) REPOSITORY

3.1 ALGORITHM INTRODUCTION

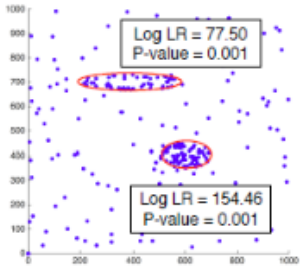
The very first-time user clicks into a repository page, the outline is shown followed by a visualized hotspot result. In the EHD repository, users will see Figure 10.

The outline can take users direct to the chosen section or the wiki page. The first bullet point is the introduction part. In What Can You Get section, there are two elliptical hotspots detected by the algorithm proposed in the paper based on 273 activities, with a p-value threshold of 0.001. It tells users what the result of this algorithm looks like. Three hyperlinks are contained in this part, shows application domain, basic concepts of the algorithm, and the algorithm explanation. It provides more detail to academic and advance users. Those details are provided using wiki pages. All the wiki page in this repository is listed in Figure 11. This is a tool for users who want to have a deep understanding of the algorithm.

Elliptical Hotspot Detection

What Can You Get

Elliptical Hotspot Detection (EHD) finds ellipse shaped hotspot areas where the concentration of activities inside is significantly higher than the concentration of activities outside.



See application domain to see where you can use elliptical hotspot detection.
Basic concepts can help better understanding the problem.
For academic users, see here for algorithm explanation.

Figure 10 EHD Repository Algorithm Introduction Part

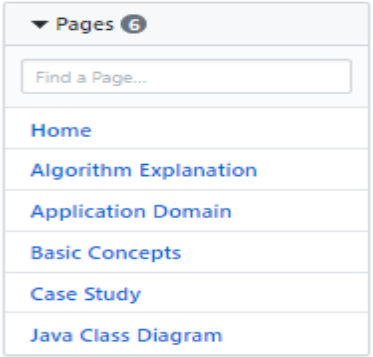


Figure 11 EHD Repository Wiki Page List

3.2 USAGE

The instruction for users to execute their own dataset is shown here. It also contains two parts. Data format and how to run. Each part has another two sub-parts. Data format introduces input and output data. The input file format is displayed in Figure 12. An example file: activity, is provided for users to run a test.

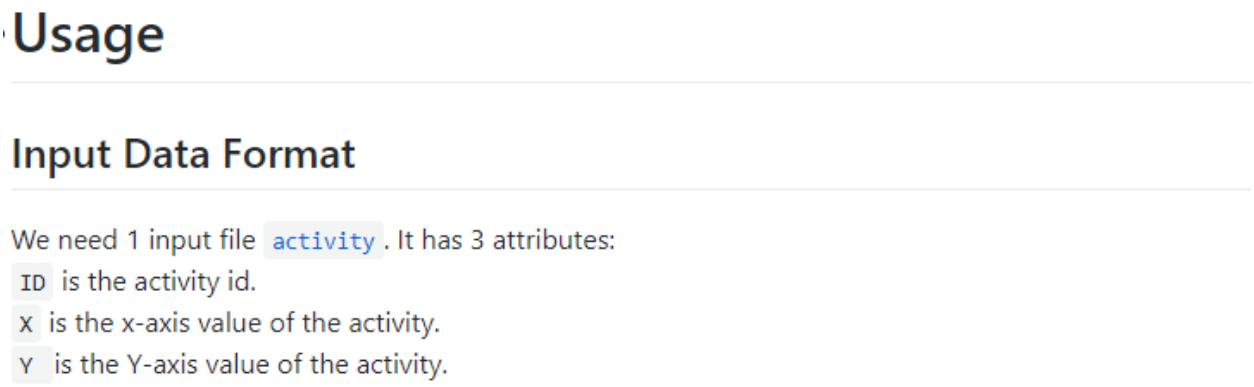


Figure 12 EHD Repository Usage Part – Input Data Format

Once the input data is ready, the executing step is listed in the Download and Run part. Since this EHD algorithm has two methods: Grid approach and naïve approach, users can switch branch as showed in Figure 13, to use different methods. Master branch contains the grid approach.

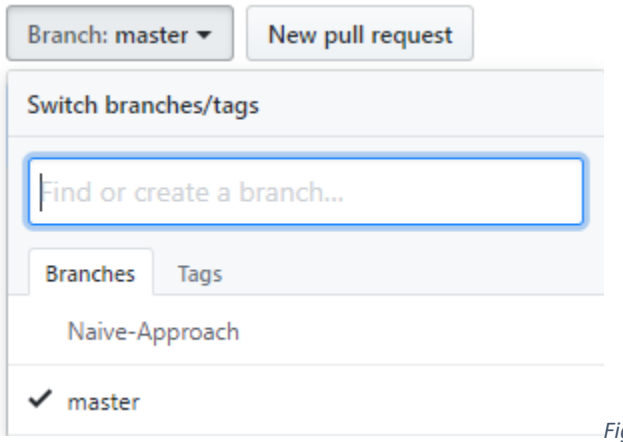


Figure 13 EHD Repository Switch Methods Through Branch

When choosing master branch to use the grid approach, there are two variables that need to be set. The result will pop up in terminal with all the dataset information. Details can be found in Figure 14.

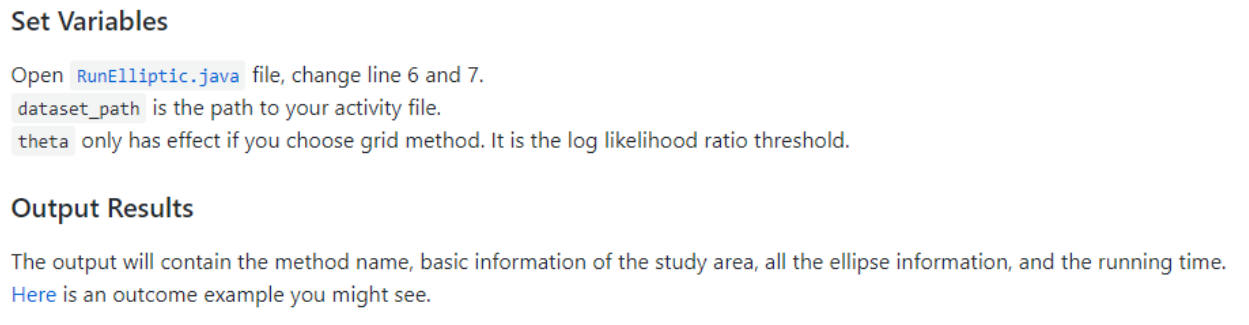


Figure 14 EHD Repository Run Process Under Master Branch

If users choose the naïve approach on the other branch, there are three variables that need to be set, and the result will be put into a file, contains all the ellipses that found by the approach as well as the activities information within each ellipse. The execution detail of this branch is shown in Figure 15

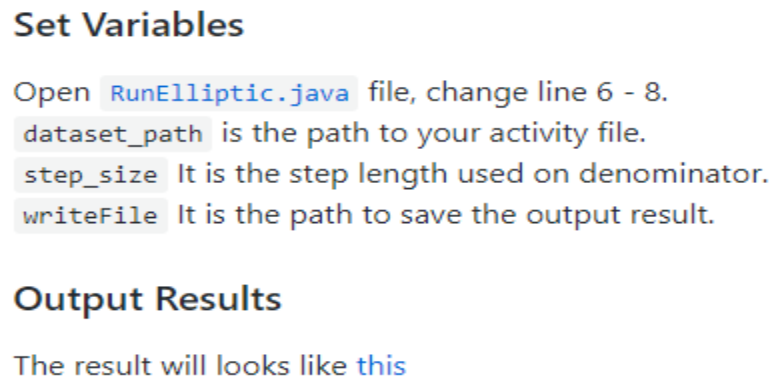


Figure 15 EHD Repository Run Process Under Naïve-Approach Branch

3.3 CODE EXPLANATION

Java diagram is posted on wiki page to help users understand the relationship between each class in this method, also shown in Figure 16.

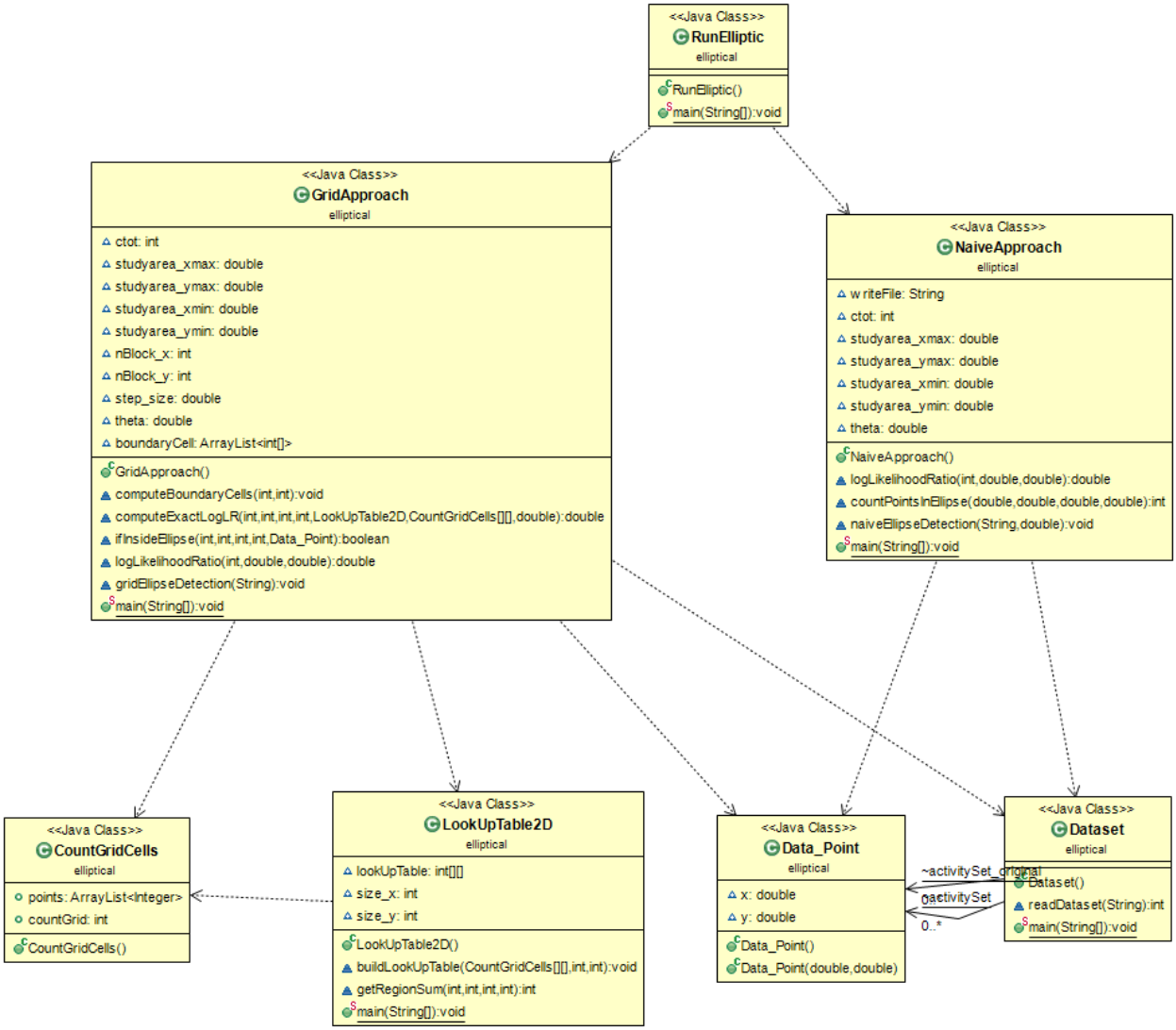


Figure 16 EHD Repository Java Class Diagram

3.4 SUPPORT SECTION

There are three parts in this section: Case study, bug report and the link to the paper. The case study compares the output quality of SatScan and FastEHD on a real-world crime dataset. This dataset contains 3181 crime records of unarmed robbery that occurred in the downtown area of Denver, Colorado. The detail can be found through the hyperlink on a wiki page.

Bug report is used for users to ask questions and report bugs (Figure 17). The link to the paper helps academic users to learn the method well.

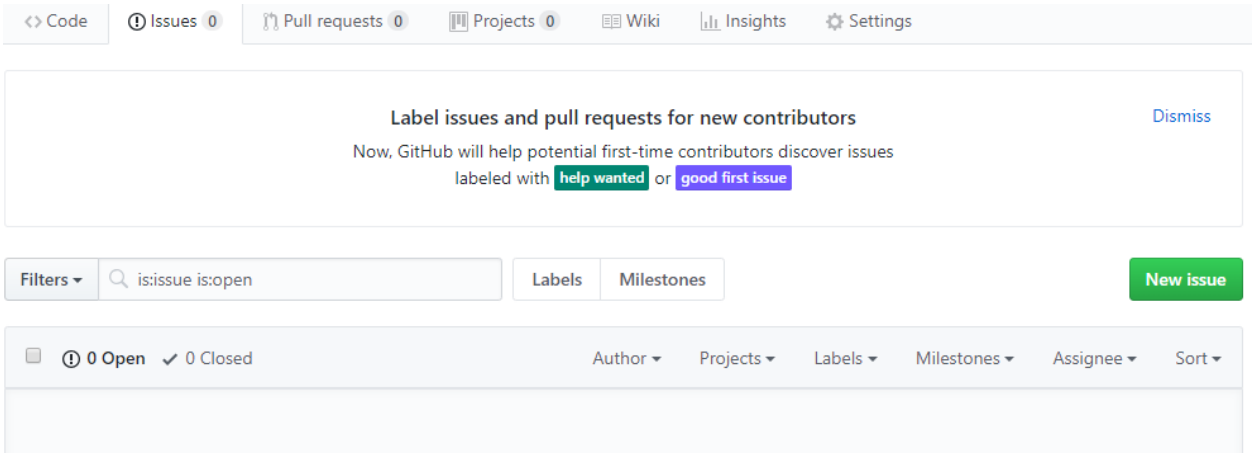


Figure 17 Bug Report Page

4 Case Study 2: Significant Linear Hotspot Discovery (SLHD) Repository

4.1 Algorithm Introduction

From Figure 18 we can see a clear structure of this SLHD repository. The outline, like the previous two case studies, is shown at the beginning. Compare with the previous two version, SLHD repository has the section called Why Linear Hotspots right after the outline. It shows the motivation of developing this algorithm, which aims to solve two problems among urban computing area: transportation planning and public safety and security. Details can be found in wiki page through the hyperlink. Researchers can find the details of our motivation there and learn the importance of identify accident prone road segment.

README.md

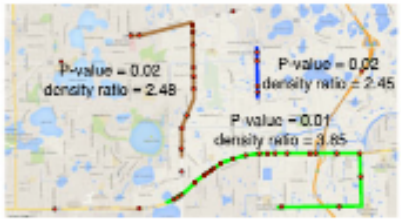
Significant Linear Hotspot Discovery

Why Linear Hotspot

Linear Hotspot is a line shape area of significant activity. Our motivation of developing linear hotspot detection method is to identify accident prone road segment.

What Can You Get

You will get the shortest path between two activities. The shortest path has high density ratio that pass the threshold, so we can call it a potential hotspot. Then Monte Carlo simulation is executed. Those potential hotspots that have low p-value will get picked to be the linear hotspots.



Basic concepts can help better understanding the problem

For academic users, see here for algorithm explanation.

Figure 18 SLHD Repository Algorithm Introduction Part

In What Can You Get section, there are three linear hotspots detected by the algorithm proposed in the paper based on 43 pedestrian fatalities (represented as dots) in Orlando, Florida occurring between 2000 and 2009. Two hyperlinks are contained in this part, shows basic concepts of the algorithm and the algorithm explanation. It provides more detail to academic and

advance users. Those details are provided using wiki pages. All the wiki page in this repository is listed in Figure 19.

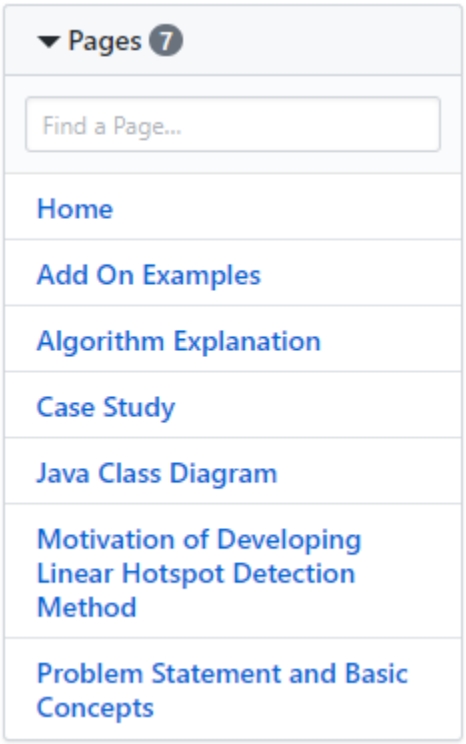


Figure 19 SLHD Repository Wiki Page

4.2 USAGE

The instruction for users to execute this algorithm on their own dataset is shown here. It also contains two parts. To start, users have to have their input file ready. The format of the input file is illustrated clearly in the first section, which is also shown in Figure 20. In this SLHD repository, users have to execute two files to get the final linear hotspots result. So, the Input Data Format part has two sections to illustrate the format for two execution files. Sample input data is prepared for users to run test, which can be accessed through the links.

Input data format

Generate Information for Monte Carlo Simulation

- We need 2 input files: `Node` and `Edge`:
  Node is the road intersaction file that has three attributes: Node ID, Longitude and Latitude. Attributes are separated by `;`
  Edge is the road segment file that has four attributes: Edge ID, Start node id, End node id, and Distance between start node and end node (weight). Attributes are separated by `;`

Detect Linear Hotspots

- We need 3 input files: `Node`, `Edge` and `Activity`:
  The `Node` and `Edge` file formats are the same as the files mentioned in previous session with same names.
  Activity is the car accident file (or whatever interesting events in the research) that has five attributes: Activity ID, Longitude, Latitude, Edge start node id and Edge end node id. The edge here is the one that contains the activity. Attributes are separated by `;`
- The output file contains linear hotspot road segment:
  result has 4 attributes:
  Activity 1 id, Activity 2 id, Shortest path between two activities and Density ratio

Figure 20 SLHD Repository Data Input Format Section

After having all the input files ready, users can follow the Download and Run sections (Figure 21) set all the variables and run the algorithm. To get the linear hotspots result, firstly users need to run Monte Carlo simulation multiple times to generate activities. The number of activities that need to be generated equals to the number of activities in the users input activity file. In each Monte Carlo simulation, there is a function to calculate the density ratio between each activity pair. The highest density ratio will be returned in each simulation. Users will get a sorted array that contains all the highest density ratio in each simulation. This will be used again later to calculate p-value. Then users can find all the density ratios between activity pairs in their own activity data file. Put each density ratio in the array we got before to calculate the p-value. If the p-value is less than the set threshold, then the path between that activity pair is a linear hotspot path.

Download and Run

How to import a GitHub project into Eclipse

To Generate Information for Monte Carlo Simulation:

After import project into Eclipse, find the GetHotspots.java file, change line 6 - 11
`nodePath` and `edgePath` are the path to your node and edge files.
`outputActivityFilePrefix` is the path to the folder that you will save all the generated activity files from Monte Carlo simulation.
`filenumber` is the number of time to run Monte Carlo simulation. `result` is the path to save the final outcome of linear hotspots.
`nNewActivities` is the number of total activities in your activity file.
`pValue` is the p-value that measure the significent of your hotspots.

Then run GetHotspots.java file.

To Detect Linear Hotsots:

Find the RunLinear.java file, change line 4 and 5. Set the `activityPath` to your activity file that probably contain hotspots. Change `result` to where you want the final result to be put. Finally run RunLinear.java file

Figure 21 SLHD Repository Download and Run Section

4.3 CODE EXPLANATION

Java diagram is posted on wiki page to help users understand the relationship between each class in this method, also shown in Figure 22.

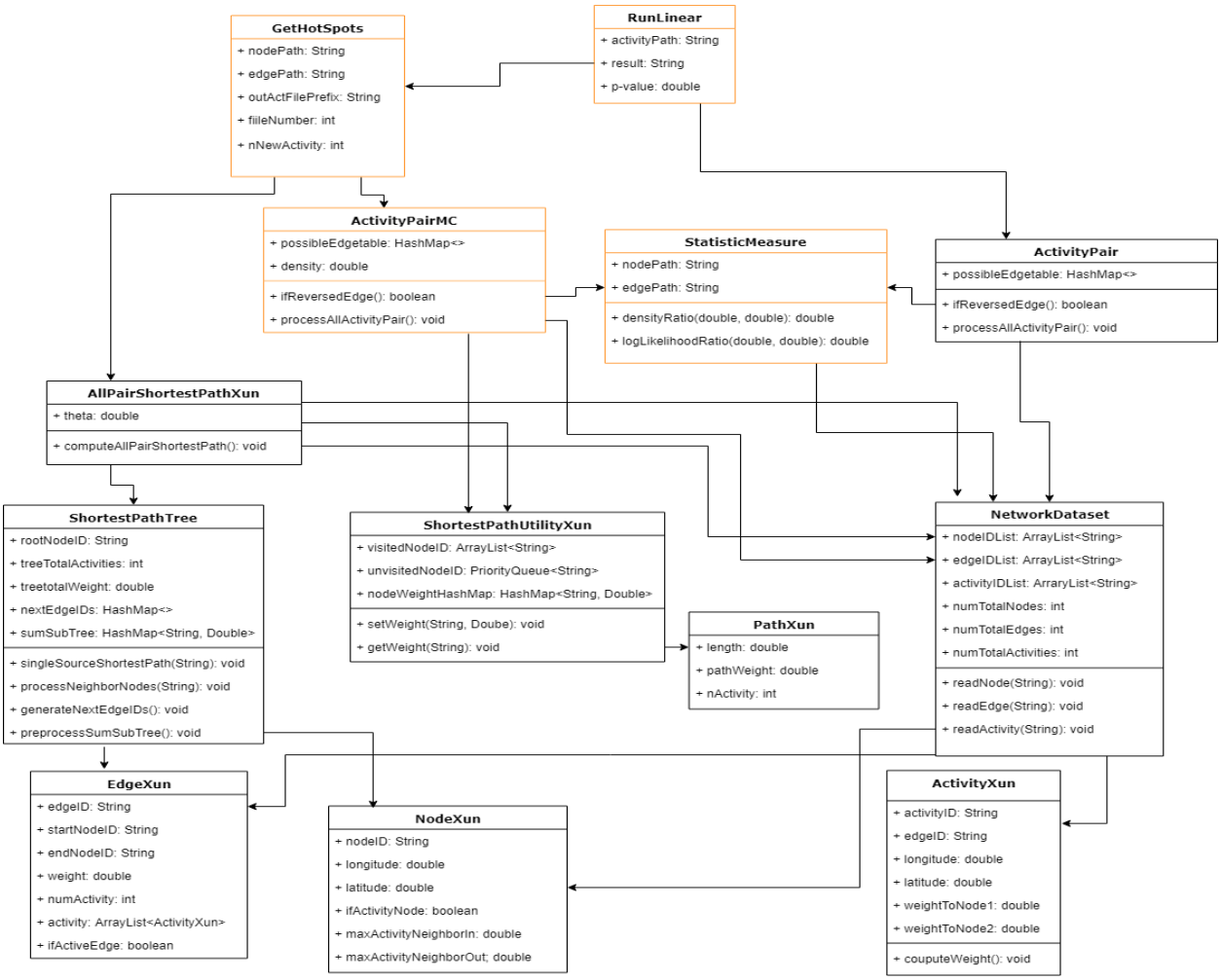


Figure 22 SLHD Repository Java Class Diagram

The code has been modularized. I re-design the relationships between classes, add classes to avoid manually generating result separately. The Average Cross Class Link reflects the dependency between classes. The goal is to reduce the dependency. The posted diagram in Figure 22 has reduced average cross class links. The original diagram is shown in Figure 23. The compare table is showing the calculation details in figure 24.

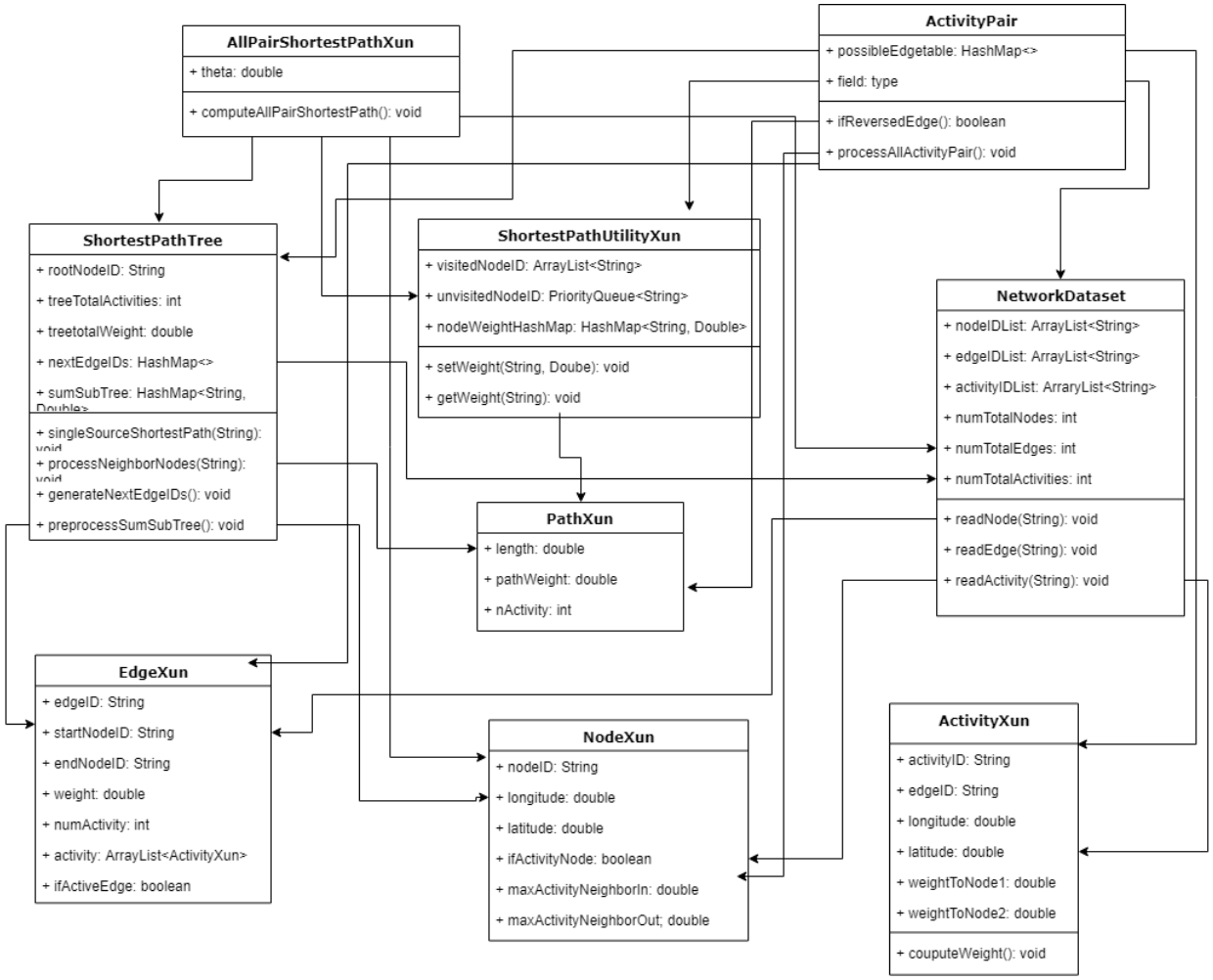


Figure 23 SLHD Repository Java Class Diagram (Before)

	Number of Class	Number of Link	Average Cross Class Link
Before	9	18	2
After	13	20	1.5

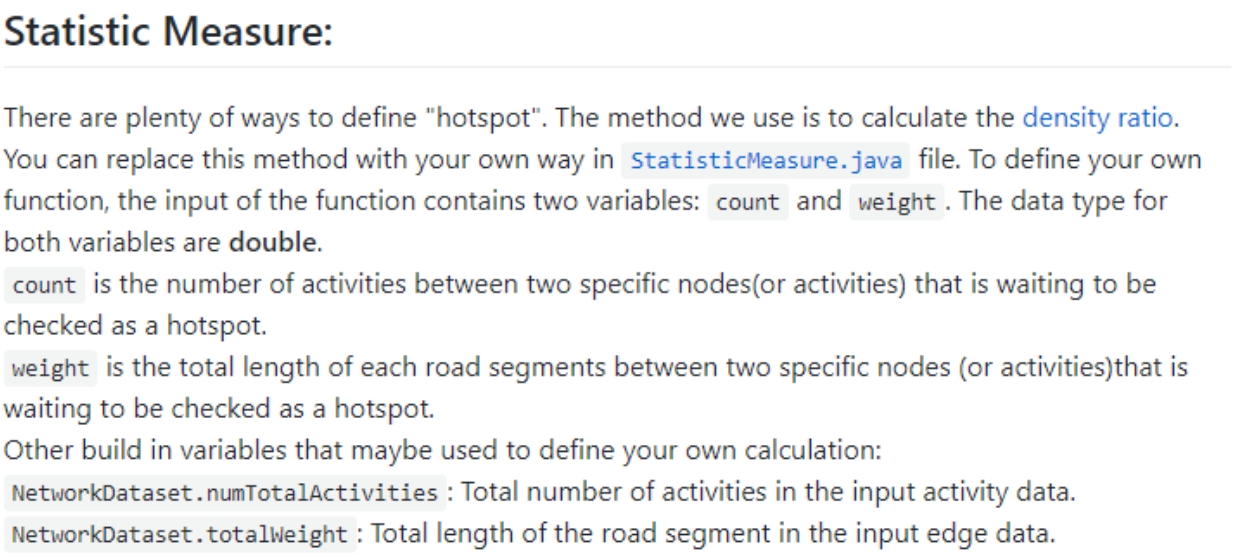
Figure 24 Compare Table of Two Diagrams

4.4 USE YOUR OWN CODE

Detecting linear hotspots is a multi-step process. The core methods are displayed using branch as mentioned before. However, some sub-methods can be manipulated by users. Advance

users can change selection methods and categories to get different linear hotspot for their own purpose.

One example of using users-defined-function is the StatisticMeasure.java file. In this file, we can define what we mean by “hot” in hotspots detection. The method that was used in the original paper was to calculate the density ratio. That is also the method we used on GitHub page. However, there are plenty of ways to define hotspots, which is the interesting patterns users want to find. For example, log likelihood is another way to do the statistic measurement. The hyperlink will take users to one of the wiki pages, which has clear explanation of what the users need to do to replace this part from the entire project. Figure 25 is the instruction.



Statistic Measure:

There are plenty of ways to define "hotspot". The method we use is to calculate the density ratio. You can replace this method with your own way in `StatisticMeasure.java` file. To define your own function, the input of the function contains two variables: `count` and `weight`. The data type for both variables are **double**.

`count` is the number of activities between two specific nodes(or activities) that is waiting to be checked as a hotspot.

`weight` is the total length of each road segments between two specific nodes (or activities)that is waiting to be checked as a hotspot.

Other build in variables that maybe used to define your own calculation:

`NetworkDataset.numTotalActivities` : Total number of activities in the input activity data.

`NetworkDataset.totalWeight` : Total length of the road segment in the input edge data.

Figure 25 SLHD Repository Instruction of How to Replace the Statistic Measurement with Users Defined Function

Another part that users can use whatever methods they want is the data cleaning part. The input data format is mentioned in Chapter 6.2. Users can use any methods, even different software to get the standard format. One example is been provided in Figure 26.

Data Cleaning:

The input format of `Activity` file has 5 attributes: Activity ID, Longitude, Latitude, Edge start node id and Edge end node id. The edge here is the one that contains the activity. Attributes are separated by `;`
In some cases, the activity file is separate with the road net work file. In other word, the activity file will only contain it own location information but no edge information. In this case the `OriginalActivity` file only has 3 attributes: Activity ID, Longitude and Latitude, separated by `;`.
MapMatching add on can help prepare a readable `Activity` file. Open the `MapMatching` file, change Line 12 - 15.
`nodePath` is the path to the node file.
`edgePath` is the path to the edge file.
`activityPath` is the path to the original activity file.
`activityPathWrite` is the path of the output.

Figure 26 SLHD Repository Data cleaning instruction

4.5 SUPPORT SECTION

This section also contains three parts: Case study, bug report and the link to the paper. The case study in this SLHD repository is conducted on a dataset of 1,529 simple assaults in Denver, Colorado during 2015 with a p-value threshold of 0.01. The rest parts are the same as previous case study.

5 CASE STUDY 3: GEOGRAPHICALLY ROBUST HOTSPOT DETECTION (GRHD) REPOSITORY

5.1 ALGORITHM INTRODUCTION

The first bullet is the introduction part of this GRHD repository, detail is shown in Figure 17. What Can You Get section is a highly generalized summary for ordinary users. The graph gives users a direct idea of what they can get from this method. It is an output example visualized on map. The blue hotspot points out the location of the infected water pump based on

1854 London Cholera Outbreak data. The application domain and basic concept links will take users to wiki pages. The wiki pages in this repository are the same as those in the EHD repository.

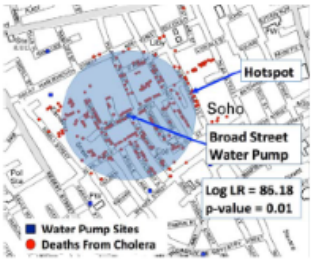
README.md

Geographically Robust Hotspots Detection

- What Can You Get
- Usage
  - Data Format
  - Download and Run
    - How to import a GitHub project into Eclipse
    - Set Variables
    - Output Result
- Code Explanation - Java Diagram
- Case Study
- Bug Report
- Link to Paper

What Can You Get

Geographically Robust Hotspot Detection (GRHD) finds hotspot areas where the concentration of points inside is significantly high.

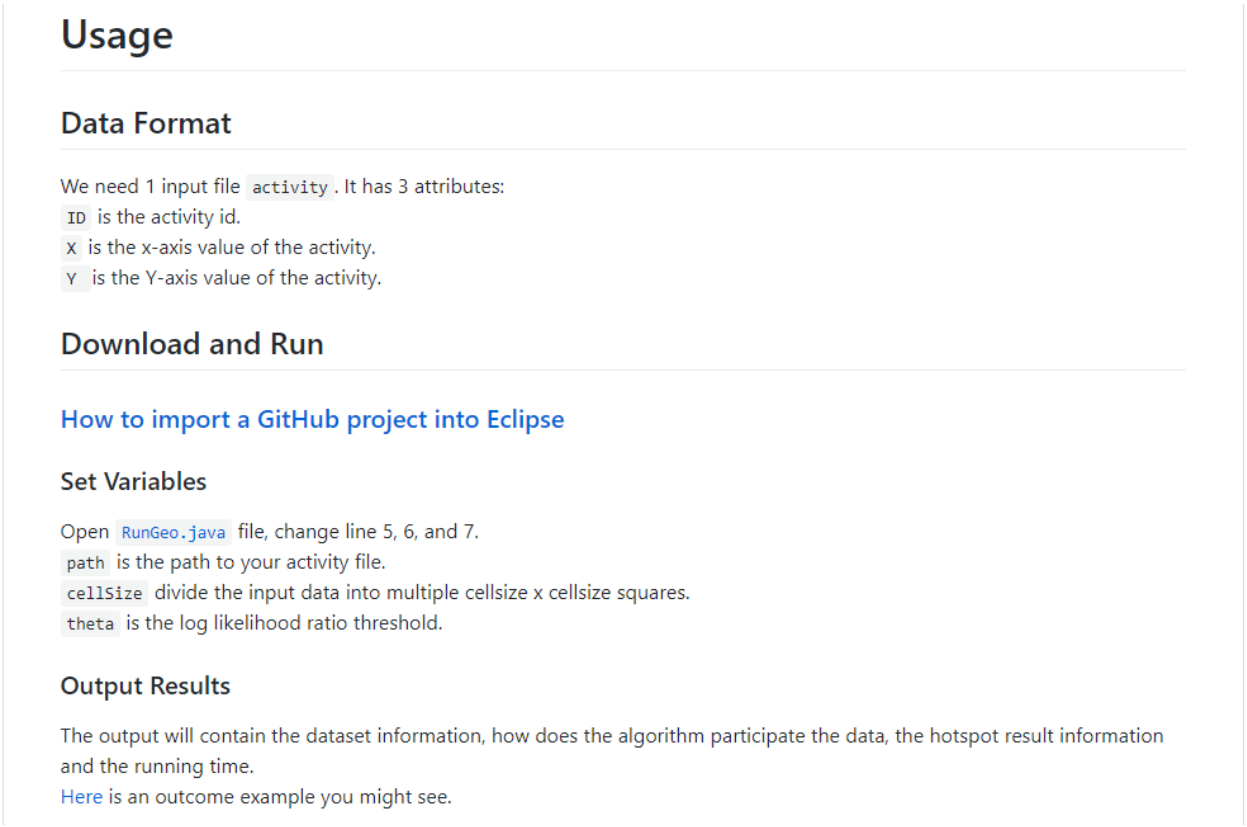


See application domain to see where you can use GRHD.
Basic concepts can help better understanding the problem.

Figure 27 GRHD Repository Algorithm Introduction Part

5.2 USAGE

The usage section in this GRHD repository also contains two parts: Data format and How to run. The input data format is the same as the input data used in EHD repository mentioned in chapter 3.



Usage

Data Format

We need 1 input file `activity`. It has 3 attributes:
`ID` is the activity id.
`X` is the x-axis value of the activity.
`Y` is the Y-axis value of the activity.

Download and Run

How to import a GitHub project into Eclipse

Set Variables

Open `RunGeo.java` file, change line 5, 6, and 7.
`path` is the path to your activity file.
`cellSize` divide the input data into multiple cellsize x cellsize squares.
`theta` is the log likelihood ratio threshold.

Output Results

The output will contain the dataset information, how does the algorithm participate the data, the hotspot result information and the running time.
Here is an outcome example you might see.

Figure 28 GRHD Repository Usage Part

In the Download and run part, users can find RunGeo.java file in the code file list. The variables that need to be changed are listed in Figure 28. CellSize and theta variables are introduced in the basic concept wiki page.

The output file contains the dataset information, how does the algorithm participate the data, the hotspot result information and the execution time of this algorithm. An example of the output result can be found through the hyperlink.

5.3 CODE EXPLANATION

There are seven java classes in this repository. Their relationship is shown in the diagram. Naïve SatScan is a test case and the RunGeo is the trigger for the algorithm. Detail can be found in Figure 29.

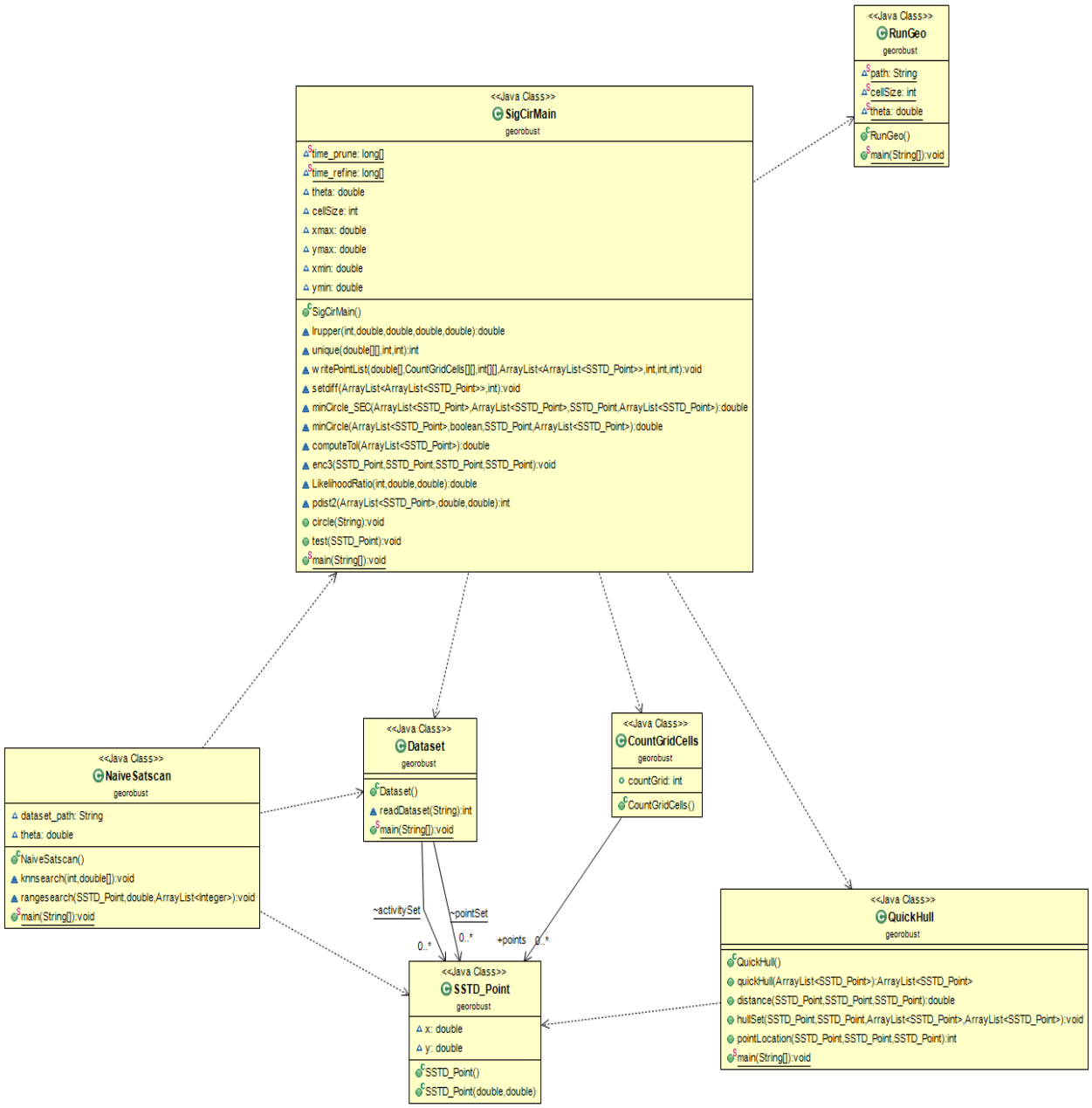


Figure 29 GRHD Repository Java Class Diagram

5.4 SUPPORT SECTION

The support section contains the same three parts as the previous repository. The case study in this GRHD repository used a crime dataset. The input point set includes 64 unarmed robbery cases in San Diego between March 2013 – 2014.

6 CONCLUSIONS AND FUTURE WORK

This project publishes lab achievements on GitHub page in relation to open science domains, meet the open data, open source and open access categories. I cleaned and reorganized the original lab code, modified the code structure for advance users to choose different methods through branch function on GitHub and add their ways by following the Use Your Own Code instruction on GitHub to achieve their own research goals. I also designed the GitHub page to make sure it is both easy to follow by users who do not have strong computer science background as well as provides enough details for academic users to refer this method in their own academic paper. I had three-version of designs to achieve all these goals. The summary of these three versions is shown in Table 1

First Version (baseline)	Second Version	Third Version
Provide a brief introduction of linear hotspot as well as a link to the motivation of developing this algorithm. Followed by problem statement, basic concepts definition and the algorithm explanation. A usage session is provided to guide users on how to use the algorithm	Replace the problem statement with a case study output visualized on road map. Problem statement is moved to wiki page. Show the expected result in README file for users to understand the outcome. Reorganize code, make replaceable sessions for academic users to replace with their own methods	Content is added at the beginning with hyperlinks to guide all users. Use different branches to show different methods if applicable. No redundant code or instruction in each branch. Link to the paper is added at the end for users to reference.

Table 1. Compare Three Versions of GitHub Page Design

By the time the final version of this project completed, two students from a different class in GIS department were interested in the method and were able to use it follow the GitHub page. The GitHub link provided an easy way to share and explain the code demonstrating the usefulness. The GitHub link make the lab code accessible to ordinary users.

In future work, the goal is to achieve reproducible research. The term reproducible research refers to the idea that the ultimate product of academic research is the paper along with the laboratory notebooks and full computational environment used to produce the results in the paper such as the code, data, etc. that can be used to reproduce the results and create new work based on the research [7]. I will create code developing guide for researchers in the lab, so the code follows a specific format at the beginning. We will also continue contributing to open science by publishing the lab achievements on the GitHub page. After the algorithm has been developed followed by standard, the code can be published online with only little modification. Thus, the lab achievement can be reproduced by people who are interested in the same topic even without a strong computer science background.

7 REFERENCES

[1] R. Vicente-Saez and C. Martinez-Fuentes, Open Science now: A systematic literature review for an integrated definition, vol. 88, València: Journal of Business Research, 2018, pp. 428-436.

[2] R. D. Peng, Reproducible Research in Computational Science, vol. 334(6060), Science, 2011, p. 1226(2).

[3] F. Thung, T. F. Bissyande, D. Lo and L. Jiang, Network Structure of Social Coding in GitHub, Genova: 17th European Conference on Software Maintenance and Reengineering, 2013, pp. 232-236.

[4] E. Eftelioglu, X. Tang and S. Shekhar, Geographically Robust Hotspot Detection: A Summary of Results, IEEE 15th International Conference on Data Mining Workshops, 2015.

[5] X. Tang, E. Eftelioglu, D. Oliver and S. Shekhar, Significant Linear Hotspot Discovery, vol. 3, IEEE TRANSACTIONS ON BIG DATA, 2017.

[6] X. Tang, E. Eftelioglu and S. Shekhar, Elliptical Hotspot Detection: A Summary of Results, New York: ACM, 2015, pp. 15-24.

[7] S. N. Goodman, D. Fanelli and J. P. A. Ioannidis, What does research reproducibility mean?, vol. 8, Science Translational Medicine, 2016, pp. 341-353.