Building Reproducible Workflows using Transportation Data and COVID-19 Data

Yuanzhi Ren

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2021

Committee:

Cynthia Chen

Ka Yee Yeung-Rhee

Xiangyang Guan

Ling-Hong Hung

Wes Lloyd

Program Authorized to Offer Degree:
Computer Science and Systems

University of Washington Tacoma

Abstract

Building Reproducible Workflows using Transportation Data and COVID-19 Data

Yuanzhi Ren

Co-Chairs of the Supervisory Committee:
Professor Cynthia Chen
Civil and Environmental Engineering

Professor Ka Yee Yeung-Rhee
School of Engineering and Technology

Human mobility analysis research, or the analysis of individuals' daily activity and travel patterns, is an important topic as the results on people's travel patterns are the basis of hundreds of billions' investment for the nation's transportation infrastructures. With ubiquitous mobile sensors, human mobility analysis has been increasingly performed with an enormous amount of mobile sensor data such as GPS location data generated from mobile devices. Transportation researchers have developed pre-processing and analysis methods to extract mobility patterns from those big mobile sensor data. These analytical methods are often used by transportation researchers who might not be trained in computer science. Thus, it is time-consuming and non-trivial to configure the computing environment required to execute these methods so as to reproduce published results.

This thesis develops the Mobility Analysis Workflow, a containerized software tool, to enhance reproducibility of research in the transportation community. Providing an interactive graphical user interface, the Mobility Analysis Workflow allows users to create and customize mobility analysis pipelines, also known as workflows, in which each graphical module represents a data processing task. Furthermore, we conducted benchmarking experiments to identify the computational bottlenecks of our case study workflow. In addition, we expanded

the functionality of the Mobility Analysis Workflow by creating new use cases in modeling the spread of the Coronavirus Disease 2019.

Acknowledgements

I would first like to express my sincere gratitude to Professor Cynthia Chen for her expertise and continuous support in my thesis project, and for the funding opportunity to undertake my thesis project. This thesis project is funded by US Federal Highway Administration (FHWA), the Washington State Department of Transportation (WSDOT), the US National Institute of Health (NIH) (1R01GM108731-01A1) and the Center for Teaching Old Models New Tricks (TOMNET), a University Transportation Center sponsored by the US Department of Transportation through Grant No. 69A3551747116.

I would like to extend my sincere thanks to Professor Ka Yee Yeung, whose immense knowledge is invaluable in formulating the research methodology. Her insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would like to acknowledge Dr. Xiangyang Guan for his suggestions and guidance throughout this thesis project.

I would like to thank Dr. Ling-Hong Hung and Dr. Wes Lloyd for their assistance at every stage of my thesis project.

I would like to thank Grace Jia for her contributions to my thesis project.

Glossary

Workflow
A sequence of widgets where internal connections between the widgets specify the order of execution and data flow to constitute a multi-step computational job.

Widget
The graphical front end of a container in the Biodepot-workflow-Builder (BwB). A self-contained graphical execution module containing the commands, executables, dependencies, and interface. Each widget represents a modular computational task.

Biodepot-workflow-Builder (BwB)
Biodepot-workflow-Builder (BwB) [2] is a software tool which provides a graphical user interface (GUI) to build bioinformatics workflows using Docker containers.

Compartmental model
The compartmental model is a type of mathematical model for the spread of infectious diseases. The model comprises several compartments, each representing a population segment with a certain disease status (e.g., susceptible, infectious or recovered), models how the number of people in each compartment changes as the disease spreads over time.

COVID-19
Coronavirus disease 2019 (COVID-19) is a contagious disease caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2).

Docker container
A container instance which has a running environment provided/pre-defined by the Docker image.

Docker image
An image file which encapsulates the dependencies of a software application that is then instantiated into a running instance contextualized by the Docker engine. A Docker container can be created and launched based on a Docker image.

Epidemiology
Epidemiology is the study and analysis of the distribution (who, when, and where), patterns, and determinants of health and disease conditions in observed populations [5].

Reproducibility
Reproducibility is a major principle of scientific methods. It means that the result obtained by one study can be reproduced when the study is carried out with the same data and method by other researchers [47].

SEIR model

The Susceptible, Exposed, Infectious, Recovered (SEIR) model [48] is a typical compartmental model in epidemiology which divides the observed population into four compartments, the “Susceptible” compartment, the “Exposed” compartment, the “Infectious” compartment, and the “Recovered” compartment.

SIR model

The Susceptible, Infectious, Recovered (SIR) model [49] is a typical compartmental model in epidemiology which divides the observed population into three compartments: “Susceptible”, “Infectious”, and “Recovered”.

Chapter 1

Introduction

1.1 Background

In data science, software and data processing tools help researchers to analyze experimental data. Generally, a data processing pipeline, also known as a workflow, consists of several analytical steps, such as clustering of data points and visualization of results. Ideally, after the workflow is designed and implemented, researchers should be able to upload their experimental data to the software tool and execute the workflow. As a result, when tasked with analyzing new empirical data, researchers can focus on studying the experimental results rather than building the computational workflow from scratch.

Reproducibility, defined as a property that different researchers can generate the same results following the same experimental setting and procedure, plays an important role in research [12]. In general research practices, differences in experimental and environment configuration and different versions of required software tools can potentially lead to different research results. When upgrading a data analysis pipeline, dependencies and running platforms can change, which can result in reproducibility issues.

Software tools to enhance reproducibility of research, such as Jupyter [27], Github [50], Conda [9] and Docker [32], are utilized in data science research. Jupyter and Github facilitate the sharing of computer programs and softwares. However, due to complexities and dependencies of the required computing environments and insufficient documentation for some projects, it is often non-trivial to run the shared code. Conda, as a software package management tool, helps the users to manage the numerous libraries and packages of different versions from different programming languages, and to provide dependencies to reproduce the original research. Docker enables the containerization of the programs and softwares, which eliminates the need of manually installing the required dependencies. On the other hand, even though the dependencies are provided in Docker, the incorrect settings and input parameters of analysis methods could lead to the failure of reproducing the original results. A one-stop solution to enhance reproducibility of research is urgently needed.

In transportation engineering, reproducibility of research is still an exploratory area. Mobile sensor data contains the geological location and time information of when and where a person has been to. Processing mobile sensor data to infer human mobility patterns [19] [52] could facilitate policy decisions of when

and where a new transportation infrastructure, such as a parking lot, should be built. To process these mobile sensor data, each individual transportation research lab has its own data processing algorithms and pipelines, such as human mobility pattern extraction algorithms and human mobility trace clustering. After sharing with the transportation research community, the data processing algorithms and pipelines can be applied to the different cases according to different research needs. However, because of the lack of domain knowledge in programming and software dependency updates, it is non-trivial for the transportation research community to reproduce the analysis results.

In addition to reproducibility, another challenge for transportation researchers is that these mobile sensor datasets could be hundreds of gigabytes [72], thus the corresponding workflows are computational intensive. Most transportation engineering labs are not equipped with computer clusters for these types of big data analyses. Cloud computing services such as Amazon Web Service (AWS) [10] provides on-demand and scalable computational resources. These resources could be used to deploy computational intensive workflows using large datasets without any upfront hardware costs.

1.2 Contributions

The contributions of this thesis include the development of Mobility Analysis Workflow (MAW), a software tool to build transportation workflows, to enhance reproducibility of research on mobility analysis. Through benchmarking experiments on AWS, we studied performance trends and identified the computational bottlenecks of the implemented case study workflows. In addition, we expanded the functionality of Mobility Analysis Workflow to create workflows for the Coronavirus Disease 2019 modeling use cases to explore the possibility of applying the MAW beyond the scope of the transportation research.

Chapter 2

Related work

In this chapter, we review the related work in this thesis. Specifically, in Section 2.1, we review the related concepts discussed in Chapter 3; Section 2.2 and Section 2.3 cover the related methods used in Chapter 4; lastly, we discuss the related research which are relevant to Chapter 5 in Section 2.4.

2.1 Software Tools for Reproducibility in Data Science

Docker [32] is widely used in the data science research field and beyond to provide container running environments for the softwares and programs. As an example, the BioContainers project [3] creates Docker containers to package software tools for bioinformatics applications. As a Docker image storing registry and sharing platform, Docker Hub [33] facilitates the image sharing process within the Docker user community. Jupyter [27] project, including "Jupyter notebook", a visualizable and interactive code running environment, and "JupyterLab", a online version of "Jupyter notebook" that deployed on the server provided by Jupyter, provides a solution for executing code visually and interactively. Github [50] allows users to publish their code to share with the other users, which contributes to reproducibility of research. Conda [9] can help to manage the software packages with different versions from different programming languages, and support the program running with the required libraries. However, it requires knowledge in programming to use these software tools and platforms.

To solve this issue, Hung, Lloyd, Yeung and colleagues proposed a Docker based software tool called the Biodepot-workflow-builder (Bwb) [2]. The Bwb allows users to build and reproducibly execute modular bioinformatics workflows. A multi-step workflow in Bwb consists of multiple widgets, each of which will launch a Docker container in the backend to execute an analytical task. Through the graphical user interface (GUI) provided by Bwb, the users can easily run their own workflows by draggings and clickings without requiring any programming knowledge.

2.2 Geographical Distance Measurement

Measuring geographical distance between two geographical coordinates is a basic building block and is performed repeatedly in our case study workflow introduced in the Chapter 3. To identify the possible bottleneck of computing geographical distances in our case study workflow, we implemented and investigated the performance of three different geographical distance measuring methods. Specifically,

the Ellipsoidal Earth projected to a plane [61] distance function is a geographical distance estimation function which projects the spatial distance between two coordinates on an ellipsoid into a plane and calculates the distance using the Euclidean distance formula. The Spherical Law of Cosines [62] is a theorem defining the mathematical relationship of the side lengths and the degree of the angle of a spherical triangle.

$$\cos c = \cos a \cdot \cos b + \sin a \cdot \sin b \cdot \cos C \tag{2.1}$$

In Equation 2.1, we let $a,\ b,\ c$to be the three sides of a spherical triangle shown as Figure 2.1, and C to be the angle between side a and side b, then we can calculate $cos\ c$ using this equation. Further, we can compute side$c$using Inverse trigonometric functions. As a result, the distance between two endpoints of a spherical triangle can be estimated.

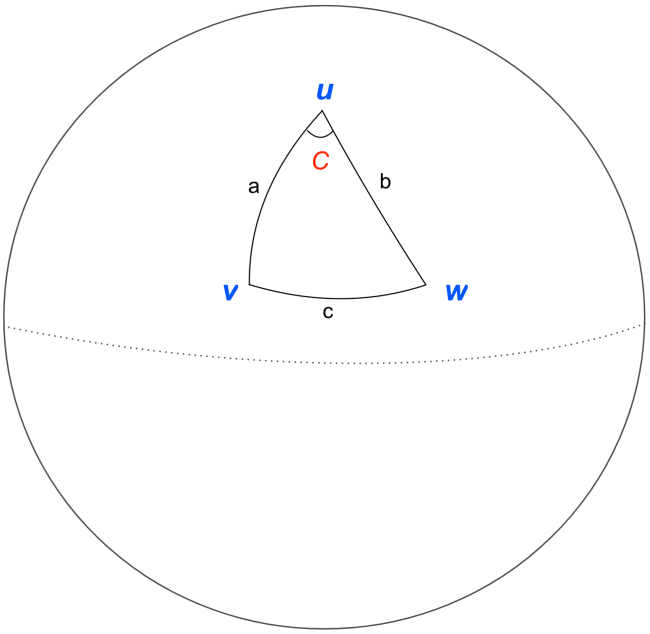

Figure 2.1: Spherical triangle Illustration

Similarly, the Ellipsoidal Earth projected to a plane (also called equirectangular projection) [63] distance function is to project the variation of latitude to the vertical line and the variation of longitude to the horizontal lines. After applying the Euclidean distance formula, the distance between the two geographical coordinates can be estimated.

2.3 Computational Resources Using the Cloud

Cloud computing services such as Amazon Web Service (AWS) [10] provides on-demand and scalable computational resources for business applications [14], as well as scientific research [11] [20]. AWS Elastic Compute Cloud (EC2) provides elastic computing resources, computing instances, for users to run their applications. Users can scale up or scale down their instance group(s) for their applications according to their demands. AWS Simple Storage Service (S3) is a file storage service for users to store and manage their files.

2.4 Modeling the Spread of COVID-19

In epidemiology, compartmental model [42] is an important tool in disease infection modeling. For a compartmental model [42], initially, every person is assigned to one “compartment” according to his/her disease status such as "susceptible", "infectious", and "recovered". Each “compartment” is a homogeneous entity, in which the persons are regarded as identical. With external factors such as close contact with disease hosts, the disease status of the persons may change. As a result, these persons will be moved to a new compartment. As an example, when a person becomes infectious, he/she moves from the S (susceptible) compartment to the I (infectious) compartment.

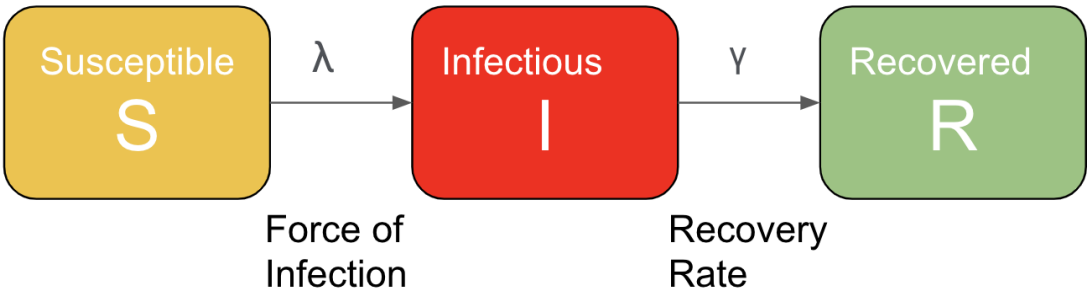

Figure 2.2: Illustration of SIR model

Figure 2.2 shows a typical compartmental model in epidemiology called the Susceptible, Infectious, Recovered (SIR) model [49] which divides the observed population into three compartments, the “Susceptible” compartment, the “Infectious” compartment, and the “Recovered” compartment. Every person is assigned to one compartment according to his/her disease status at the beginning of the experiment. The “Susceptible” compartment consists of people who are susceptible to the disease. People

from the “Susceptible” compartment are moved to the “Infectious” compartment at the force of infection rate λ, a percentage that the susceptible people become infectious. The “Infectious” compartment is a group of people who are infectious. People from the “Infectious” compartment are moved to the “Recovered” compartment at the recovery rate γ, a percentage at which infectious people recover from the disease. The “Recovered” compartment only contains people who recovered from the disease and are immune to the disease.

Compartmental models have been widely used in modeling the spread of COVID-19. To give examples, Peng et al. [43] proposed a generalized compartmental model to predict the inflection point of the epidemic curve and ending times in the major cities in China. The conclusion is that the epidemics in Beijing and Shanghai would end by March 2020. With a modified version of the compartmental model, Yang et al. [44] tried to derive the epidemic curve using 2019 population migration data and epidemiological data in mainland China, and showed that the peak of the epidemic in China would appear in late February, 2020. Tuite et al. [45] employ an age-structured compartmental model of COVID-19 transmission of the population of Ontario, Canada, and estimate that 56% of the local population would be infected in the epidemic. With the common efforts which are devoted in vaccine development, vaccination becomes an available method to fight against COVID-19. Xia et al. [64] studied the age-based vaccination strategies of different countries considering each country has the unique population construction using the SEIR model, and evaluated their effectiveness.

Chapter 3

Mobility Analysis Workflows (MAW) : support reproducible and modular mobility workflows

3.1 Transportation Data

In this study, mobile sensor data (data generated by users' usage of their various apps on their mobile devices), provided by Professor Chen's THINK Lab (https://sites.uw.edu/thinklab), is used. We make use of this data to build Mobility Analysis Workflows (MAW) that can be deployed on the local host or the cloud. To illustrate this application, we provide examples of mobility sensor data in Table 3.1.

Table 3.1 shows selected examples of synthetic mobile sensor data in which each row records the time ("Timestamp" column) when the user ("Device ID" column) used the mobile application in a specific location ("Latitude", "Longitude" column) with a location accuracy [15]. All the data used in this thesis project were collected within the Puget Sound region, WA, US.

Time stamp	Device ID	Latitude	Longitude	Location accuracy (meters)
1491398264	4ab844ff98c206b8d7	47.9205809	-122.2535626	5
1491403834	4ab844ff98c206b8d7	47.9229781	-122.2903396	25
1491403961	4ab844ff98c206b8d7	47.9222743	-122.2998663	60
1491412669	4ab844ff98c206b8d7	47.8994576	-122.2915348	60
1491412963	4ab844ff98c206b8d7	47.8856073	-122.2908753	300
1491413263	4ab844ff98c206b8d7	47.8850917	-122.2806468	1399

Table 3.1: Example synthetic mobile sensor data [15]

3.2 Background of Case study

As a case study of MAW, we developed a mobility workflow to process multi-sourced mobile sensor data generated by multiple positioning technologies. The workflow is based on the method developed by Wang et al. [52]. In general, the method first partitions input data into multiple subsets depending on their location accuracy. For example, data signaled by GPS usually have much higher accuracy than those by cellular tower triangulation. It then processes each subset independently to extract locations where an individual spends time. Finally, it integrates the processed data to form the mobility pattern for the individual.

3.3 Method

In this section, we introduce the terminology used, the design of each essential component and the implementation of the mobility workflow. Building on the Biodepot-workflow-builder (Bwb) platform [2], we design and implement new functionalities for transportation applications in this thesis.

3.3.1 Terminology Explanation

In MAW, a **module** is an independent computing unit which will execute a data processing task by executing the processing algorithms/tools, while a **Docker container** is an isolated running environment with the installed dependencies. A **widget** is a graphical representation of a module in the graphical user interface (GUI) of the MAW. A **workflow** is a sequence of modules/widgets where the internal connections between the modules specify the order of execution and data flow to constitute a multi-step computational job.

3.3.2 Modularization of mobility analysis method

To design a module in MAW based on a mobility analysis method, we first need to divide a mobility analysis method into several computational steps. Then, we divide the code of the mobility analysis method into different software modules according to the different functionalities of each procedure. For each software module, we specify the input data, output data and the parameters for the interface, such that each module can work independently with the input data from the previous module, and its output data would serve as the input data for the next module.

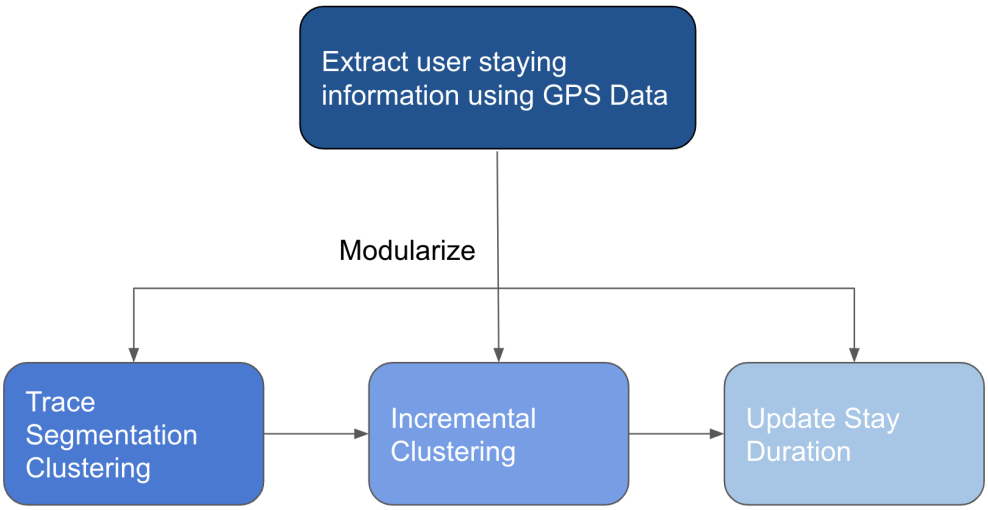

Figure 3.1: Modularization of the "extract user stay information using GPS data" method in MAW

We illustrate the modularization of the "extract user stay information using GPS data" method in Figure 3.1. As part of the method proposed in [52], the "extract user stay information using GPS data" method, as the name suggests, is to extract user stay information using GPS trace data. We modularize this method into 3 modules, "Trace Segmentation Clustering", "Incremental Clustering" and "Update Stay Duration". The three modules work sequentially to complete the mobility analysis task, extracting user stay information. The input and output data of the "extract user stay information using GPS data" method is in spreadsheet file format. To maintain the consistency of input and output data format, we kept this format for the three modules. The parameters of the "extract user stay information using GPS data" method includes "duration threshold" and "distance threshold". The duration threshold is used in all three modules and the distance threshold is used in the "Trace Segmentation Clustering" and "Incremental Clustering" modules. As a result, we rewrote the code to receive the passing parameters for each module.

3.3.3 Creation of containers

Following the modules we created, we create Docker containers for each module. Encapsulating the required software dependencies, Docker containers provide an isolated runtime environment for the execution of the computational task. Specifically, first we create a Docker image which contains the required dependencies and the programs of computational tasks, could be a set of Python scripts, for each module using the "docker build" command. Then we create Docker containers based on the Docker images to conduct the computational tasks using the "docker run" command.

As an example, we created Docker containers for the 3 modules as shown in Figure 3.1. For the “Trace Segmentation Clustering” module, we first built a Docker image that included the scripts to support running the “Trace Segmentation Clustering” job and encapsulated the dependencies to run the scripts. The scripts read in the input data and the passing parameters, “duration threshold” and “distance threshold”, and write the results as the output file. Then, we can create Docker containers based on the Docker images we created.

3.3.4 Workflow Development

Given the Docker containers and the Docker images created, we need to create the corresponding widgets using the GUI of MAW, and connect the widgets sequentially to build a workflow.

To create the widgets in MAW, we use the “New widget” functionality of MAW, shown in Figure 3.2, to define the configuration of one widget, including the widget name, the Docker image the widget based on, the input and output data path and the passing parameters.

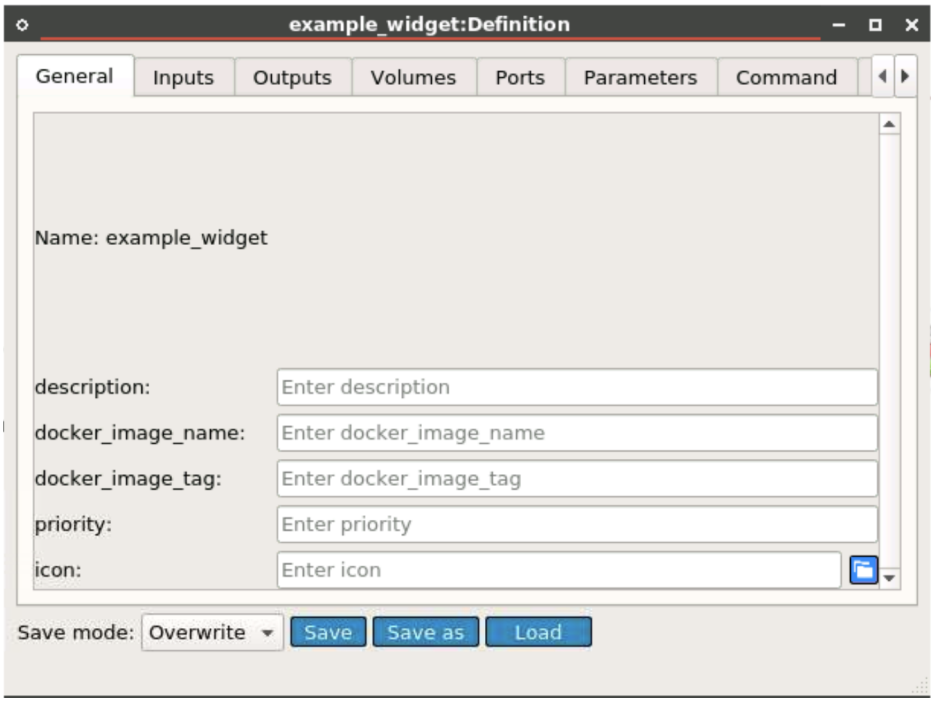

Figure 3.2: Create new widget in MAW

After all the widgets of their corresponding Docker containers are created, we can simply connect the widgets following the modularization of the original mobility analysis method.

3.3.5 Graphical User Interface of MAW

To provide a user-friendly access method, MAW provides an interactive GUI for users, such that users can work with MAW without knowledge in programming. By simple draggings and clickings, users can build and customize new workflows to conduct mobility analysis research. Note that the form-based user interface and drag-and-drop functionalities in the GUI of MAW are inherited from the Bwb platform [2].

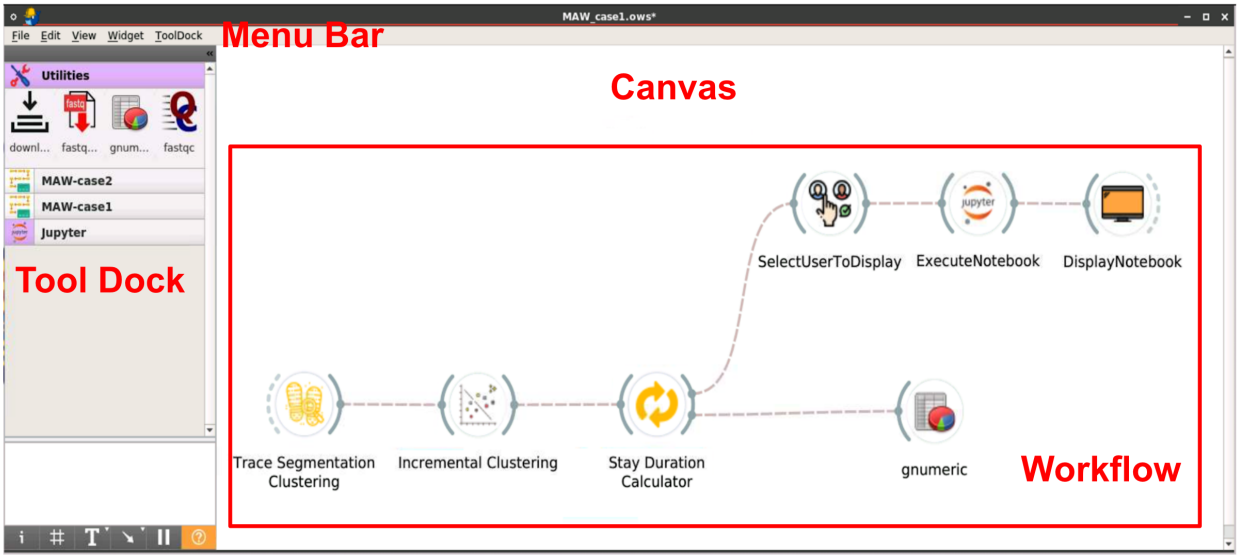

Figure 3.3: the GUI of MAW

Shown in Figure 3.3, the GUI of MAW consists of 3 components: the menu bar, tool dock and canvas. The menu bar presents many features of MAW, such as loading a new workflow, saving a workflow, and adding new widgets. The tool dock allows users to select and drag a new widget and drop at the canvas to build a new workflow. The canvas is the space where users can build and run the workflow. The shown workflow starts with the "Trace Segmentation Clustering" widget, which executes the trace segmentation clustering task by running a docker container in the backend. After the execution, the next widget will be triggered to conduct the incremental clustering task as the name suggested. In this way, the execution of a workflow will complete after all the widgets finish their tasks.

3.4 Results

Following the methodology discussed, We implemented the case study workflow of the mobility analysis method proposed in [52] to analyze mobile sensor data, shown in Figure 3.4.

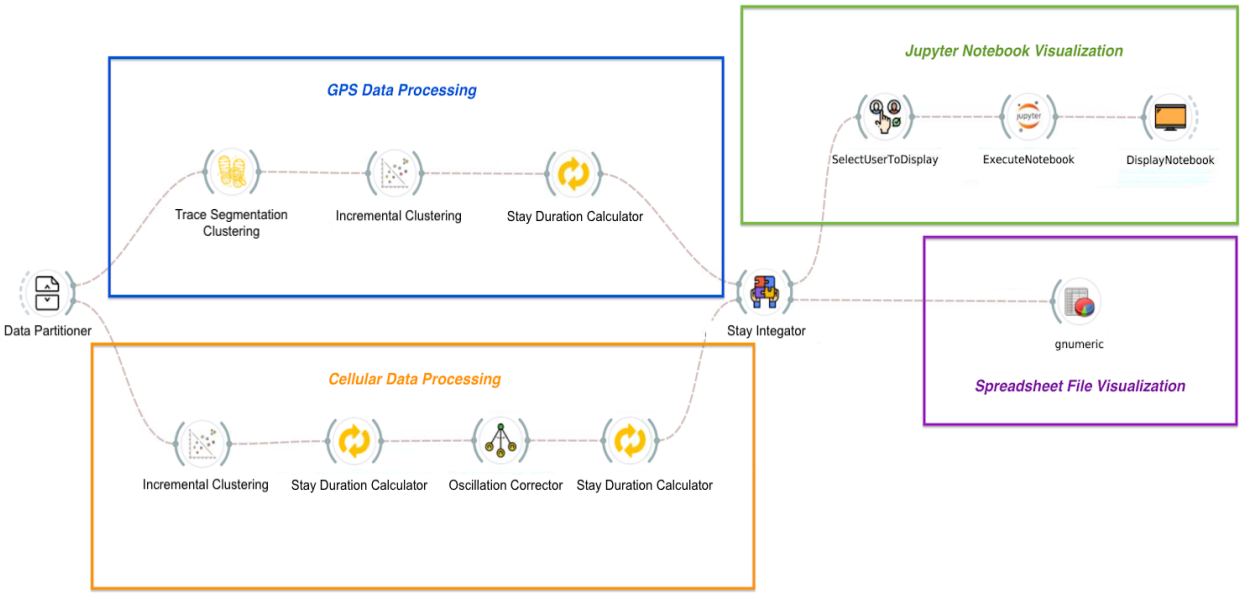

Figure 3.4: The case study mobility workflow in MAW

The case study workflow starts with the "Data Partitioner" widget, which partitions the input mobile sensor data into the data signaled by GPS, known as GPS data, and the data signaled by cellular tower triangulation, known as cellular data. After receiving the input data, the "GPS Data Processing" branch and "Cellular Data Processing" branch will start to work simultaneously to extract users' stays information. The "GPS Data Processing" branch and "Cellular Data Processing" branch have a common goal: extracting the users' stay information given different data types. With the GPS data as input, the "Trace Segmentation Clustering" widget clusters the locations that are spatially close with a large duration, larger than the passing "duration threshold", and calculates the centroids of the clusters as users' stay locations. "Incremental Clustering" widget works similarly to the "Trace Segmentation Clustering" widget but it does not consider the duration threshold. Given a "duration threshold", "Stay Duration Calculator" widget calculates the durations of the users' stays and eliminates the stays with a low staying duration. The "Oscillation Corrector" widget of the "Cellular Data Processing" branch is to correct the oscillation issue of the mobile sensor data. The oscillation issue [15] refers to the case that the mobile device rapidly moved from one place to another in a short time, while in fact, the mobile device is stationary. This could lead to inaccurate computation of the users' stay locations. The "Oscillation Corrector" widget is here to detect the oscillation issue and eliminate the error. After gathering the users' stays extracted from both GPS data and cellular data, the "Stay integrator" widget is to calculate the combined user stay locations by integrating the staying information generated by the "Cellular Data Processing" branch to the staying information calculated by the "GPS Data Processing" branch as the GPS data is more accurate than the cellular data, as well as the stay information extracted from them [15].

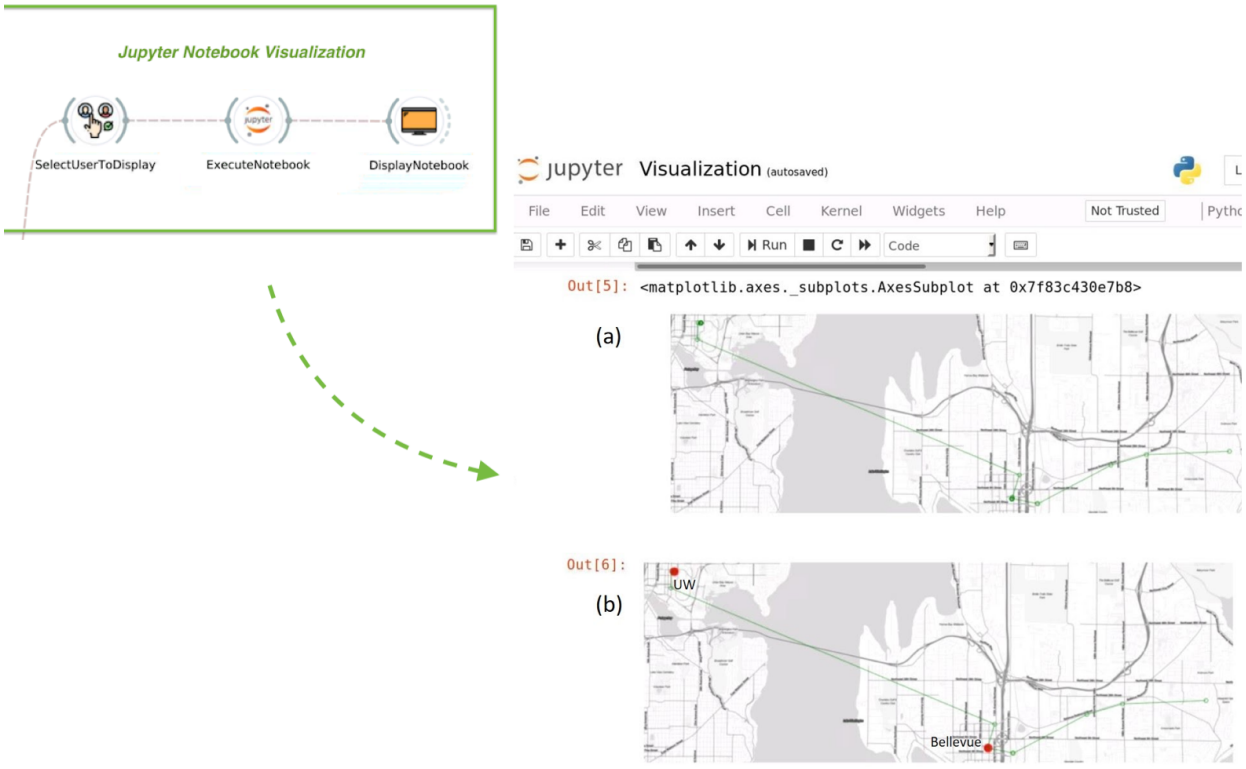

Figure 3.5: Jupyter notebook visualization of the case study mobility workflow in MAW. The green lines on the figure represent the user traces, and the red dots represent the locations the user stayed for a period of time, e.g. 5 minutes.

After generating the stay locations results, the “Jupyter Notebook Visualization” branch, consisting of “Select User To Display” widget, “Execute Notebook” widget and “Display notebook” widget, and the “Spreadsheet File visualization” branch, consisting of “gnumeric” widget will show the visualization results, shown in Figure 3.5. The “Select User To Display” widget passes a selected user ID to the “Execute Notebook” widget, and the “Execute Notebook” widget will execute the pre-configured Jupyter notebook to show the user’s traces and stay locations. After that, the “Display notebook” widget will pop the executed Jupyter notebook up. The “gnumeric” widget will show a spreadsheet of the integrated user stay locations at the same time.

By building our case study workflow, we show that we provide a reproducible and modular workflow building tool, MAW. Through the interactive GUI of the MAW, creation and execution of a workflow are

simplified. Users without any programming experience can perform mobility analysis after the analytical workflow is loaded, which addresses the need of many transportation researchers.

Chapter 4

Optimization of the mobility workflow

In this Chapter, we describe how we conducted systematic benchmarking to inspect the performance of our case study workflow of the mobility analysis method proposed in [52] (referred to as "the mobility workflow" in the later sections) developed in Chapter 3. All the benchmarking experiments were completed on Amazon Web Services (AWS) [10]. Specifically, we used the AWS Elastic Compute Cloud (EC2) for running the experiments and used AWS Simple Storage Service (S3) for storing the experimental datasets. AWS EC2 provides elastic computing resources for users to run their applications. AWS S3 is an object storage service for users to store and manage their files.

The modules "Trace Segmentation Clustering" and "Incremental Clustering" of the mobility workflow involve large amounts of geographical distance calculations using user's location traces, which could become a bottleneck of the performance of the workflow considering the number of location traces. User location trace data are stored in data files with a spreadsheet file format. These files capture the geographical locations of the user traces. Each data file can have up to 20 million rows. As a result, we further investigated the possible computational bottlenecks of the mobility workflow.

4.1 Systematic Benchmarking

4.1.1 Data

The mobile sensor data described in Chapter 3 Section 3.1 was used in our benchmarking experiments covering the four counties (i.e. King, Kitsap, Pierce and Snohomish) of the central Puget Sound region. The data was collected in March, April, and November of 2019, consisting of 2 trillion geographical location records for 562,294 individuals, with a volume of 212 GB. With the help from Dr. Xiangyang Guan, the mobile sensor data is preprocessed to group the traces from the same user into one data file such that each data file contains the traces from a fixed number of users to enable a better understanding of the user's mobility trends. The fixed number of traces per file was set to 5000 in our experiments. Then, we created data subsets using the preprocessed mobile sensor data by selecting seven subsets of data files such that each subset has a data volume of 1 GB, 10 GB, 25 GB, 50 GB, 75 GB and 100 GB respectively. Details are shown in Table 4.1.

Data subset size	1GB	10GB	25GB	50GB	75GB	100GB
Number of data files	2	12	32	69	100	133

Table 4.1: Number of data files of the data subsets

4.1.2 Workflow Execution

Given the data subsets we created, we then executed the mobility workflow in MAW and monitored the runtime and memory utilization of the workflow. Shown as Figure 4.1, starting with the "Data Partitioner" widget, the mobility workflow can be easily executed by clicking the "start" button of the configuration form of the first widget, shown as Figure 4.2.

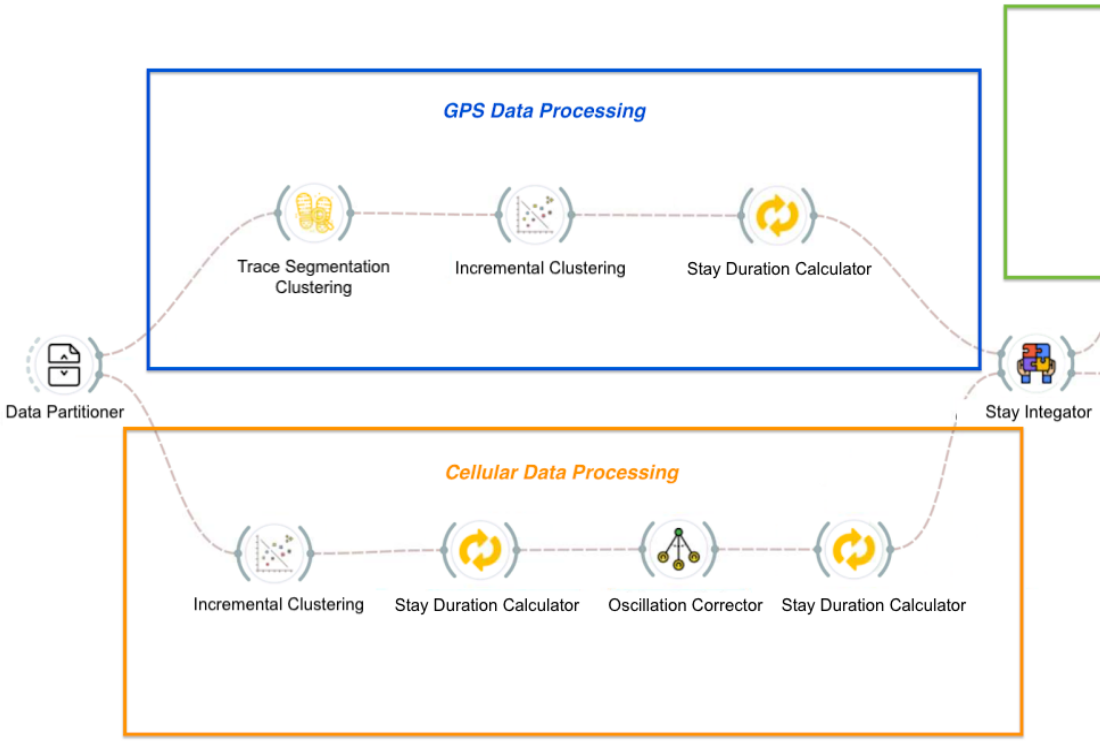

Figure 4.1: The mobility workflow in MAW

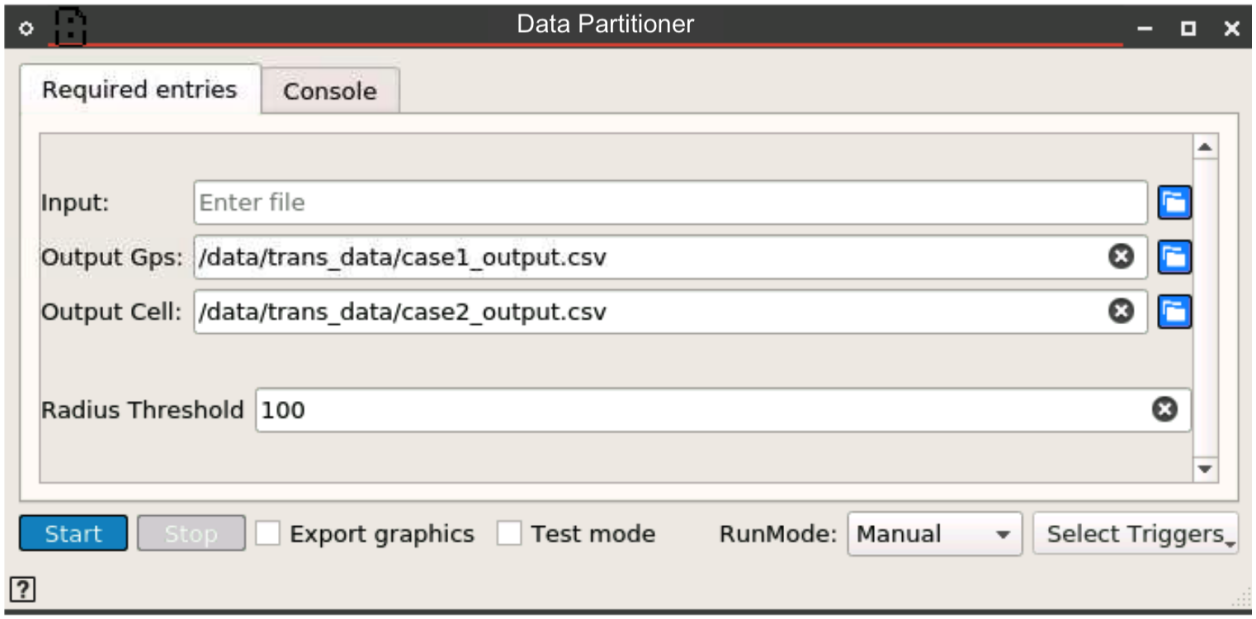

Figure 4.2: The configuration form of "Data Partitioner" widget

For the workflow execution, we need to specify the data file names of one data subset, and any required parameters. The workflow will output a processed data file for each input data file. In the empirical experiments, we executed the workflow multiple times to calculate the average runtime and memory usage of the mobility workflow.

4.2 Bottleneck Identification

4.2.1 Geographical distance measurement

To investigate how different distance measurement methods could influence the runtime of the workflow, we implemented 3 distance estimation functions, shown in Equations 4.1, 4.2, 4.3. To determine the most accurate and computationally efficient distance function, we compared their runtimes and accuracies given the same input data. Eventually, we tested the runtimes of the workflow with the different distance estimation functions.

$$d = \sqrt{(K1 * (long_1 - long_2))^2 + (K2 * (lat_1 - lat_2))^2} \tag{4.1}$$

$$K1 = 111.13209 - 0.56605cos(2\varphi_m) + 0.0012cos(4\varphi_m)$$

$$K2 = 111.41513cos(\varphi_m) - 0.09455cos(3\varphi_m) + 0.00012cos(5\varphi_m)$$

$$\varphi_m = (\varphi_1 + \varphi_2)/2$$

Equation 4.1 is the Ellipsoidal Earth projected to a plane function [61], where $long_1$, lat_1 and $long_2$, lat_2 are two pairs of geographical coordinates in decimal degrees, φ_1 and φ_2 representing the latitudes in radians, and φ_m is the mean value of the latitudes in radians.

$$d = acos\left(sin\varphi_1 \cdot sin\varphi_2 + cos\varphi_1 \cdot cos\varphi_2 \cdot cos(\lambda_1 - \lambda_2)\right) \cdot R \tag{4.2}$$

Equation 4.2 applies the Spherical Law of Cosines function [62] to compute the geographical distance, where φ_1 and φ_2 represent the latitudes in radians, and λ_1 and λ_2 represent longitudes in radians given two pairs of geographical coordinates. *R* is the radius of the earth in kilometers.

$$d = R \cdot \sqrt{((\lambda_1 - \lambda_2) \cdot cos(\varphi_m))^2 + (\varphi_1 - \varphi_2)^2} \tag{4.3}$$

$$\varphi_m = (\varphi_1 + \varphi_2)/2$$

Equation 4.3 provides equirectangular approximation [63] distance functions where φ_1 and φ_2 represent the latitudes in radians, and λ_1 and λ_2 represent the longitudes in radians. Given two pairs of geological coordinates, φ_m is the mean value of the latitudes in radians, and *R* is the radius of the earth in kilometers.

4.3 Results

4.3.1 Runtime and memory utilization of the mobility workflow

We store the data subsets as described in Section 4.1.1 in an AWS S3 bucket. We then monitor the runtime and memory utilization of running the mobility workflow on the AWS EC2 instances. Specifically, we implement the runtime monitoring shell script and the memory utilization monitoring shell script and launch the scripts at the beginning of the workflow execution. The scripts measure the workflow's runtime and output the memory utilization every 5 seconds during the workflow execution. In this way, by collecting runtime and memory utilization, we can understand the dynamic changes of memory consumption throughout the execution of the workflow.

The mobility workflow executes each computing task sequentially. During the execution of "Trace Segmentation Clustering", different data files of the same data subset, as described in Section 4.1.1, are processed sequentially. The mobility workflow will not execute the next computing task "Incremental Clustering", until all the data files of one data subset are processed and completed. To speed up the processing of each data file, a multi-processing mechanism depending on the Python library "multiprocessing" is implemented. Specifically, our data processing scripts divide a data file into 16 shards, as the EC2 instances used in the experiment have 16 vCPU cores, and launch 16 sub-processes where each of them is assigned one data shard to run the function that implements "Trace Segmentation Clustering" in Python. Note that the number of data shards and the number of sub-processes are set to be equal to the number of vCPU cores in the implementation. When we change to a new AWS EC2 instance with a different number of vCPU cores to run the mobility workflow, the number of data shards and the number of sub-processes will automatically be adjusted.

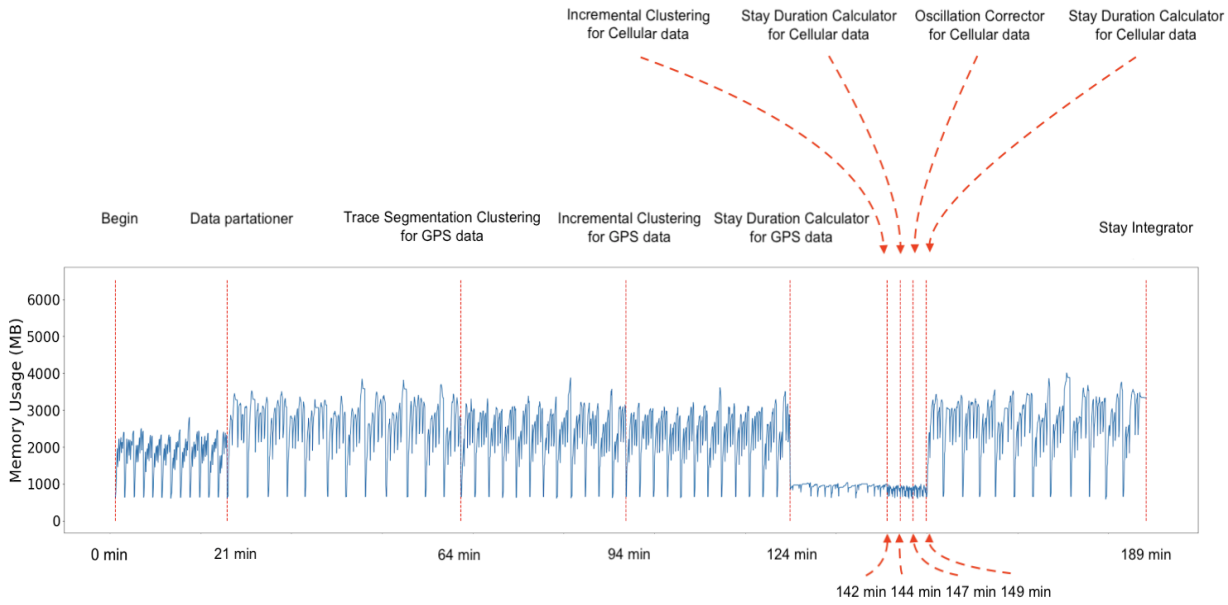

Figure 4.3: Runtime and memory utilization of the mobility workflow with a 10 GB input dataset

Figure 4.3 shows the monitored runtime and memory utilization when the workflow is executed on an "m5d.4xlarge" EC2 instance. The x-axis of the figure represents the runtime in minutes starting at 0 minute, and the y-axis shows the memory usage in the process of workflow execution. The red dotted lines indicate the time points when a specific widget is finished, and the extended red dotted lines are utilized for better annotation. For example, we can see from the figure that the "Data Partitioner" widget is finished after 21 minutes of workflow execution. The time interval of the lowest memory utilization appeared when the workflow was processing cellular data, while the highest memory utilization time

period is when the “Stay Integrator” widget is running, from the 149 to 189 minutes. From Figure 4.3, we can see that the peak memory utilization is about 4000 MB, while the minimum memory utilization is about 1000 MB. The “Trace Segmentation Clustering” task consumes the largest portion of computational time among all the tasks.

Taking different data subsets as input, the workflow is executed on the AWS EC2 instances (type: “m5d.4xlarge”). The time costs of finishing the workflow are compared as shown in Table 4.2. We repeated each experimental configuration 3 times to determine the average runtime to account for the system variability of EC2 instances, e.g. we repeated the workflow 3 x using the 1 GB data subset; we reused the data subsets without recreating them. From the table, we can see that as the data subset size increases, the time cost of finishing the workflow grows significantly. Through our empirical experiments, the m5d.4xlarge EC2 instance provides sufficient memory for each set of experiments and, as a result, we use the m5d.4xlarge EC2 instance as our default instance.

Data subset size	1GB	10GB	25GB	50GB	75GB	100GB
Average Time cost (H:M:S)	0:16:28	3:09:18	7:42:20	15:17:40	23:21:10	30:44:59

Table 4.2: Average time cost of running the workflow with different data subsets

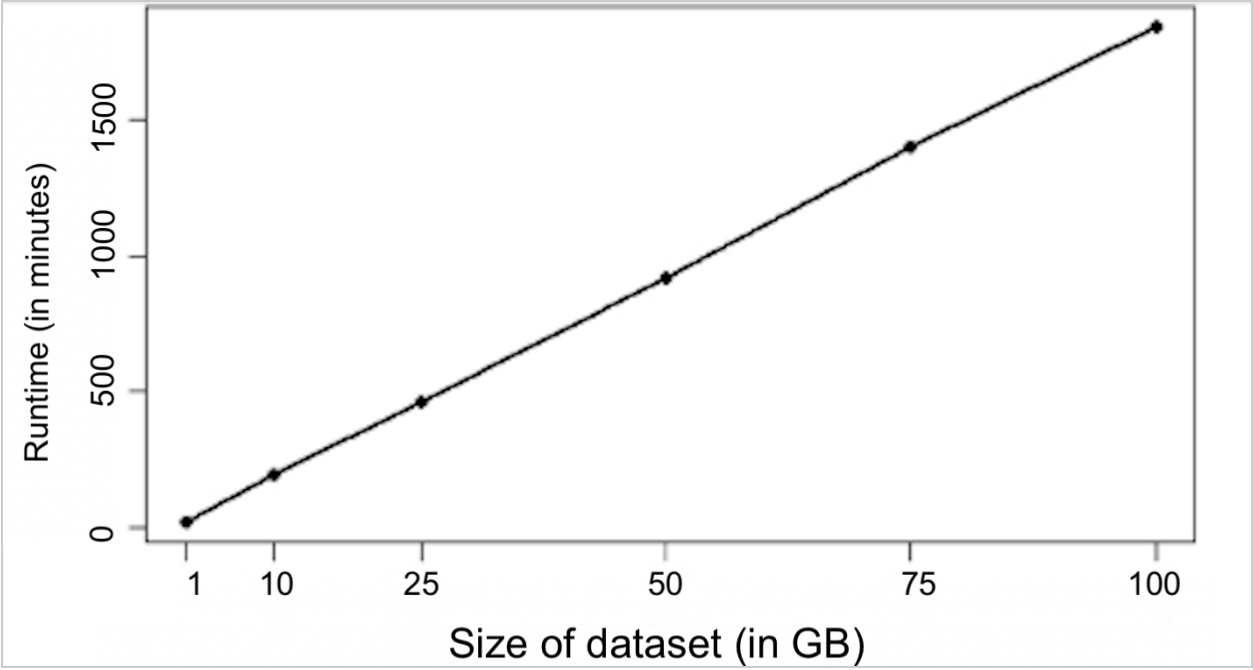

Figure 4.4: Average time cost of running the mobility workflow with data subsets of different sizes (No recreation of data subset)

As shown in Figure 4.4, the x-axis represents the sizes of the datasets, and the y-axis represents the runtime (in minutes) of the mobility workflow. When the size of the dataset increases, the average time cost of completing the workflow, and the size of the dataset show positive correlation. We also observe that, if we compare the runtime of processing 1 GB data and the runtime of processing 10 GB data, while the 10 GB data subset is 10 times of 1 GB data subset in terms of size, the runtime is not necessarily 10 times larger.

4.3.2 Performance comparison of distance estimation methods

To compare the performance of different distance estimation functions, regarding the runtime and accuracy, we performed another set of empirical experiments. First, we randomly sampled geographical coordinates from the Puget Sound region as experimental data. Then for each pair of coordinates, we computed the "pseudo ground truth" distance using the "geodesic" distance function. The "geodesic" distance function was computed using the "geopy" Python library and we regarded the distance measurement result from the "geodesic" distance function as the "pseudo ground truth" in our experiments. We also calculated the estimated distances using the three self-implemented distance functions to compare the accuracy and runtime.

Runtime comparison

To compare the runtimes of the 3 distance estimation methods, we randomly sampled 1000 geographical coordinates from the Puget Sound region as input for each experiment and calculated the average time cost of computing distances for all coordinate pairs. Then, we repeated the experiments 15 times to avoid any sampling bias.

Distance Function	Ellipsoidal Earth projected to a plane	Spherical Law of Cosines	Equirectangular approximation
Average runtime (*in microseconds*)	2.792	2.576	2.153

Table 4.3: Average runtimes of (in microseconds) distance functions for all coordinate pairs

As shown in Table 4.3, the Equirectangular approximation distance function gives better performance in terms of the average runtime over the pairwise distance calculations of 1000 geographical coordinates. We also provide a visualization of the comparison using a bar chart shown in Figure 4.5.

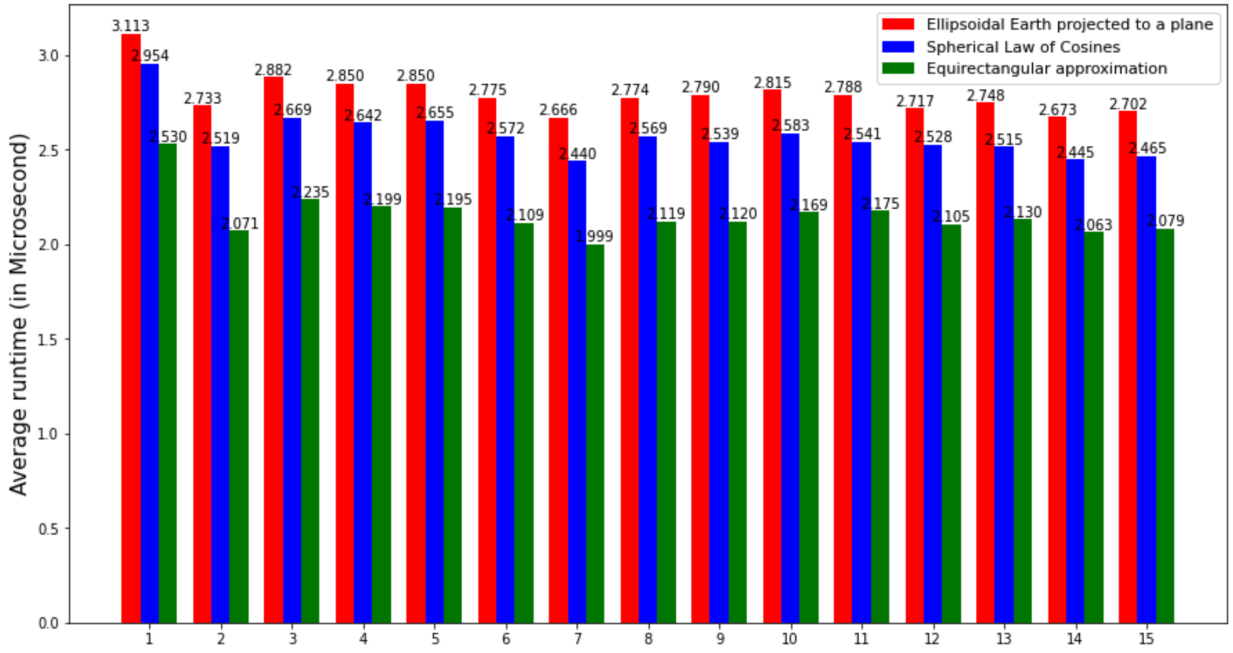

Figure 4.5: Average runtime (in microseconds) of distance functions in different experiments

Figure 4.5 shows the average runtime of distance functions in different experiments over a total of 15 repeated experiments. The x-axis of Figure 4.5 represents the different experiments. For example, "1"

represents the first experiment we conducted. The y-axis shows the average runtime in microseconds of each distance function. The average runtime of the Ellipsoidal Earth projected to a plane function, the Spherical Law of Cosines function, and the Equirectangular approximation function are shown using a red bar, blue bar and green bar, respectively. From Figure 4.5, we can observe that the Equirectangular approximation functions outperformed the other two distance functions given the different input data of each experiment.

Furthermore, we created box plots with standard deviation error bars to investigate the variability of the runtimes in the process.

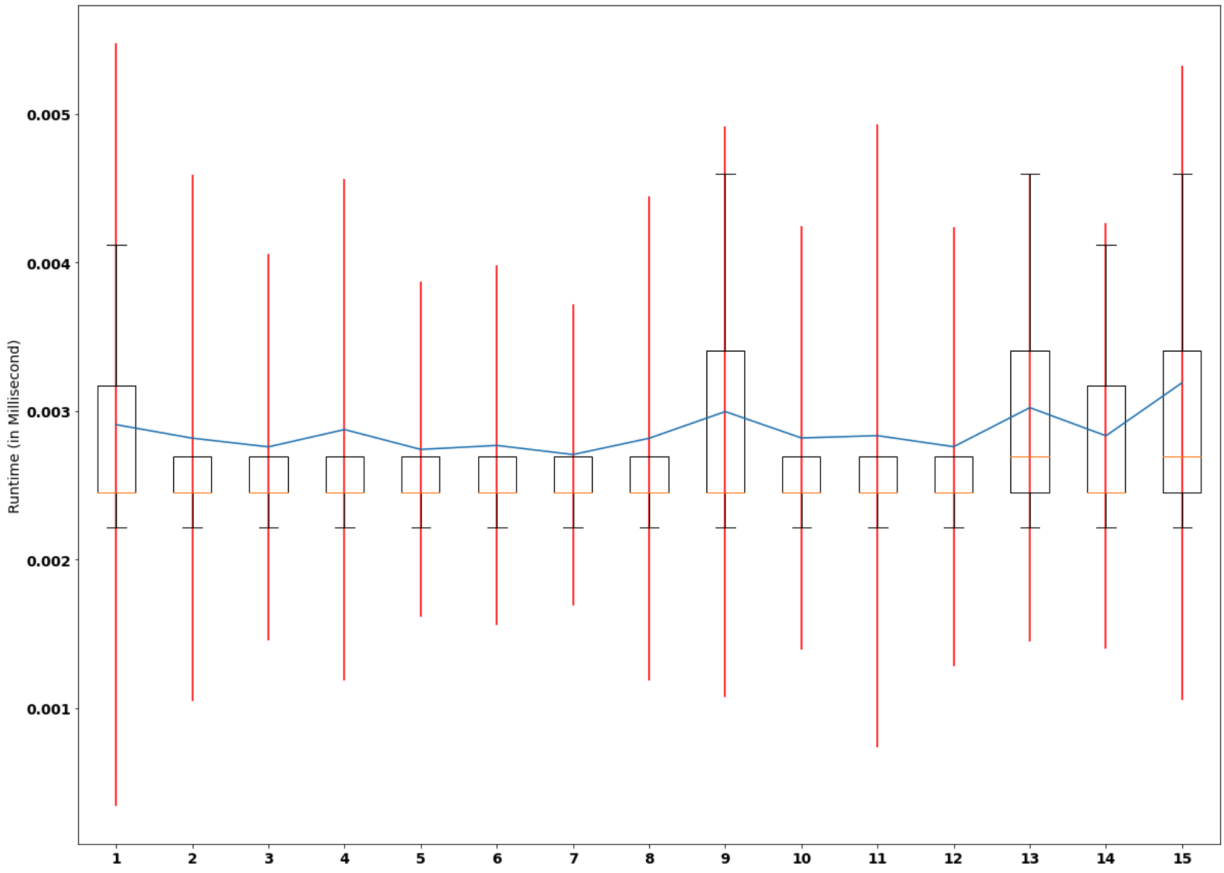

Figure 4.6: Box plot with standard deviation error bars using the runtimes of the Ellipsoidal Earth projected to a plane distance function

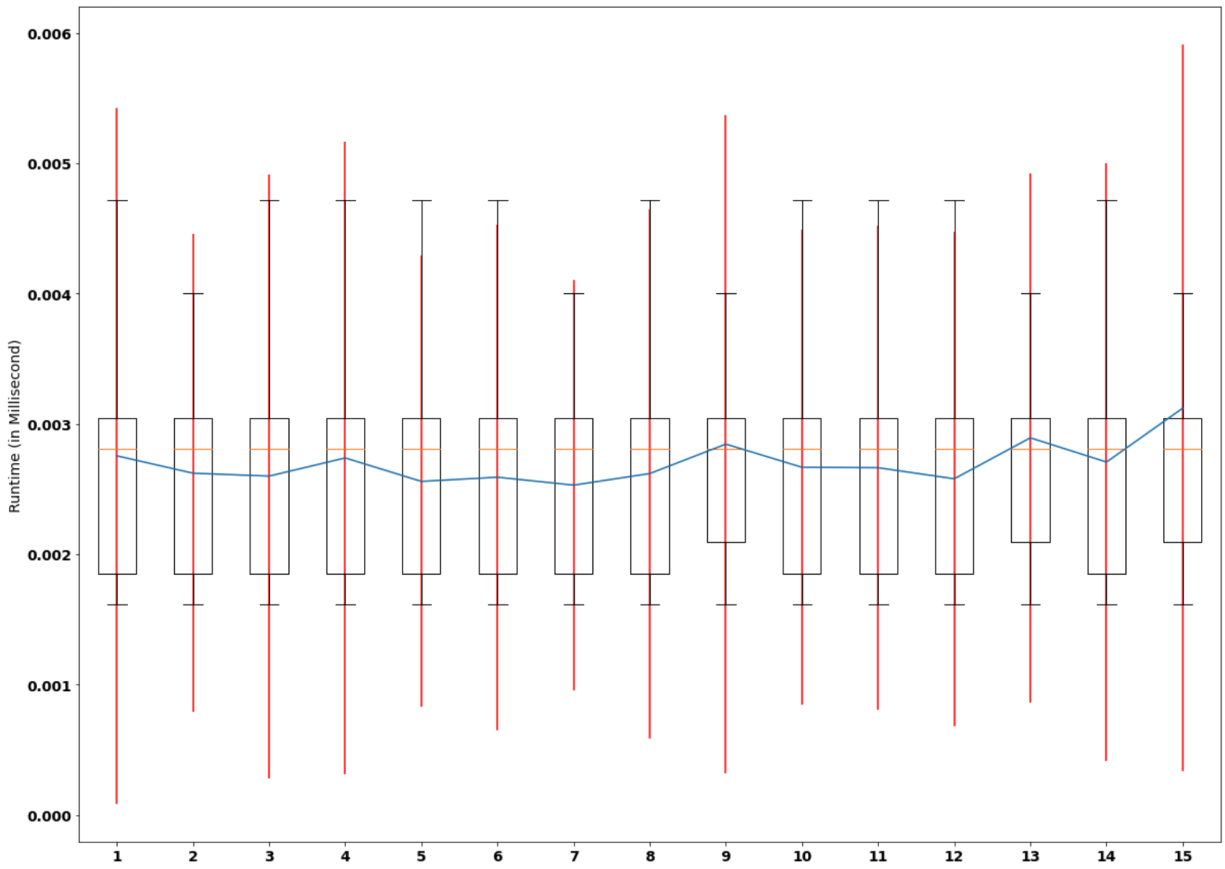

Figure 4.7: Box plot with standard deviation error bars using the runtimes of the Spherical Law of Cosines distance function

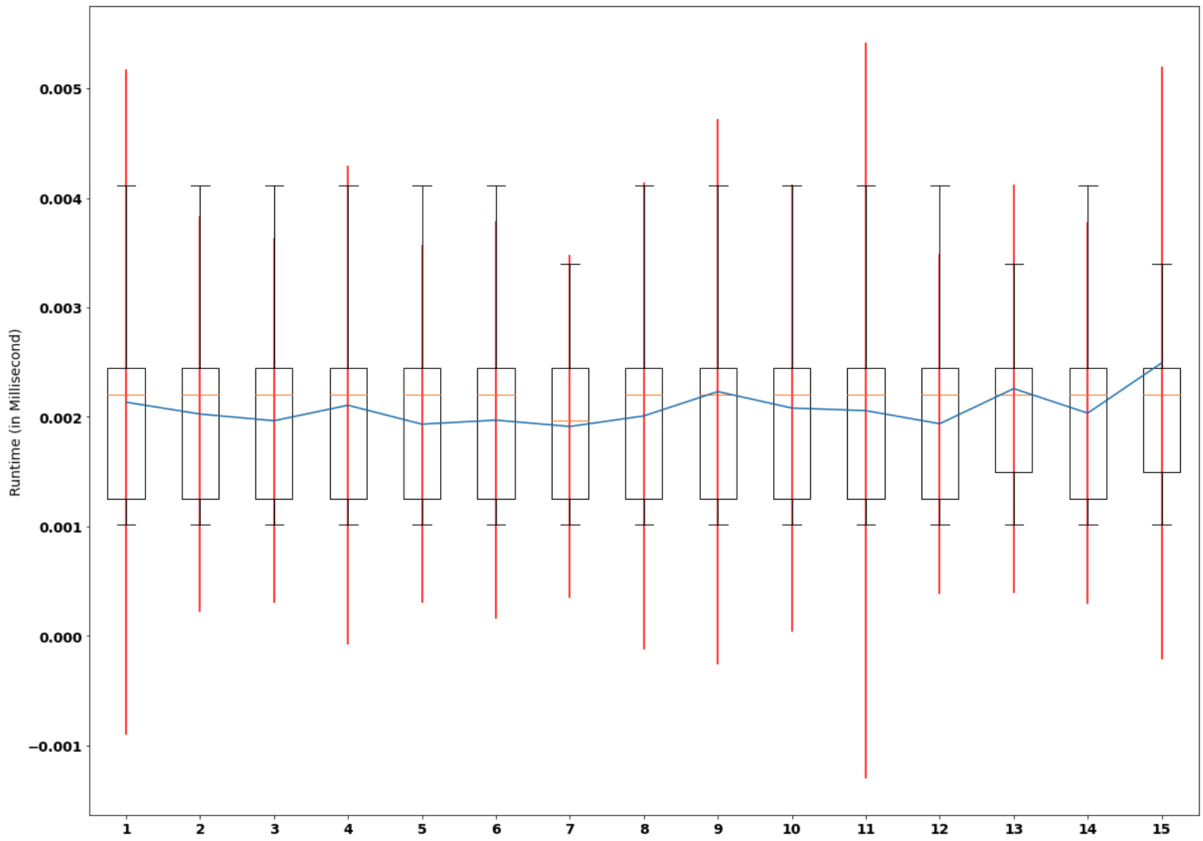

Figure 4.8: Box plot with standard deviation error bars using the runtimes of the equirectangular approximation distance function

For Figure 4.6, Figure 4.7, and Figure 4.8, the x-axis indicates the total 15 experiments, in which "1" of the x-axis represents the first experiment, and the y-axis shows the runtime (in milliseconds). The error bars, shown as red bars with blue lines, reflects the standard deviation of the runtimes. By comparing convergence of the median values, shown in the box plot, and the average values, shown in with the standard deviation error bar, we can observe that the variance of runtime of the Ellipsoidal Earth projected to a plane distance function is larger than the variances of runtime of the other two distance functions. In addition, by comparing the interquartile ranges of the box plot given the same experiment, we observe that the distribution of the runtimes of the Ellipsoidal Earth projected to a plane distance function shows relatively little variation compared to those from the other two distance functions.

Accuracy comparison

We randomly sampled 1000 geographical coordinates from the Puget Sound region as input for each experiment, and then computed and compared the error in distance measurements for the 3 distance estimation methods. We used the root-mean-square deviation (RMSD) to measure the error between the "pseudo ground truth" distance, generated by the "geodesic" distance function, and the estimated

distance, computed by the 3 distance functions. We repeated the experiments 3 times to reduce any sampling bias.

$$RMSD = \sqrt{\frac{\sum_{i=1}^{N}(d_i - \hat{d}_i)^2}{N}} \tag{4.4}$$

The root-mean-square deviation (RMSD) is a measurement which reflects the average error of estimation results over the true results. In our experiment, the estimation results are the estimated distance, and the true results are "pseudo ground truth" distance. In Equation 4.4, *i*represents the identifier of the coordinate pair; *N*is the total number of the coordinate pairs; $d_i$is the "pseudo ground truth" distance of the i th coordinate pair generated by the "geodesic" distance function; and $\widehat{d_i}$ is the estimated distance of the i th coordinate pair generated by the 3 self-implemented distance functions.

Metrics (in Meters)	Distance Functions		
	Ellipsoidal Earth projected to a plane	Spherical Law of Cosines	Equirectangular approximation
(1st) RMSD	435.2	77.7	75.0
(1st) RMSD / MEAN	0.0046	0.0008	0.0008
(2nd) RMSD	364.8	75.0	74.4
(2nd) RMSD / MEAN	0.0041	0.0008	0.0008
(3rd) RMSD	453.4	86.2	85.0
(3rd) RMSD / MEAN	0.0050	0.0009	0.0009

Table 4.4: Time cost and RMSD comparison using different distance functions

The "MEAN" in Table 4.4, is the mean value of the "pseudo ground truth" distances across all pairs of coordinates measured by "geodesic" function, and "RMSD/MEAN" is calculated using the RMSD results divided by the mean value of the "pseudo ground truth" distances, which reflects the ratio of the error of estimation to the mean distance. "(1st)", "(2nd)" and "(3rd)" under the "Metrics (in Meters)" column

indicate the first experiment, the second experiment, and the third experiment respectively, and "(1st) RMSD" represents the RMSD result generated from the the first experiment. From Table 4.4, we can see that Equirectangular approximation distance function shows relatively better performance given our sampled coordinates data.

Next, using our 3 different distance functions, we compared the runtimes of the mobility workflow with data subsets as described in Section 4.1.1 as input. We visualized the runtime results using a bar chart.

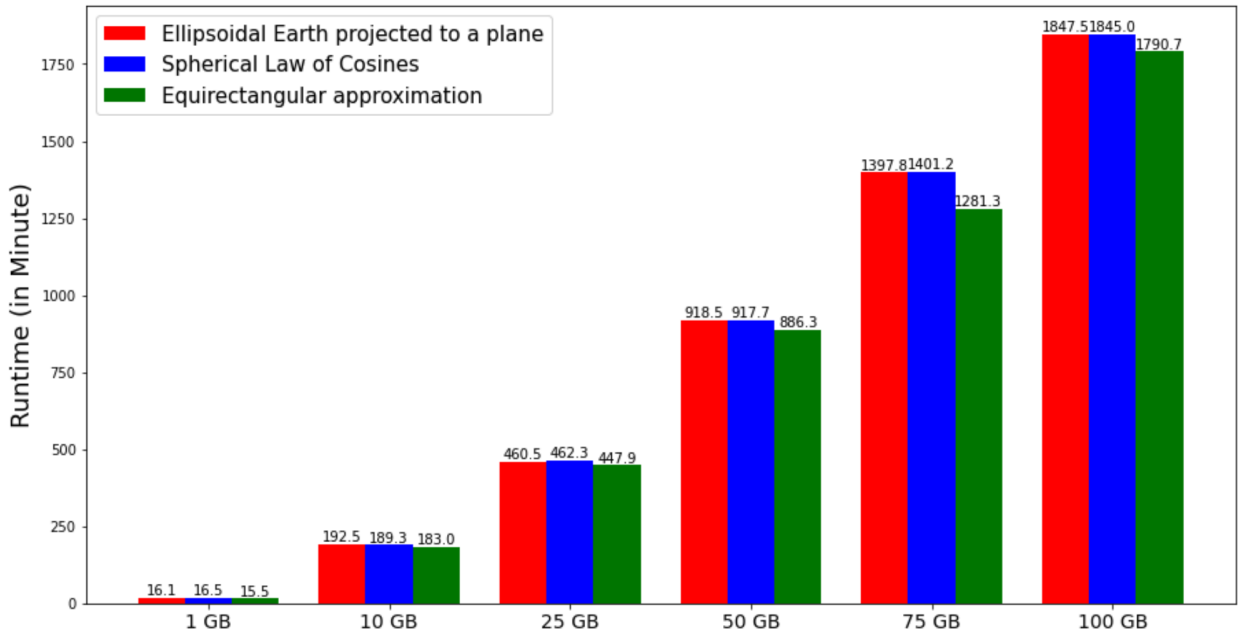

Figure 4.9: Bar chart of the runtime (in minute) of the mobility workflow with different distance functions

The x-axis in Figure 4.9 indicates the sizes of different datasets, and the y-axis represents the running time (in minutes) to complete the workflow. From Figure 4.9, we can see that given the same input dataset, the mobility workflow with the equirectangular approximation distance function takes the lowest runtime to finish compared with the workflow with the other two distance functions.

Data Processing Capacity Comparison

To measure the exact data processing capacity of the mobility workflow with different distance functions, we calculated the throughput of the mobility workflow. Throughput is a metric to represent the amount of data that can be processed in a unit of time, which can be used to describe the capacity of data processing of a system or an algorithm [65].

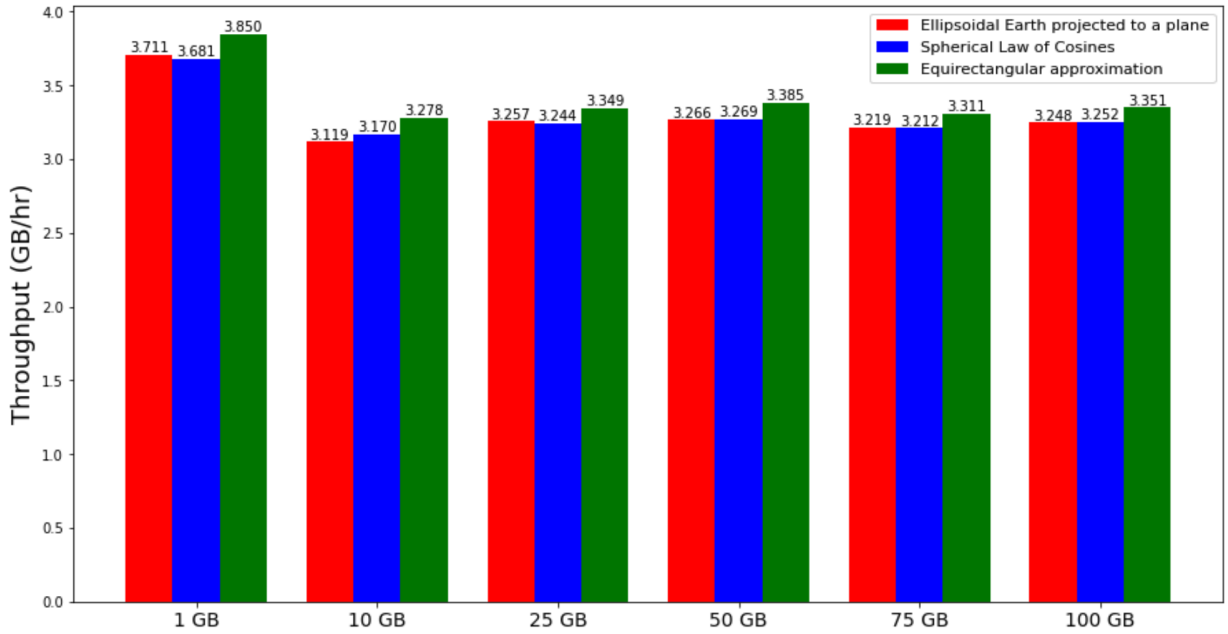

Figure 4.10: Bar plot of the throughput (in GB/hr) of the mobility workflow with different distance functions

As shown in Figure 4.10, the x-axis indicates the sizes of different datasets, and the y-axis represents the throughput (in GB/hr) to complete the workflow. We can observe that given the same input dataset, the mobility workflow with the equirectangular approximation distance function has the highest throughput compared with the workflow with the other two distance functions.

4.4 Conclusions

From our empirical experiments, we showed the performance of our case study workflow from different angles and identified performance improvement opportunities. To be specific, we discovered that by implementing the equirectangular approximation distance estimation function, we optimized our case study workflow in terms of runtime and data processing capacity. Additionally, by running the workflows on the AWS EC2 instances, we demonstrated that MAW can be deployed on the cloud to leverage on-demand and scalable computational resources.

Chapter 5

Reproducibility in Modeling the spread of COVID-19

5.1 Rationale

Compartmental models described in Chapter 2 are widely used in epidemiology [43] [44] [45]. These models have provided helpful information to the public during the current COVID-19 pandemic. For example, compartmental models are used to predict the trend of the epidemic, such as the total number of infected people over time. However, given the required programming knowledge to deploy and run compartmental models, it is non-trivial for epidemiology researchers to reproduce analytical results. Considering the compartmental models can be implemented differently with different parameters, it can be even more difficult to reproduce the compartmental models analysis results.

As another application for MAW, we implemented the Susceptible, Exposed, Infected, Recovered (SEIR) [48] model workflow as an example of modeling the spread of COVID-19 using the compartmental models.

5.2 Data

We utilize three sources of data as input to our SEIR model workflow. The first one is the airport flights statistics data collected from the U.S. Bureau of Transportation Statistics [29]. It provides the daily numbers of flights from one city to another shown in Table 5.1.

Date	New York	Los Angeles	Chicago	Dallas	Houston	Washington	Philadelphia	Miami	Atlanta	Boston	San Francisco	Phoenix	Detroit	Seattle
21-Jan	12	34	13	11	5	6	2	1	9	5	25	17	0	-
22-Jan	13	35	12	12	5	6	2	1	9	7	25	18	0	-
23-Jan	13	37	13	14	5	6	3	1	8	6	27	19	0	-
24-Jan	13	37	13	14	5	6	3	1	8	7	28	19	0	-
25-Jan	10	24	9	8	3	5	2	1	5	3	20	15	0	-
26-Jan	12	28	10	14	4	6	2	1	8	9	23	18	0	-
27-Jan	13	37	14	13	5	6	3	1	9	6	27	19	0	-
28-Jan	13	34	13	11	5	6	2	1	9	5	25	17	0	-
29-Jan	13	35	12	12	5	6	2	1	9	7	25	18	0	-
30-Jan	13	37	14	14	5	6	3	1	9	7	26	19	0	-
31-Jan	13	37	14	14	5	6	3	1	9	5	28	19	0	-
1-Feb	10	23	9	8	3	5	1	1	5	3	20	15	0	-
2-Feb	12	26	10	14	4	5	2	1	8	8	22	17	0	-
3-Feb	12	36	14	13	5	6	3	1	9	6	28	19	0	-
4-Feb	13	32	13	11	5	6	2	1	9	5	25	17	0	-
5-Feb	13	35	12	12	5	6	2	1	9	7	25	17	0	-
6-Feb	13	36	14	14	5	6	3	1	9	7	27	19	0	-
7-Feb	13	37	14	14	5	6	3	1	9	7	28	19	0	-
8-Feb	10	24	9	8	3	5	2	1	5	3	20	15	0	-

Table 5.1: Airport flights statistics data departing from Seattle [29]

Table 5.1 shows an example of the airport flights statistics data departing from Seattle. The column "Date" is the date when the airport flights departed from Seattle, and the other columns with a city name

as header show the number of flights flying from Seattle to that city. Each row of the table represents the number of flights flying from Seattle to the other cities, such as New York City, Los Angeles, on a specific date. We note that all the airport flight data provided by the U.S. Bureau of Transportation Statistics only contains airport-wise information, such as the number of flights from one airport to the other. To gather flight data at city level, we collected all the airport-wise data for airports that are located in one city and did the summations.

The second data source is city-wise population data in the U.S, collected by the U.S. Census Bureau [66].

	City	Population
2	Chicago	9458539
3	Dallas Fort Worth	7573136
4	Houston	7066141
5	Atlanta	6020364
6	Boston	4873019
7	San Francisco	4731803
8	Phoenix	4948203
9	Seattle	3979845
10	San Diego	3338330
11	Minneapolis Saint Paul	3640043
12	Los Angeles	13214799
13	Denver	2967239
14	Tampa Bay area	3194831
15	New York	19216182

Table 5.2: City-wise population data from [66]

We focus on the population data of the 14 U.S. cities shown in Table 5.2 in our study. The population data shown are collected from the 2020 U.S. population census. Each row of the table, except for the header, represents the estimated population of one city as of April 1, 2020. For example, the estimated population in Chicago is 9,458,539 as of April 1, 2020.

The third data source of our SEIR model workflow is the daily confirmed cases of COVID-19 data from New York Times [56] shown in Figure 5.1.

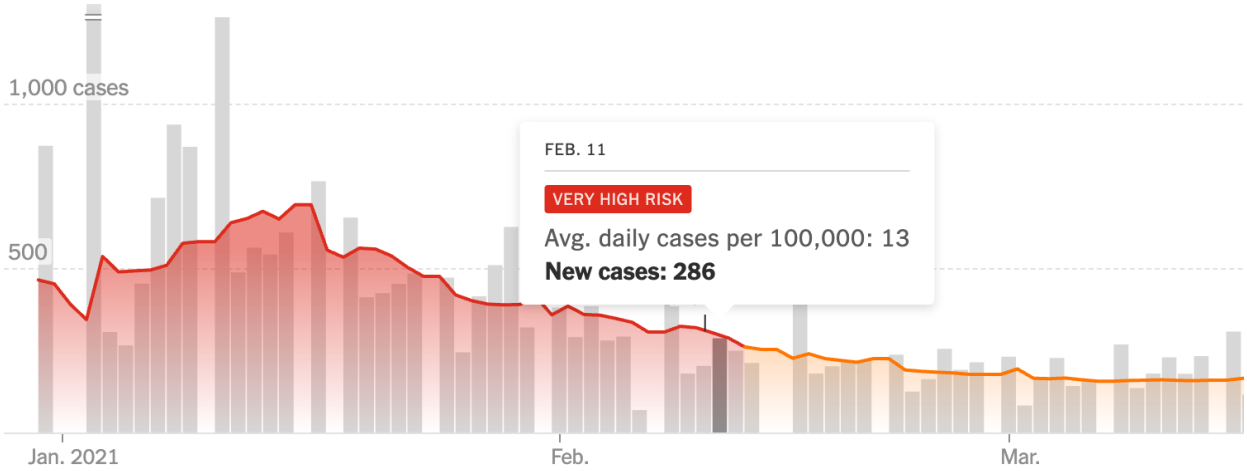

Figure 5.1: Daily confirmed cases of COVID-19 data from New York Times [56] in King county, WA

The x-axis of Figure 5.1 represents the date, and the y-axis shows the number of daily confirmed cases of COVID-19 in King county. We can observe that the number of new cases on February 11 in King county is 286 from Figure 5.1, and the estimated risk level is "very high risk" given the trend of the confirmed cases in one county or state. In our study, we utilize the daily confirmed cases data to compute the initial value of the Susceptible, Exposed, Infected, and Recovered compartments using the SEIR model.

5.3 Method

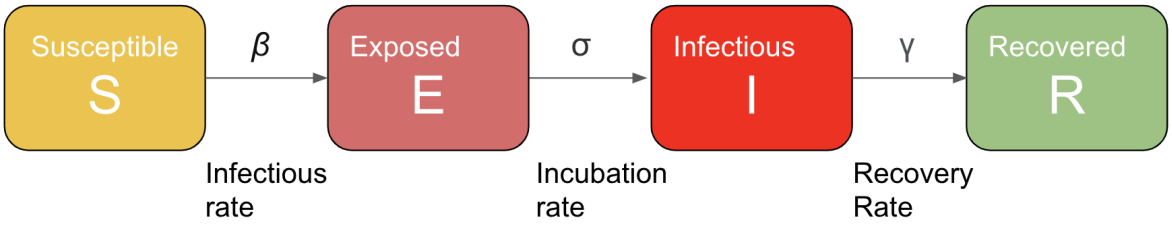

Figure 5.2: SEIR model illustration

Shown in Figure 5.2, the SEIR [48] model, as a compartmental model, is proposed to describe the dynamic changes of four different compartments: the "Susceptible" compartment, the "Exposed" compartment, the "Infectious" compartment, and the "Recovered" compartment, during the spread of COVID-19. Different from the SIR model introduced in Chapter 2, the SEIR model assigns the people who are infected but not yet infectious to the "Exposed" compartment.

$$
\begin{aligned}
S(t+1) - S(t) &= -\beta \cdot I(t) \cdot \frac{S(t)}{N} \\
E(t+1) - E(t) &= \beta \cdot I(t) \cdot \frac{S(t)}{N} - \frac{E(t)}{\sigma} \\
I(t+1) - I(t) &= \frac{E(t)}{\sigma} - \gamma \cdot I(t) \\
R(t+1) - R(t) &= \gamma \cdot I(t)
\end{aligned}
\qquad (5.1)
$$

Equation 5.1 represents the mathematical formula behind the SEIR model. Specifically, as shown in Equation 5.1, $S(t)$, $E(t)$, $I(t)$ and $R(t)$ are the numbers of persons in the "Susceptible", "Exposed", "Infected" and "Recovered" compartments at time step t, respectively. "$t+1$" represents the next time step of the time step t. Specifically, we use "day" as our time unit for each time step. $N$is the total population of a geographical region, such as a city. There are three parameters in the model, the infection rate β, incubation rate σ, and recovery rate γ. The infection rate β represents the percentage at which susceptible individuals transfer from the susceptible compartment to the exposed compartment. The incubation rate σ is the time interval from a person being infected to becoming infectious. The recovery rate γ is the daily proportion of infectious people who recover or die.

In our empirical experiment, we use the estimated values for β,σ and γ, which are collected from [67] [68] [69] [70] [71], as those estimated values are scientifically tested to generate accurate COVID-19 trend estimates, and they gained community recognition in epidemiology. Accordingly, the number of persons in the four compartments can be calculated given the daily confirmed cases data. Then, the calculated numbers of persons in the four compartments, known as the states of the four compartments, are fed to the SEIR model to forecast the states in the next time step.

The SEIR model workflow was built with three steps: modularization, creation of containers, and workflow development. These three steps are outlined in the following subsections.

5.3.1 Modularization of SEIR Model

Following the same modularization method described in Section 3.3.2, we modularized the SEIR model code into multiple software modules, such that each of them could be a set of programs which can run independently from other modules.

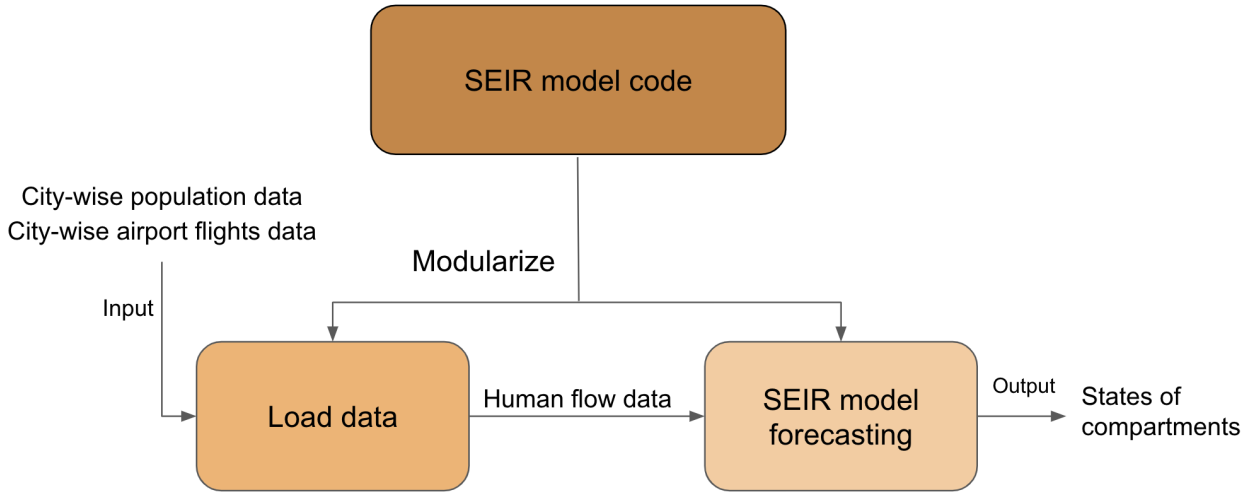

Figure 5.3: Modularization of SEIR model

We illustrate the modularization of the SEIR model in Figure 5.3. Based on the functionality, we modularized the SEIR model into 2 modules, "Load data" and "SEIR model forecasting" modules. The "Load data" module reads in the city-wise population data and city-wise airport flights data to generate the human flow data between different cities by running the existing data preprocessing scripts. Specifically, the human flow data describes the number of people "flowing" into one city from the others on a specific date, and we can also know the number of people moving out of one city given a date. The "SEIR model forecasting" module takes the human flow data, the city-wise population data, and the calculated states of the four compartments of each city studied, as mentioned above, to conduct forecasting using the SEIR model with the pre-configured parameters, β, σ, and γ. Given the states of the four compartments, the "SEIR model forecasting" module only runs the SEIR model for one iteration to forecast the new states in the next time step, i.e. the next day. Considering the dynamic human flow among multiple cities, we recalculate the states of the four compartments after performing forecasting using the SEIR model by directly adding the number of inflow people and reducing the number of outflow people to each compartment.

5.3.2 Creation of Containers

After the software modules are designed and created, we create Docker containers to provide an isolated running environment to encapsulate the required dependencies. Specifically, the "Load data" and "SEIR model forecasting" modules are implemented using R. As a result, we create Docker images to encapsulate the R running environment and the running scripts of the "Load data" and "SEIR model forecasting" modules. To give an example, the Docker container running the "Load data" job reads in

city-wise population data and city-wise airport flights data and outputs the human flow data. Given the human flow data, the created Docker container for running the "SEIR model forecasting" job outputs the SEIR model forecasting result as the states of the four compartments.

5.3.3 Workflow Development

Lastly, we used the "New Widget" functionality in MAW to add new widgets that incorporate the created Docker images and Docker containers. We connected these widgets sequentially to build the workflow. Note that each "SEIR model forecasting" widget only conducts the forecasting jobs for one time step, i.e. one day. To conduct the forecasting jobs for multiple time steps, we need to duplicate multiple "SEIR model forecasting" widgets with the interconnection to stimulate a multiple steps forecasting procedure.

5.4 Results

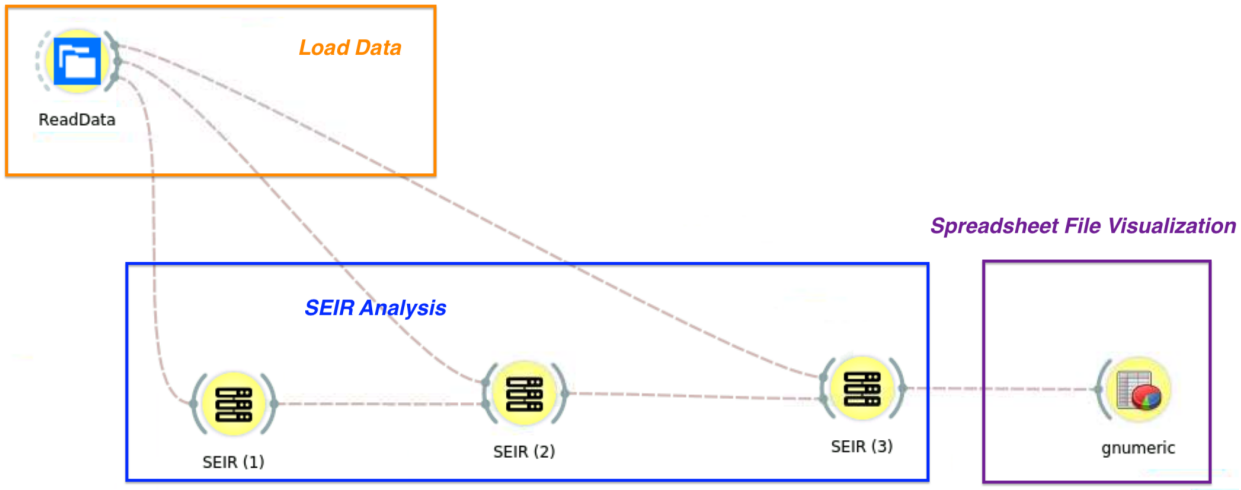

Figure 5.4: SEIR workflow

Shown in Figure 5.4, the SEIR model workflow consists of the "Load Data" branch, the "SEIR Analysis" branch, and the "Spreadsheet File Visualization" branch. The "ReadData" widget of the "Load Data" branch performs the functionality of the "Load Data" module: reading population data and airport flights data and outputting the human flow data to the "SEIR Analysis" branch. The "SEIR (1)", "SEIR (2)" and "SEIR (3)" widgets of the "SEIR Analysis" branch takes the human flow data, the population data of multiple cities, and the calculated states of the four compartments of multiple cities and forecasts the new states of the four compartments using the SEIR model. Specifically, we create and connect multiple widgets, "SEIR (1)", "SEIR (2)" and "SEIR (3)" widgets, to stimulate a multiple steps forecasting procedure with each one of them representing one time step. Finally, the "gnumeric" widget of the

"Spreadsheet File Visualization" branch provides a spreadsheet visualization showing the SEIR analysis result in Table 5.3.

1		St	Et	It	Rt
2	Chicago	9413422	11174	4594	664
3	Dallas	7609251	3033	1223	174
4	Houston	7045137	832	482	108
5	Atlanta	5973476	1962	1339	306
6	Boston	4856815	502	974	461
7	San Francisco	4752147	1317	1118	368
8	Phoenix	4977942	660	327	68
9	Seattle	3986273	3975	3671	1563
10	San Diego	3330839	2165	936	147
11	Minneapolis	3631826	411	324	88
12	Los Angeles	13231867	2083	1491	381
13	Denver	2975787	717	677	199
14	Tampa	3099578	1062	474	76
15	New York	18983797	213629	78805	9735

Table 5.3: SEIR analysis result in spreadsheet

Table 5.3 shows the SEIR analysis results of executing the "SEIR" widget for 3 time steps. The first column shows the city names for each row, and the other columns with header "St", "Et", "It" and "Rt" represent the states of the four compartments of different cities. For example, in the second row of Table 5.3, the forecasted numbers of people in the "Susceptible" compartment, the "Exposed" compartment, the "Infected" compartment, and the "recovered" compartment are 9413422, 11174, 4594 and 664 respectively.

Compartmental model workflows in MAW, to the best of our knowledge, is a novel application of reproducibility research in epidemiology. Using our designed workflows, users will be able to reproducibly execute these workflows to generate forecasts of the numbers of people in different disease status with a user-friendly interface that supports interactive graphical output. We developed containers and widgets of the compartmental model workflow. The containers will be publicly available to the transportation and public health research communities from Docker Hub.

Chapter 6

Discussion and Conclusion

6.1 Discussion and Conclusion

In this thesis project, by implementing the case study mobility workflow and the SEIR model workflow, introduced in the Chapter 3 and Chapter 5 respectively, we show that the Mobility Analysis Workflow (MAW) is a reproducibility research tool that can be applied to mobility analysis problems as well as modeling infectious diseases. Thus, MAW can be utilized as a common framework for different applications. The workflows can be built using the same procedure: modularization, creation of containers, and workflow development.

Given the same research data as input, MAW can be used to reproduce published results. In addition, each workflow consists of multiple modules that could be reused in different workflows and applications. For example, the "Incremental Clustering" module of the mobility workflow could be applied to the other mobility analysis cases. Beyond that, MAW provides an interactive graphical user interface (GUI) to the researchers to visualize the experimental process and interactive analyses with a form-based user interface to specify parameters and supporting files.

By deploying MAW on the cloud, large transportation datasets can be efficiently processed by harnessing cloud-based computational resources. Also, as a Docker application, the MAW can be easily deployed on any virtual servers, such as an AWS EC2 instance. With cloud-based storage, large datasets can be stored and shared. Most importantly, researchers can directly upload new data to the cloud and run MAW to update the analytical results.

The future work of this thesis project includes improving the algorithmic efficiency of the case study mobility workflow by leveraging matrix processing libraries in Python such as numpy [73] and pandas [74]. We could also conduct additional benchmarking experiments using different types of AWS EC2 instances and compare the experimental results to have a better understanding of the case study mobility workflow. In addition, the design and implementation of the SEIR model workflow can be improved to

integrate a forecasting module which can perform forecasting over multiple time steps.

Bibliography

[1] Bioconductor, https://www.bioconductor.org/

[2] Hung, Ling-Hong, Jiaming Hu, Trevor Meiss, Alyssa Ingersoll, Wes Lloyd, Daniel Kristiyanto, Yuguang Xiong, Eric Sobie, and Ka Yee Yeung. "Building containerized workflows using the BioDepot-Workflow-Builder." Cell systems 9, no. 5 (2019): 508-514.

[3] da Veiga Leprevost, Felipe, Björn A. Grüning, Saulo Alves Aflitos, Hannes L. Röst, Julian Uszkoreit, Harald Barsnes, Marc Vaudel et al. "BioContainers: an open-source and community-driven framework for software standardization." Bioinformatics 33, no. 16 (2017): 2580-2582.

[4] Grüning, Björn, Ryan Dale, Andreas Sjödin, Brad A. Chapman, Jillian Rowe, Christopher H. Tomkins-Tinch, Renan Valieris, and Johannes Köster. "Bioconda: sustainable and comprehensive software distribution for the life sciences." Nature methods 15, no. 7 (2018): 475-476.

[5] Last JM, editor. Dictionary of epidemiology. 4th ed. New York: Oxford University Press; 2001. p. 61.

[6] BioBoxes, http://bioboxes.org/

[7] Folarin, Amos A., Richard JB Dobson, and Stephen J. Newhouse. "NGSeasy: a next generation sequencing pipeline in Docker containers." F1000Research 4 (2015).

[8] O'Connor, Brian D., Denis Yuen, Vincent Chung, Andrew G. Duncan, Xiang Kun Liu, Janice Patricia, Benedict Paten, Lincoln Stein, and Vincent Ferretti. "The Dockstore: enabling modular, community-focused sharing of Docker-based genomics tools and workflows." F1000Research 6 (2017).

[9] Anaconda Software Distribution, https://anaconda.com/

[10] Amazon Web Service (AWS), https://aws.amazon.com/

[11] Bitam, Salim, and Abdelhamid Mellouk. "Its-cloud: Cloud computing for intelligent transportation system." In 2012 IEEE global communications conference (GLOBECOM), pp. 2054-2059. IEEE, 2012.

[12] Ball, P. "High-profile journals put to reproducibility test." Nature (2018).

[13] Meiss, Trevor, Ling-Hong Hung, Yuguang Xiong, Eric Sobie, and Ka Yee Yeung. "Software solutions for reproducible RNA-seq workflows." bioRxiv (2017): 099028.

[14] Netto, Marco AS, Rodrigo N. Calheiros, Eduardo R. Rodrigues, Renato LF Cunha, and Rajkumar Buyya. "HPC cloud for scientific and business applications: taxonomy, vision, and research challenges." ACM Computing Surveys (CSUR) 51, no. 1 (2018): 1-29.

[15] Ban, X., C. Chen, F. Wang, J. Wang, and Y. Zhang. "Promises of Data from Emerging Technologies for Transportation Applications: Puget Sound Region Case Study (No. FHWA-HEP-19-026)." Federal Highway Administration report FHWA-OR-15-01 (2018).

[16] Gonzalez, Marta C., Cesar A. Hidalgo, and Albert-Laszlo Barabasi. "Understanding individual human mobility patterns." nature 453, no. 7196 (2008): 779-782.

[17] Kang, Chaogui, Yu Liu, Xiujun Ma, and Lun Wu. "Towards estimating urban population distributions from mobile call data." Journal of Urban Technology 19, no. 4 (2012): 3-21.

[18] Wang, Jingxing, Feilong Wang, Xuegang J. Ban, and Cynthia Chen. Comparative Analysis of Big and Small (Survey) Data for Deriving Human Mobility Patterns. No. 19-04203. 2019.

[19] Chen, Cynthia, Jingtao Ma, Yusak Susilo, Yu Liu, and Menglin Wang. "The promises of big data and small data for travel behavior (aka human mobility) analysis." Transportation research part C: emerging technologies 68 (2016): 285-299.

[20] Fiore, Sandro, Donatello Elia, Carlos Eduardo Pires, Demetrio Gomes Mestre, Cinzia Cappiello, Monica Vitali, Nazareno Andrade et al. "An integrated big and fast data analytics platform for smart urban transportation management." IEEE Access 7 (2019): 117652-117677.

[21] Bray, Nicolas L., Harold Pimentel, Páll Melsted, and Lior Pachter. "Near-optimal probabilistic RNA-seq quantification." Nature biotechnology 34, no. 5 (2016): 525-527.

[22] Pimentel, Harold, Nicolas L. Bray, Suzette Puente, Páll Melsted, and Lior Pachter. "Differential analysis of RNA-seq incorporating quantification uncertainty." Nature methods 14, no. 7 (2017): 687.

[23] GenomeNet, https://www.genome.jp/en/gn_tools.html

[24] BUMHMM: Probabilistic computational pipeline for modelling RNA structure probing data, https://www.bioconductor.org/packages/devel/bioc/vignettes/BUMHMM/inst/doc/BUMHMM.pdf

[25] MSstats: Protein/Peptide significance analysis,
https://www.bioconductor.org/packages/devel/bioc/vignettes/MSstats/inst/doc/MSstats.html
[26] GO Semantic Similarity Analysis,
https://www.bioconductor.org/packages/devel/bioc/vignettes/GOSemSim/inst/doc/GOSemSim.html
[27] Jupyer, https://jupyter.org/
[28] Pypi, https://pypi.org/
[29] the U.S. Bureau of Transportation Statistics, https://www.transtats.bts.gov/
[30] Python, https://www.python.org/
[31] pip, https://pip.pypa.io/en/stable/
[32] Docker, https://www.docker.com/
[33] Docker Hub, https://hub.docker.com/
[34] Tyner, Jeffrey W., Cristina E. Tognon, Daniel Bottomly, Beth Wilmot, Stephen E. Kurtz, Samantha L. Savage, Nicola Long, et al. 2018. “Functional Genomic Landscape of Acute Myeloid Leukaemia.” Nature 562 (7728): 526–31.
[35] Lee, S., Celik, S., Logsdon, B.A. et al. “A machine learning approach to integrate big data for precision medicine in acute myeloid leukemia.” Nat Commun 9, 42 (2018). https://doi.org/10.1038/s41467-017-02465-5
[36] Nichols DJ. Fluid volumes in rainbow trout, Salmo gairdneri. “Application of compartmental analysis.” Comparative Biochemistry and physiology. A, Comparative Physiology. 1987 ;87(3):703-709. DOI: 10.1016/0300-9629(87)90386-0.
[37] Gresch, Markus, Raphael Brügger, Alain Meyer, and Willi Gujer. "Compartmental models for continuous flow reactors derived from CFD simulations." Environmental science & technology 43, no. 7 (2009): 2381-2387.
[38] Silva, Cristiana J., and Delfim FM Torres. "A SICA compartmental model in epidemiology with application to HIV/AIDS in Cape Verde." Ecological Complexity 30 (2017): 70-75.

[39] Watabe, H., Ikoma, Y., Kimura, Y. et al. “PET kinetic analysis—compartmental model.” Ann Nucl Med 20, 583 (2006). https://doi.org/10.1007/BF02984655

[40] Le Moullec, Y., C. Gentric, O. Potier, and J. P. Leclerc. "Comparison of systemic, compartmental and CFD modelling approaches: application to the simulation of a biological reactor of wastewater treatment." Chemical engineering science 65, no. 1 (2010): 343-350.

[41] Bansal, Shweta, Bryan T. Grenfell, and Lauren Ancel Meyers. "When individual behaviour matters: homogeneous and network models in epidemiology." Journal of the Royal Society Interface 4, no. 16 (2007): 879-891.

[42] Brauer, Fred. "Compartmental models in epidemiology." In Mathematical epidemiology, pp. 19-79. Springer, Berlin, Heidelberg, 2008.

[43] Peng, Liangrong, Wuyue Yang, Dongyan Zhang, Changjing Zhuge, and Liu Hong. "Epidemic analysis of COVID-19 in China by dynamical modeling." arXiv preprint arXiv:2002.06563 (2020).

[44] Yang, Zifeng, Zhiqi Zeng, Ke Wang, Sook-San Wong, Wenhua Liang, Mark Zanin, Peng Liu et al. "Modified SEIR and AI prediction of the epidemics trend of COVID-19 in China under public health interventions." Journal of Thoracic Disease 12, no. 3 (2020): 165.

[45] Tuite, Ashleigh R., David N. Fisman, and Amy L. Greer. "Mathematical modelling of COVID-19 transmission and mitigation strategies in the population of Ontario, Canada." CMAJ 192, no. 19 (2020): E497-E505.

[46] Xiangyang Guan, Yuanzhi Ren, Ling-Hong Hung, Cynthia Chen, Ka Yee Yeung, Wes Lloyd. “Building Accessible, Reproducible and Interoperable Mobility Analysis Workflows (MAW) Using Emerging Mobile Device Data.” Manuscript under preparation.

[47] Wikipedia, https://en.wikipedia.org/wiki/Reproducibility

[48] Aron, Joan L., and Ira B. Schwartz. "Seasonality and period-doubling bifurcations in an epidemic model." Journal of theoretical biology 110, no. 4 (1984): 665-679.

[49] Kermack, William Ogilvy, and Anderson G. McKendrick. "A contribution to the mathematical theory of epidemics." Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character 115, no. 772 (1927): 700-721.

[50] Github, https://github.com

[51] JupyterHub, https://jupyter.org/hub

[52] Wang, Feilong, Jingxing Wang, Jinzhou Cao, Cynthia Chen, and Xuegang Jeff Ban. "Extracting trips from multi-sourced data for mobility pattern analysis: An app-based data example." Transportation Research Part C: Emerging Technologies 105 (2019): 183-202.

[53] Hung, Ling-Hong, Jiaming Hu, Trevor Meiss, Alyssa Ingersoll, Wes Lloyd, Daniel Kristiyanto, Yuguang Xiong, Eric Sobie, and Ka Yee Yeung. "Building Containerized Workflows Using the BioDepot-Workflow-Builder." Cell Systems 9, no. 5 (2019): 508-514.

[54] Apple mobility trends dataset, https://covid19.apple.com/mobility

[55] Google mobility dataset, https://www.google.com/covid19/mobility/

[56] New York Times COVID-19 dataset, https://www.nytimes.com/interactive/2020/us/coronavirus-us-cases.html

[57] USA Facts COVID-19 dataset, https://usafacts.org/visualizations/coronavirus-covid-19-spread-map/

[58] WHO COVID-19 dataset, https://covid19.who.int/

[59] Johns Hopkins University COVID-19 dataset, https://coronavirus.jhu.edu/us-map

[60] Inman, James. Navigation and Nautical Astronomy, for the Use of British Seamen. F. & J. Rivington, 1849.

[61] Gendzwill, D. J., and M. R. Stauffer. "Analysis of triaxial ellipsoids: Their shapes, plane sections, and plane projections." Journal of the International Association for Mathematical Geology 13, no. 2 (1981): 135-152.

[62] Gellert, Walter, M. Hellwich, H. Kästner, and H. Küstner, eds. The VNR concise encyclopedia of mathematics. Springer Science & Business Media, 2012.

[63] Snyder, J.P. “Flattening the Earth: Two Thousand Years of Map Projections.” University of Chicago Press, 1997, SN 9780226767475

[64] Wang, Xia, Hulin Wu, and Sanyi Tang. "Assessing Age-Specific Vaccination Strategies and Post-Vaccination Reopening Policies for COVID-19 Control Using SEIR Modeling Approach." medRxiv (2021).

[65] Li, Jingshan, Dennis E. Blumenfeld, Ningjian Huang, and Jeffrey M. Alden. "Throughput analysis of production systems: recent advances and future topics." International Journal of Production Research 47, no. 14 (2009): 3823-3851.

[66] the U.S. Census Bureau, https://www.census.gov/

[67] Lauer, Stephen A., Kyra H. Grantz, Qifang Bi, Forrest K. Jones, Qulu Zheng, Hannah R. Meredith, Andrew S. Azman, Nicholas G. Reich, and Justin Lessler. "The incubation period of coronavirus disease 2019 (COVID-19) from publicly reported confirmed cases: estimation and application." Annals of internal medicine 172, no. 9 (2020): 577-582.

[68] Li, Ruiyun, Sen Pei, Bin Chen, Yimeng Song, Tao Zhang, Wan Yang, and Jeffrey Shaman. "Substantial undocumented infection facilitates the rapid dissemination of novel coronavirus (SARS-CoV-2)." Science 368, no. 6490 (2020): 489-493.

[69] Chinazzi, Matteo, Jessica T. Davis, Marco Ajelli, Corrado Gioannini, Maria Litvinova, Stefano Merler, Ana Pastore y Piontti et al. "The effect of travel restrictions on the spread of the 2019 novel coronavirus (COVID-19) outbreak." Science 368, no. 6489 (2020): 395-400.

[70] Peirlinck, Mathias, Kevin Linka, Francisco Sahli Costabal, and Ellen Kuhl. "Outbreak dynamics of COVID-19 in China and the United States." Biomechanics and modeling in mechanobiology 19, no. 6 (2020): 2179-2193.

[71] Wu, Joseph T., Kathy Leung, Mary Bushman, Nishant Kishore, Rene Niehus, Pablo M. de Salazar, Benjamin J. Cowling, Marc Lipsitch, and Gabriel M. Leung. "Estimating clinical severity of COVID-19 from the transmission dynamics in Wuhan, China." Nature medicine 26, no. 4 (2020): 506-510.

[72] Mobile sensor data in Puget Sound Region, https://www.psrc.org/

[73] numpy, https://numpy.org/

[74] pandas, https://pandas.pydata.org/