Web-based Data Visualization and Optimization Methods for Applications in Urban Planning

by

Wesam Manassra

S.B., Massachusetts Institute of Technology (2013)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 2014 [September 2014]

Signature redacted

Author .....
Department of Electrical Engineering and Computer Science
August 22, 2014

Signature redacted

Certified by...
.....
Dr. Sepandar D. Kamvar
Associate Professor
Thesis Supervisor

Signature redacted

Accepted by..
.....
Dr. Albert R. Meyer
Chairman, Masters of Engineering Thesis Committee

Web-based Data Visualization and Optimization Methods for Applications in Urban Planning

by

Wesam Manassra

Submitted to the Department of Electrical Engineering and Computer Science
on August 22, 2014, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis deals with the problem of enhancing the understanding of urban information through interactive data-driven visualization and planning tools. In the first part of the thesis, I present a series of web-based data visualizations through maps to provide users with thorough understanding of their living environments, and highlight patterns and trends to raise their awareness of how cities evolve over time. In the second part, I focus on tools that support the planning of interventions on the urban level. Specifically, I tackle the problem of selecting the most optimal locations for placing new entities within the city using a modified k-means algorithm.

Thesis Supervisor: Dr. Sepandar D. Kamvar
Title: Associate Professor

Acknowledgments

First and foremost, I would like to sincerely thank my thesis advisor, Professor Sep Kamvar, for his great supervision, guidance, and support throughout my research at the MIT Media Lab.

Big thanks go to my colleagues and friends at the Social Computing Group: Yonatan Cohen, Salman Ahmad, Stephen Rife, Kimberly Smith and Jia Zhang. This work would not have been possible without their mentorship, insightful feedback, and their perspectives on design and technical concepts. Additionally, I would like to thank Pranav Ramkrishnan, who I had the pleasure to work with on several projects and classes. Furthermore, I am very thankful to my academic advisor, Professor George Verghese, for his generous academic guidance and mentorship throughout my time at MIT.

I also want to thank the great friends I had the chance to meet at MIT and elsewhere. They have always been a source of inspiration, and have made my five year experience in Cambridge worthwhile.

Finally, I would like to thank my parents and dedicate this work to them. Their continuous efforts and unconditional love and support has shaped the person I am today. While I will never be able to match that kindness, I hope that this thesis provides a milestone for their investment in me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The commonality between science and art is in trying to see profoundly - to develop strategies of seeing and showing.

— Edward Tufte

1.1 Enabling Data Analysis with Visualizations on the Web

With the ubiquity of information exchanged over the web, and the ongoing harnessing of data around us via various devices and interfaces, the efficient presentation of large magnitudes of information has become increasingly challenging. Data visualizations tackle this challenge by leveraging computer science concepts, statistics, and artistic design to expose hidden patterns and "stories" within the data, and empower the cognition of complex information. Visualizations provide a platform for story-telling, to communicate a narrative, raise awareness, and engage viewers to make decisions [17].

The advances in web technologies and the increasing availability of datasets have made information visualization an essential component of communication mediums, and web-based visualizations have made their way into the fields of politics, journalism, public health, and economics. From mapping human emotions and sentiment [9], to explaining the evolution in life expectancies in the past century [1], interactive visualizations have revolutionized the way we present information, and made it ac-

cessible to the public to conduct analysis and draw insights from data. Additionally, news organizations such as the New York Times, the Guardian have been increasingly integrating visualizations to assist the coverage of stories relevant to the general public [18] [19] [6].

1.2 Data Visualizations for Urban Planning

Our connection to the surrounding urban environment greatly defines the trajectory of our life experiences, and affects our mental and physical health. Cities and towns are social platforms for individuals to form communities and build relationships with one another. Therefore, it is of a general interest to city planners, governments, and citizens to actively and continually study our the ecosystem, and find ways to improve living conditions. The recent "Open Data" movement has bridged the gap between city governments and citizens by making an increasing amount of collected information publicly available. This movement not only improves the transparency and accountability of the urban planning process, but it also enables citizens to become agents of change. Visualizations of this data makes this process easier, and simplify the interpretation of complex information to empower this agency.

1.3 Contribution

In this thesis, I leverage computer science and interactive design methods to visualize and analyze different aspects of the urban environment. The thesis is divided into two parts: one aims to raise awareness of urban information, and the other suggests actions and interventions that can be made on the city level. In the first part, I present a portfolio of web-based visualizations and tools to provide users with a better understanding of their living environments, highlight trends and patterns, and draw conclusions. In the second part, I present tools that enables viewers to initiate change in their urban surroundings. More specifically, I tackle the problem of optimally placing urban entities in the city such as schools, clinics, or coffee shops.

The work presented in thesis is part of a the You Are Here[1] ongoing project at the Social Computing Group at the MIT Media Lab.

1.4 Thesis Outline

The thesis is structured as follows. Chapter 2 describes the architecture of the visualization system. Chapter 3 explores the concept of functional distance. In Chapter 4, a series of health surveillance visualizations are presented. In Chapter 5, we map independent coffee shops. Chapter 6 visualizes the distribution of graffiti incidents within selected cities. Chapter 7 explores the change in property values in New York city over the past five years. Chapter 8 examines the distribution of independent and chain restaurants. In Chapter 9, we evaluate the visualizations by describing the design principles applied, and the user reaction to the ones published. Chapter 10 tackles the problem of optimally placing new entities within the city. Finally, chapter 11 concludes the thesis.

[1] www.youarehere.cc

Chapter 2

Architecture

In this chapter, I discuss the infrastructure of the different system components, on top of which the visualizations are built.

2.1 Back-end Infrastructure

The back-end component of the infrastructure is responsible for the various data processing, validation, and aggregation tasks for each visualization, as well as serving the visualization data to the front-end component. Additionally, the backend manages the different configurations for each visualization, so that they can be easily scaled up to hundreds of cities.

The back-end server uses the Django framework, a Python based web framework to manage those interactions. Django was a preferable choice, due to its support for extendable MVC (Model View Controller) architecture. Additionally, Django provides a highly configurable template system that is a key component in enabling a fast-paced and agile development environment.

The running Django instance is served by Nginx, a scalable and open source proxy server, along with other instances on a shared server at the MIT Media Lab. A SQLite database is used for storing the configurations of each visualization, while datasets are stored as static files, for flexibility and to benefit from the advanced caching mechanisms implemented by browsers on the client side.

2.2 Front-end Infrastructure

On the front-end side, the interactive components of the visualizations are done in HTML, CSS, and JavaScript. Since the visualizations presented aim to provide an engaging and pleasing user experience to the user as well as maximizing readability of the data, the D3.js (Data-Driven Documents) JavaScript library is used [4]. D3 provides an abstract implementation for data binding and document manipulation, and bridges the gap between low level parsing and handling to high level interaction design.

Chapter 3

Functional Distance

3.1 Abstract

The topology of a city and its geographic constraints affect our movement, and pose challenges to efficiently travel from one location to another. Within the context of a map, it is hard to visualize the real travelled distance between two locations, as those constraints make a number locations much farther than they appear to be in practice. This chapter explores how a map can be morphed to account for real travelled distances using a method of transportation. Additionally, the tool presented by this chapter can be used to simulate the travel activity from a reference location over time.

3.2 Background

When one first looks at a map, the closeness of one location to another is usually assessed by the direct line traced between them. Taking a more thorough look, one starts tracing the street directions to reach the destination, and realizes that the actual path travelled is is rarely equivalent to the straight path. Instead, the path might includes turns, changes in elevation, traffic congestions and other parameters that make the destination much farther in reality than is perceived by the map.

In this visualization, I attempt to explore this concept by visualizing the functional

distances travelled within the city using different methods of transportation. For each method of transportation, I calculate the travel distance of the most optimal path between street intersections in the city to the center of the city (the reference node). Using those distances, the map is morphed to reflect the real distances travelled within the city. Locations that are least accessible using the selected mode are reprojected much farther than their original location. On the other hand, locations that are more accessible using the selected method of transportation are reprojected away from their original location with a lesser degree.

3.3 Concept

Given a source Location S and a destination location P, we would like to re-project P to P' where the direct euclidean distance between S and P' is equal to the real distance the one would travelled between S and P using a specific method such as waling, driving, public transportation or biking. Each location is expressed as a tuple of geographic coordinates as $P = \langle \lambda, \phi \rangle$ where λ is the longitude and ϕ is the latitude. Assume that the source location is $S = \langle \lambda_s, \phi_s \rangle$ and that we would like to produce the re-projected destination point $P' = \langle \lambda', \phi' \rangle$.

Using spherical trigonometry, we compute the re-projected destination P' by applying the following set of equations:

$$\phi' = acrsin(sin(\phi_s) \cdot cos(\frac{D}{R}) + cos(\phi_s) \cdot sin(\frac{D}{R}) \cdot cos(\theta)) \tag{3.1}$$

$$\lambda' = \lambda_s + acrtan(sin(\theta) \cdot sin(\frac{D}{R}) \cdot cos(\phi), cos(\frac{D}{R}) - sin(\phi) \cdot sin(\phi')) \tag{3.2}$$

Where D is the actual distance that one would travel from S to P using a selected method of transportation. R is the radius of earth, and θ is the bearing between the source and destination computed as follows:

$$\theta = atan(sin(\lambda - \lambda_s) \cdot cos(\phi), cos(\phi_s) \cdot sin(\phi) - sin(\phi_s) \cdot cos(\phi) \cdot cos(\lambda - \lambda_s)) \quad (3.3)$$

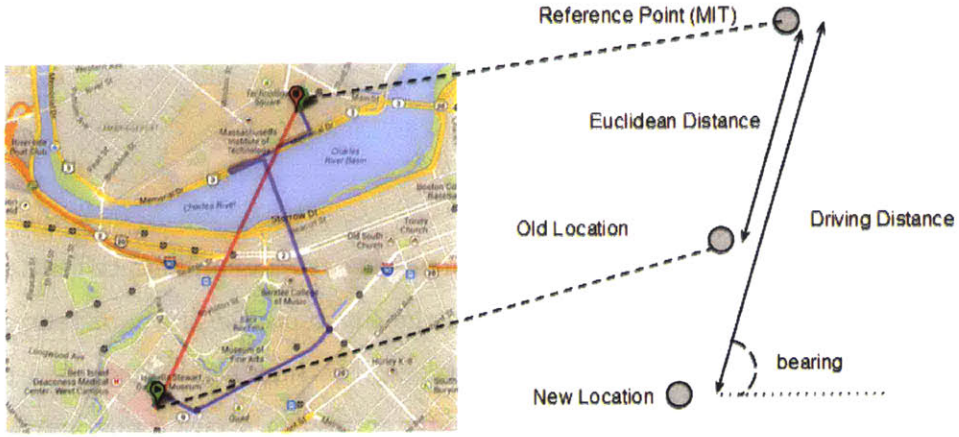


Figure 3-1: The euclidean distance between a location in MIT and another in Boston (red), and the most optimal driving route between these two locations (blue). Shown to the right is how the location is reprojected to reflect the real travel distance

3.4 Data and Implementation

Using the re-projection formulas presented above, we can visualize the the real distance between selected points in a map and a reference point. For simplicity, and to relate the users, I chose to do this on the city level where the reference node is the center of the city. The data collection process consists of two main steps. First, I find all the street segments that lie within the city using the GeoNames Geographical Database web service, where each segment consists of two $\langle latitude, longitude \rangle$ tuples for its two endpoints. Then, given that the road topography for a city can be constructed using the street segments, the list of intersections are found by hashing both endpoints of the street segments into a set.

For each city visualized, a manually selected location is considered the reference node, usually downtown or the city center. Subsequently, the distance and duration from the reference node to each street intersection is computed using the Google

Distance Matrix API. This id done by a sending an HTTP GET request to the web service the query parameters for each mode of transportation. The JSON response is then parsed to extract the distance and duration travelled between the two locations, and the data for each intersection consisting of its geographic coordinates and distance and duration from the city center is written out to the final data file.

3.5 Design

Once all the data of interest is gathered, I created a web-based interface that applies the proposed projection on locations within the city using the travel distances data gathered for several methods of transportation. The interface is composed of two main features: map morphing animation that transforms the city topography, and a travel activity simulation animation, both of which using travel distances by a selected method of transportation. A screenshot of the interface is reported in Figure 3-2

3.5.1 Map Morphing

The first component of the visualization transforms the map of the city by morphing its street topology by distances travelled from the reference node to the rest of locations on the map. The user is able to observe this transformation by clicking the *Morph By Distance* button. Using a smooth transition animation, the locations in the map are translated from their initial location to their reprojected location. The degree of this translation is proportional to the ratio of real travel distance, that one would travel between the reference node, over the direct euclidean distance between the two locations. This way, least accessible location by walking, for example, are reprojected much farther than their initial locations, while locations that are directly accessible are reprojected near their initial location. The dynamic nature of this component engages the user, and provides insight into urban design decisions that can be undertaken to improve the transportation accessibility to a certain area in the city. The morphed map of Cambridge by walking distances is reported in Figure 3-3.

3.5.2 Travel Activity Simulation

The second component is using the travel data for different methods of transportation to simulate how an activity spreads if people were to travel from the reference location to other parts of the city using a selected method of transportation. When the user clicks on the *Animate By* button, a drop-down panel list is shown to select the intended method of transportation to visualize. Say that *walking* is selected as the method of transportation, a time-lapse animation is played coloring intersections in yellow as the walking perimeter widens around the center of the city. To the left, the real time one would takes walking to the farthest yellow colored intersection is shown. As the animation advances, all the intersections in the map are colored yellow, and the timer shows the real time it takes to reach all locations within the city boundary if one would travel from the reference location (in this case the center of the city). A screenshot of this feature is shown in Figure 3-4

3.6 Discussion

While this tool is designed to work with different cities, querying for every single intersection for a large city can be expensive and time consuming. It would be useful if we can create heuristics that allows for inferring travel distances for some intersections using queried distances for a sampled set from the total list of intersections.

Additionally, the implementation of this visualization unwraps the city using the functional distance from one reference location. In practice, we ideally would like to see the functional inter-location distance dynamics between intersections without the need of a centralized node. A potential way to tackle this requirement would be to construct a network from the intersection, and model it as a spring system. The rest length of each edge in the network is then set to be proportional to the distance travelled between the two nodes. This way, the city's topology modeled as a spring system will react to converge towards a minimal energy configuration that reflects travel distance between the different intersections.

3.7 Future Work

Perhaps one of the most interesting extensions to this tool would to be implement the work proposed by [15]. Using travel distances for different modes of transportation, it could be possible to highlight the most optimal locations where the city can place hospitals and ambulances given the geographic constraints, such that a patient in need of care is reached in a fast manner. This extension can provide powerful insights to make health care delivery more effective and more accessible by the general population.

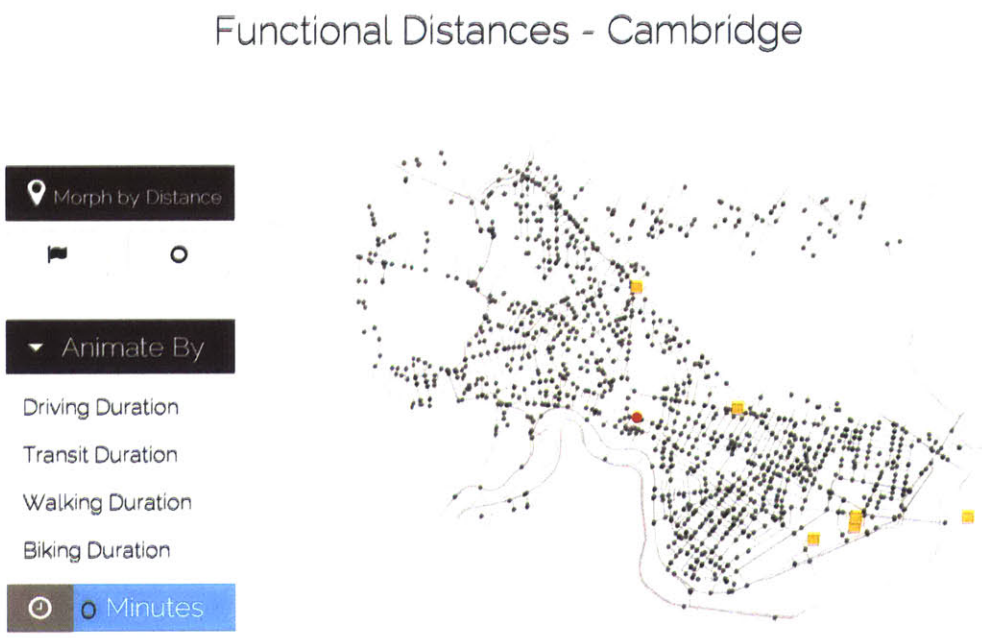


Figure 3-2: The functional distance visualization, shown for the city of Cambridge

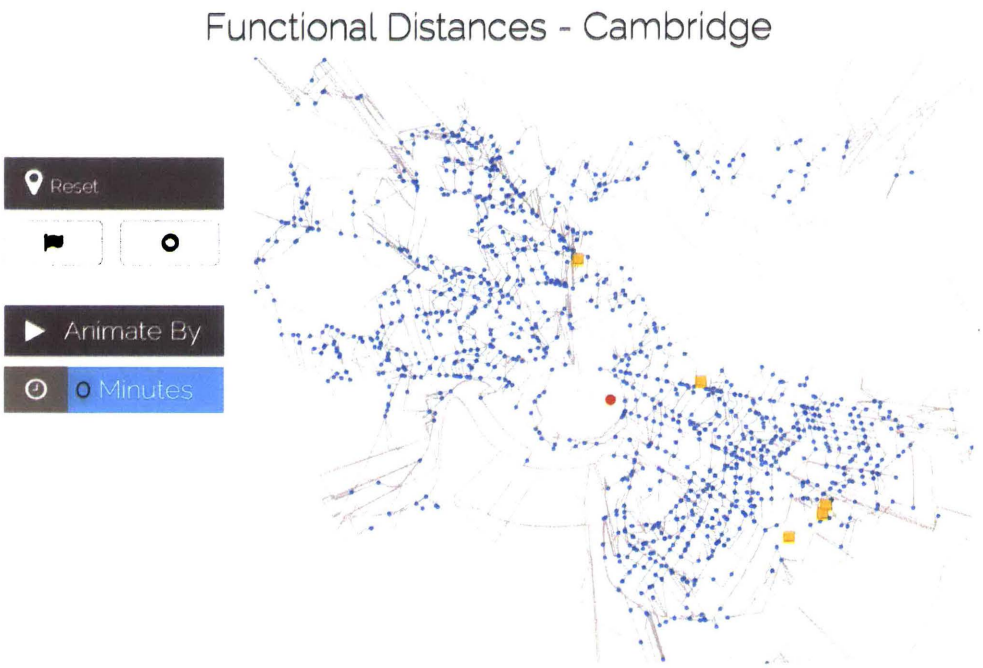


Figure 3-3: Intersections in the city of Cambridge are reprojected by walking distances

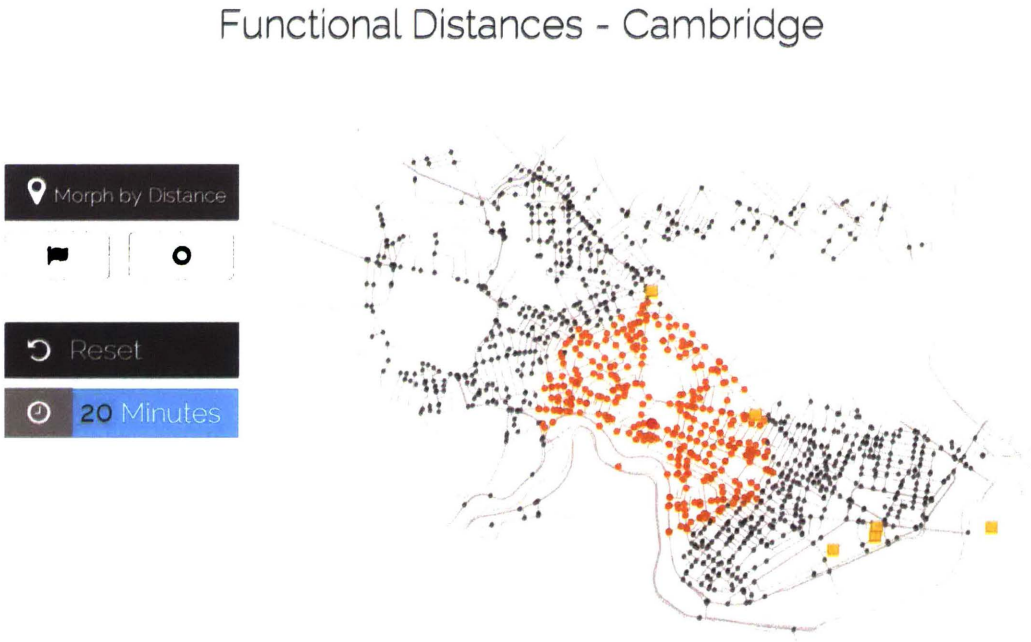


Figure 3-4: The locations that can be reached within 20 minutes from Harvard Square in Cambridge, highlighted in orange

Chapter 4

Health Surveillance

4.1 Abstract

Monitoring our health has been of a growing interest to citizens and providers alike. On the city level, understanding how changes in the environment affect our health is at similar, if not more, importance. This chapter presents a series of visualizations dealing with the distribution of births and the most common diagnoses within the city. Using these visualizations, users are able to compare the relative distribution of diagnoses between the city hospitals, and detect the areas that have higher number of discharges for a selected condition.

4.2 Background

Recently, the 'Open Internet' drive has spawned several dedicated efforts to make health care data publicly available. This push has led data scientists and programmers to draw meaningful insights from such data. One interesting way to draw such insights from large volumes of data is to create striking data visualizations that convey a rich amount of analysis in an effective manner.

Cities have a big impact on our health. We are inseparable from our environment, and unhealthy environments lead to unhealthy people. The World Health Organization, for example, estimates that 7 million annual deaths per year are linked to air

pollution, and a study done by Cornell estimates that water, air, and soil pollution contribute to 40% of deaths worldwide [14] [16].

Cities are also communities of treatment and care. We are born into hospitals, get treated in them and spend our last days in their care - much like schools and workplaces they represent an important part of our collective life experience.

4.3 Data

The primary dataset used by this series of visualizations is the SPARCS (The Statewide Planning and Research Cooperative System) dataset obtained from the New York State Department of Health [11]. This dataset contains detailed de-identified information about discharges coming out of hospitals. For each discharge, patient level information is provided, as well as information about diagnoses, treatment, and payment details [12]. The discharges dataset for 2012 was used, as it was the latest at the time of making the visualization. The detailed and localized information presented by this dataset inspired the different spins taken by each visualization in this series. The overall goal is to create a health surveillance portfolio for the city by monitoring the different diagnoses in a given period, and attempt to make readings from the geospatial and urban level.

4.4 Implementation

This section explores the implementation steps followed in the making of this visualization.

4.4.1 Data Parsing and Aggregation

The dataset used in this series was downloaded in a CSV[1] file format, consisting of tens of thousands of patient discharges. Since the dataset was very large in size (a little over a gigabyte), it was crucial that the data is aggregated in a compact form

[1]Comma Separated Values, a simple file representation for tabular data.

in the back end. An aggregator was built to perform this task, where all the rows of this dataset are condensed on three main levels: the state level, the city level, and the hospital level. For each one of those entities, the data is aggregated to the following fields:

- **Diagnoses Count**: The total number of in-patient discharges at the specified year
- **Conditions**: The list of conditions diagnosed in this location, each consisting of the CCS[2] code and the count of discharges for that code, in descending order
- **Categories**: The list of diagnosis categories, each consisting of the MDC[3] code and the count of discharges to that category, in descending order
- **Birth Weights**: A map describing the distribution of birth weights at the location, with the count of weights recorded within weight bins
- **Age of Mother**: A map describing the distribution of maternal ages when birth is given at the location
- **Delivery Method**: The counts of Cesarean births and non-Cesarean births

Afterwards, the final data is written in a hierarchal format consisting of the state, cities within the state, and hospitals within each city. To avoid redundancy and optimize the data transfer time, diagnosis ids are used and a map from id to description is places on the top level of the dataset.

4.4.2 Hospital Location Geocoding

Due to the geospatial nature of this visualization, addresses for the hospitals were extracted from the dataset, and then converted into $\langle Latitude, Longitude \rangle$ geographic coordinates. Using the Google Geocoding API, an HTTP GET request is initiated

[2]CCS (Clinical Classifications Software) is a classification system that groups procedures and conditions to a few hundred standard diagnoses.

[3]MDC (Major Diagnosis Category) codes aggregate conditions into 25 mutually exclusive categories.

with the extracted hospital address, and the coordinates are obtained from the JSON response returned by the web service. A JSON file is then written containing the id, name, and coordinates for each hospital in the city.

4.5 Design

This section explores and analyzes the design decisions considered in visualizing the diagnosis data. To get a comprehensive understanding of this data, five interfaces were designed, each of which tackles a unique aspect of the data.

4.5.1 Diagnosis by Condition Categories

The first tool made in this series of visualizations aims to show the most commonly occurring categories of diagnosis in New York City hospitals. This visualization presents diagnosis data on three levels: the state level, the city level, and the hospital level.

The user is able to select the location of interest (state, city, hospital) using a drop-down list. The interface is composed of two main panels: the left panel where the hospital's location is highlighted on the map and other secondary information, and the right panel where the most common diagnoses categories are mapped. For each selected location, the data is presented as a two dimensional grid of circles, each of which represents a specific number of diagnosed patients. In the grid, each category is represented by a uniquely colored number of circles that make up the total number of diagnosed visit for that category. The categories are presented in descending order from the most common to the least common at the specified location. Clicking on any circle within one category updates the bar chart on the left, showing the top diagnosis for each category. Additionally, hovering over any circle shows the exact percentage of diagnosed visits for the highlighted category in the selected location as well as in the city, to give the user a comparative understanding of what is more common in some parts of the city as opposed to the rest. The interface is reported in Figure 4-1.

4.5.2 Diagnosis by Condition

Similar to the previous interface, this one explores the relative frequency of actual diagnosed conditions for New York City, and within the city hospitals. For each selected location, the top twenty five diagnosed conditions are shown as a dynamic bar chart from most common to least common, each colored by the category of that diagnosis. For each condition, the colored portion represents the percentage of patients visits for that condition, while a line is shown to represent that percentage for the city. The user can can scroll through the conditions in view by dragging the red slider on top of a smaller histogram below the main bar chart, or by clicking the left and right arrows. To the left, a legend for the diagnosis categories is provided, and hovering over a category highlights the conditions in view that belong to it. The interface is reported in Figure 4-2.

4.5.3 Diagnosis by Hospital

The third interface created in this series deploys more focus to the geo-spatial dimension of the data. While the data is de-identified and it is impossible to infer an exact location for the origin of each discharge, showing patterns using hospital locations is interesting nonetheless and can provide insightful readings, especially for conditions where patients are more likely to go to the nearest hospital than travel to farther ones.

This interface consists of three main components: a map with the locations of hospitals, a histogram showing the percentage of a condition out of total patient visits at each hospital, and a histogram showing the most common diagnoses in New York City order in descending order. The user can visualize the geographic distribution of a certain condition by clicking on any bar in the diagnoses histogram in the bottom. Once a condition is clicked, the hospitals histogram is updated such that hospitals are reordered from the least percentage of visits with that condition to the highest. On the map, a circle is shown on the location of each hospital. Each circle is colored on a linear scale from light blue to dark blue to reflect the percentage of patient

visits associated with the selected conditions, and its radius reflects the total number of patient visits to that location. This two dimensional representation helps users identify outliers and detect areas that have more exposure to a certain condition. For instance, clicking on the 'Asthma' condition, one can see that the Bronx region of New York City has higher levels of patient visits to this condition than other parts of the city. The interface is reported in Figure 4-3.

4.5.4 County Morphing by Condition

In this interface, I explore visualizing diagnoses on the county level within the state of New York city. The purpose of this visualization is to interactively highlight the counties with a high ratio of diagnoses per country population for a selected condition, as well as enable the user to relatively contemplate how this ratio varies from one county to another. Initially, the original map of counties is shown without any distortion. Once the user selects a condition from the drop-down list, the map starts to deform to visualize this ratio for each county using a smooth animation. This morphing is done using a contiguous area cartogram, where the area of each country is either scaled up or down with respect to the diagnoses per population ratio to other counties, as well as to its original area. This was done using Cartogram.js, a JavaScript implementation of continuous area cartograms built on top of D3 [2] [5]. The interface is reported in Figures 4-4 and 4-5.

Since the SPARCS data is de-identified, the diagnoses for each county was counted by aggregating the visits to county hospitals and not to where the patients come from. Consequently, often patients travel far to go to specialized hospitals for treatment, such as cancer and advanced surgery. Without this consideration, counties with more specialized hospitals, such as New York City, might look much bigger than they should be when the map is morphed to a specific condition. To communicate this consideration to the user, the interface shows a histogram of the conditions with the most amount of patient visits from outside New York City.

4.5.5 Births by Hospitals

The last visualization in this series sheds the light on births in the city. The interface explores the distribution of four parameters within the city: delivery method, baby weight, age of mother, and the average number of days the mother was hospitalized. Each of these parameters can be closely related to the environment and the socioeconomic atmosphere, and show interesting patterns within the city. When the users selects a parameter to visualize, a histogram is shown for the hospitals in New York City ordered from the lowest to largest average value for that parameter, with a red line representing the city average. The bars are colored on a color scale from lighter to darker, as well as the circles representing the locations of the hospitals on the map. The radius of each circle represents the total number of births at the location relative to other locations in the city. A two directional mapping interaction is implemented where hovering over a hospital's location in the map highlights its location in the chart for the current parameter and vise versa.

The interface also shows secondary statics for the parameter in view, usually a histogram for the different classes or bins that its value can fall under. When a user hovers on a hospital, this histogram is updated to show the distribution within the hospital. For instance, when visualizing the age of mother, hovering over a hospital shows a histogram of number of mothers for each age bin that were reported for that hospital. Some striking patterns can be read from this map, as for example, most of the hospitals with older mother are located in Manhattan while the maternal age in hospitals in the Bronx seems to be much younger. The interface is reported in Figure 4-6.

4.6 Results

One of the most interesting readings that the births interface provides is the large variability in the delivery method in New York City hospitals. The ratio of cesarean delivery to natural delivery varies from 1:3 in some hospitals to 1:1 in others as shown in Figure 4-7. This reading raises many questions to think about, especially with

many of those hospitals having a minimum of a thousand births reported in 2012. It is unclear whether the source of variance is related to hospital or physician policies, condition of the mother, or whether it is a personal choices. It is clear, however, that this variability at the large scale of data includes a degree of choice.

4.7 Discussion

One of the challenged faced by this series of visualizations is the fact that hospitals are used to geo-spatially map the diagnoses information, due to the lack of location information for each discharge. Using hospitals as a geo-spatial proxy does present some interesting trends, but it does not completely explain if discharges coming from a certain areas are diagnoses with certain conditions more than others. With developing a better model that reflects this fact, the visualization can show the geo-spatial element of this dataset more clearly.

Another challenge is the public availability of healthcare related data. While patient discharge datasets are available for a few states publicly, others incur a license fee for using data. We hope that the increasing openness of data records, and the reform taking place in the healthcare sector will bridge this gap, and we are able to do such analysis for all the United States and globally.

4.8 Future Work

A natural extension to this series of visualizations is to compare and contrast diagnoses trends within the same cities over a number of different years. This way, we able to draw insights into the dynamics and detect potential trends with respect to concurrent short-term or long-term events occurring on the urban level. With such an extension, the authorities can be better informed to make any policy decisions that will lead to a healthier environment.

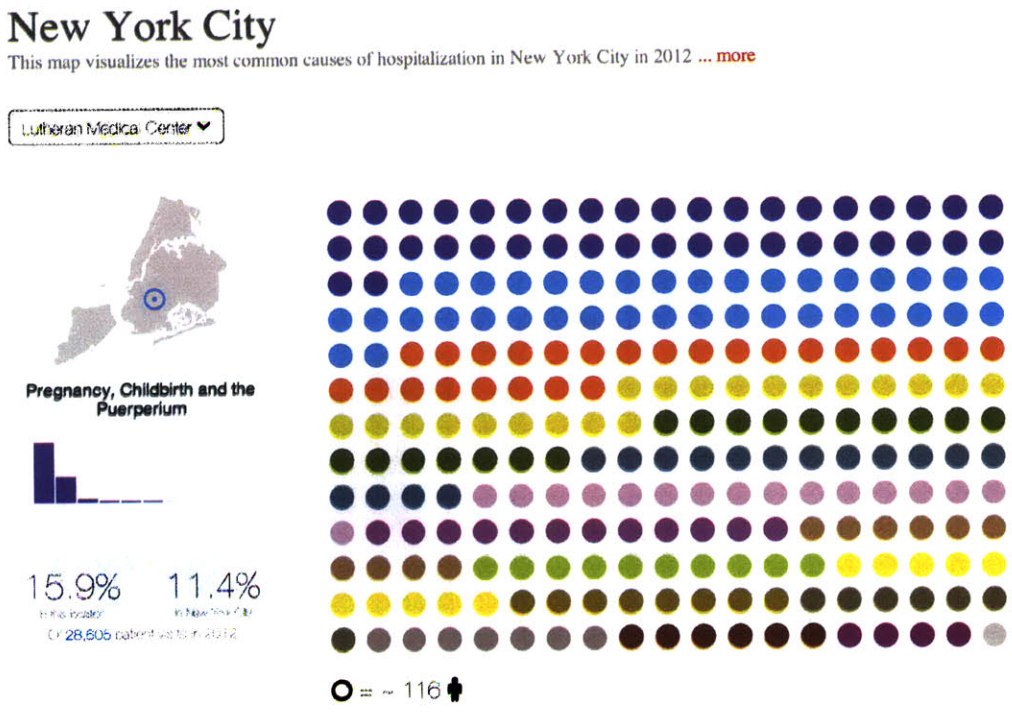


Figure 4-1: The diagnosis categories interface for 2012 New York City patient discharges

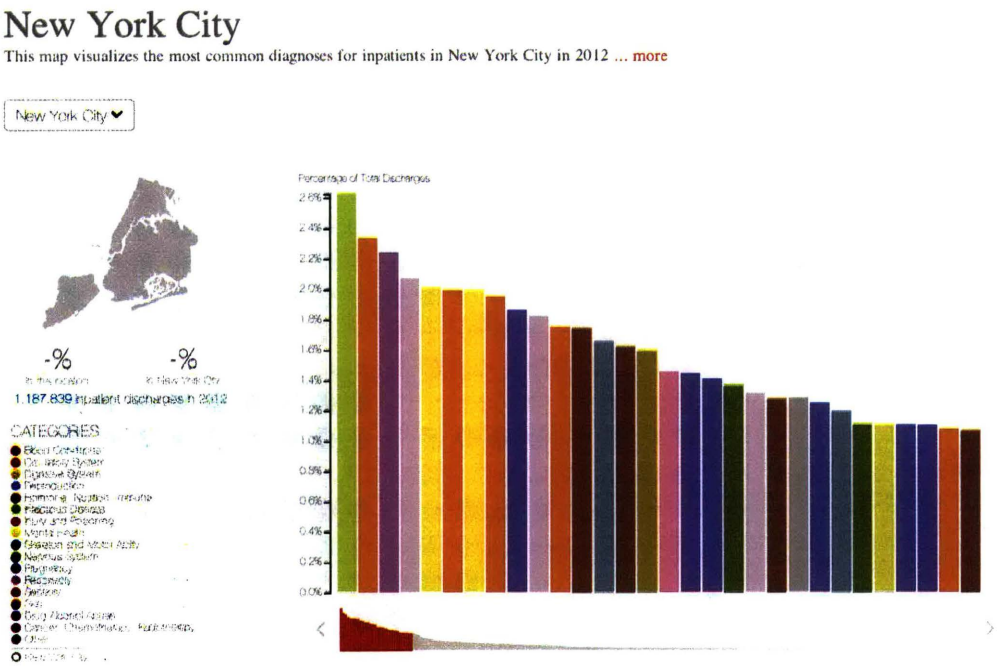


Figure 4-2: The diagnosis by condition interface for 2012 New York City patient discharges

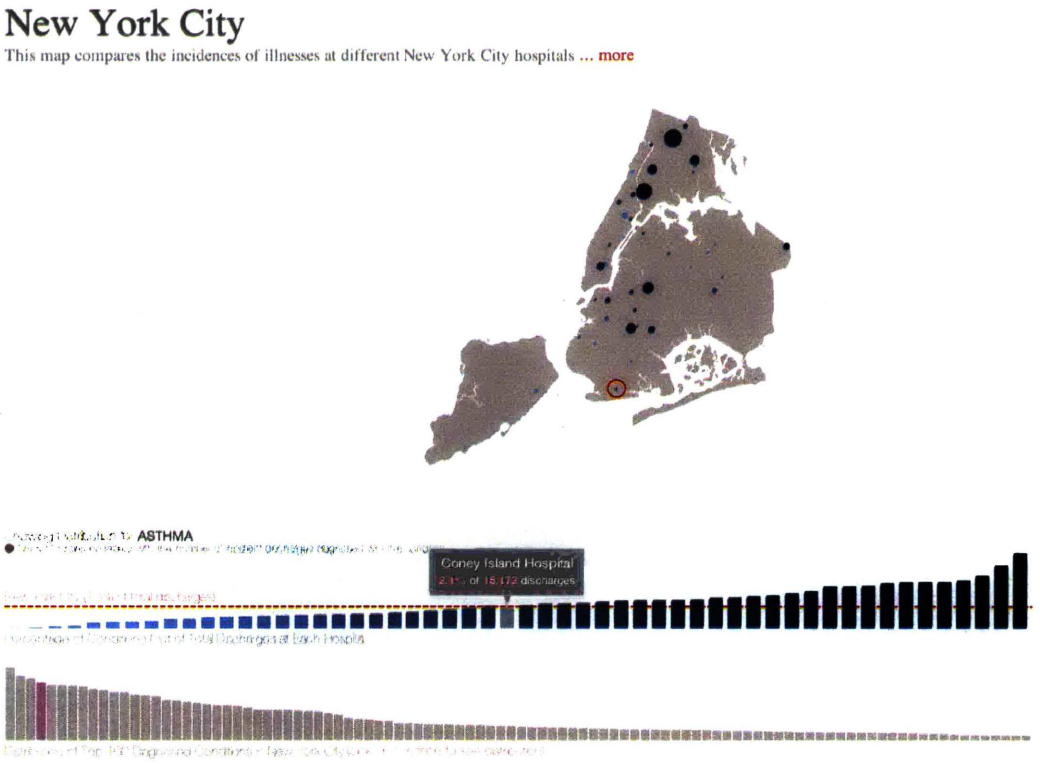


Figure 4-3: The diagnosis by hospital interface for 2012 New York City patient discharges

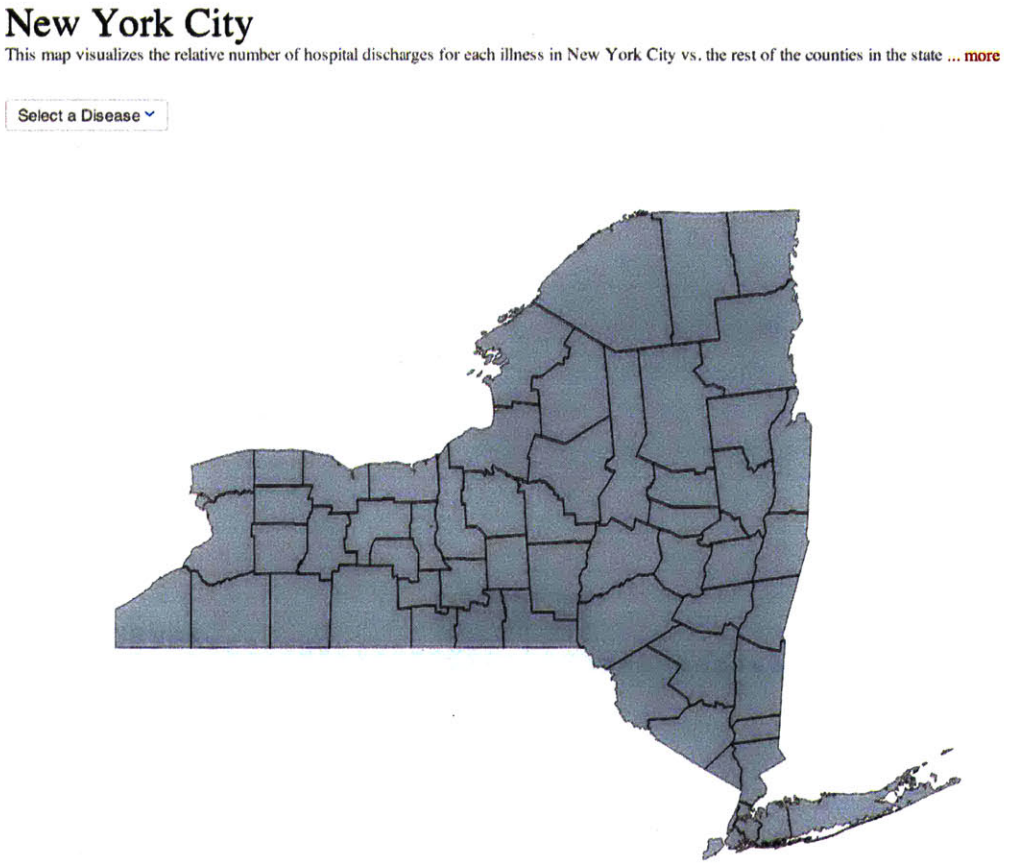


Figure 4-4: The map morphing by condition interface 2012 New York City patient discharges

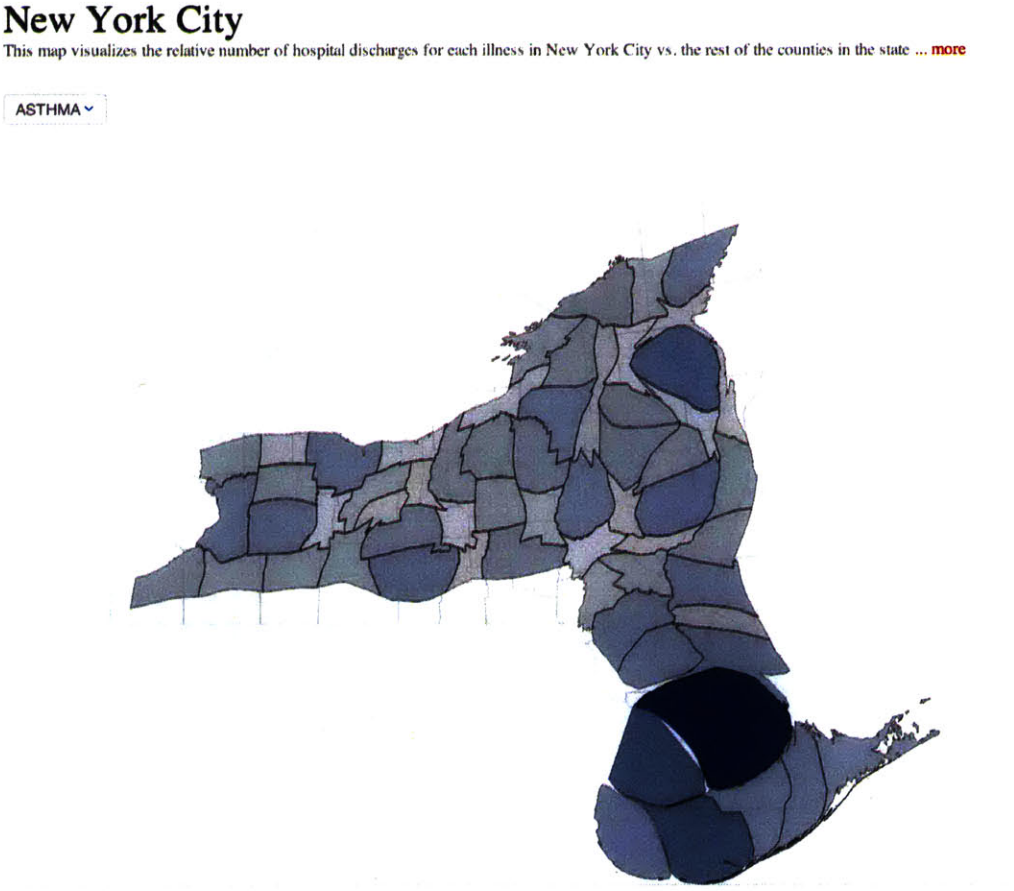


Figure 4-5: The morphed map of New York state for Asthma 2012 discharges

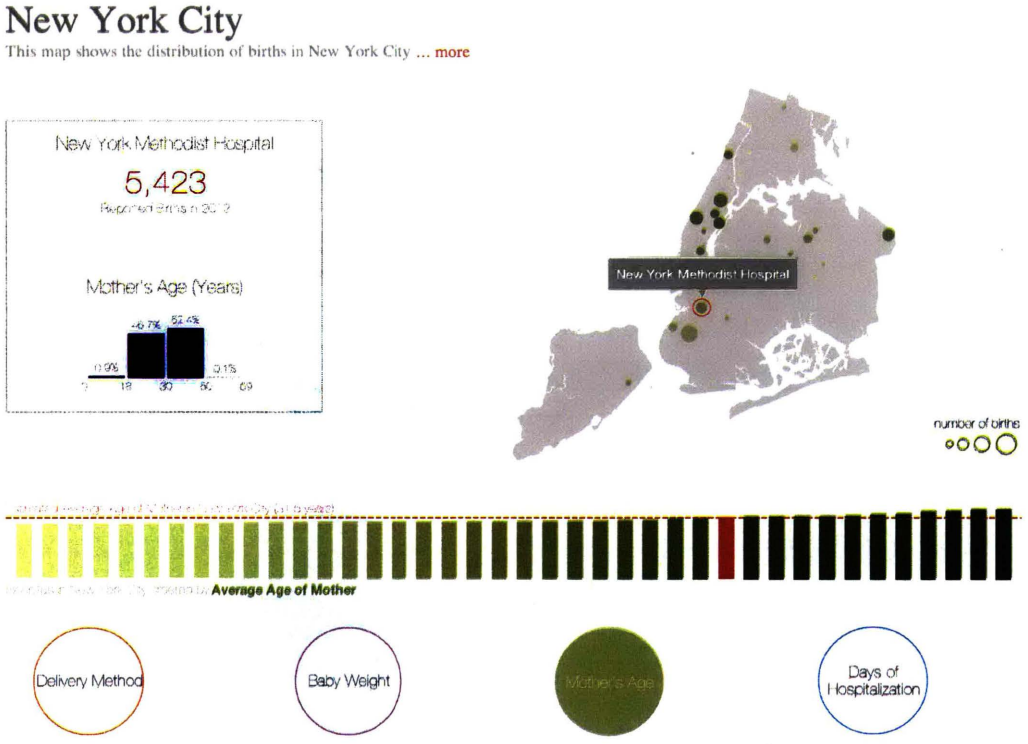


Figure 4-6: The births interface for 2012 in New York City

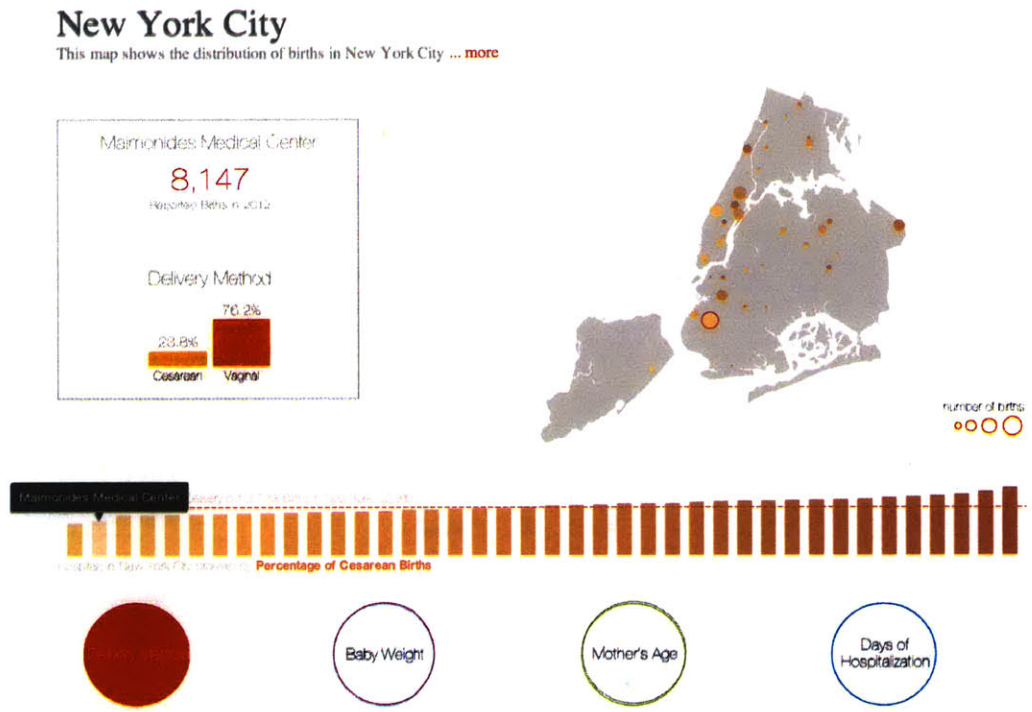


Figure 4-7: The distribution of cesarean and natural birth delivery for 2012 in New York City hospitals, ranging from (22.1%, 77.9%) in Staten Island University Hospital to (46.4%, 53.6%) in University Hospital of Brooklyn

Chapter 5

Independent Coffee Shops

5.1 Abstract

Coffee shops are not only places where people get hot coffee and other beverages. They are places for productivity and social encounters. This chapter explores the distribution of independent coffee shops in the city, and shows the walkability regions around them. This visualization can be used to identify locations that would benefit for a new establishment, or existing places that can benefit from better walking accessibility.

5.2 Background

An important parameter to assess the quality of living in a city, is the walkability; the degree to which the different urban amenities are accessible to the general public via walking. In a time where automobiles are becoming more expensive environmentally, and shortage of parking spots is developing especially in crowded cities, more importance is given to this term when a person decides their living arrangements. More focus is drawn in choosing homes where parks, schools, and similar structures are within walking distances.

On the the other hand, independent coffee shops are positive markers of a living community. They function as social spaces, urban offices, and places to see the world

go by. Communities are often formed by having spaces in which people can have casual interactions, and local and walkable coffee shops create those conditions, not only in the coffee shops themselves, but on the sidewalks around them.

This visualization intends to show where these coffee shop communities exist and where, by placing a new coffee shops, we can help form them within the city.

5.3 Data and Implementation

This section explores the different steps of the implementation process for preparing the input data to this visualization. To generate the input data for a selected city, the process consists of three steps: gridding the city, finding independent coffee shops in the city, and computing walking distances for the grid points.

5.3.1 Creating a Grid over the City

The first step to generating the the data needed by this visualization is creating a grid that covers the city. The process consists of two steps, first of which is generating a uniform set of tuples of latitude, longitude coordinates within the bounding box of the city boundary. The distance between the grid points is selected such that when visualized, the projected distance on the screen between the cells is around three to five pixels, smaller whenever possible.

The second step of the process is to limit the set of points to the ones within the boundary of the city, and discard any point that lies outside the city limits. This is beneficial in restricting our understanding to the city level, and abstract away dynamics that occur outside the city boundary from the representation of our data. Boundaries for major cities in the US are obtained from their open data portals in a shape file format, and is converted to a GeoJSON[1] format. Discarding outside points is done via a script that loads the boundary GeoJSON file, and applies a point in polygon test to each point in the generated grid.

[1]GeoJSON is an open web standard for representing geographic features in JSON (JavaScript Object Notation).

5.3.2 Finding Coffee Shops within the City

Once the grid covering the city is computed, I run a script that to find the nearest coffee shop to each point in the grid. After looking for a scalable and reliable data source to list the coffee shops within the city, I decided to use the Google Places REST API via its *Near By* module. The API takes as an input a location, a ranking flag, and a business type, and returns a JSON response containing an array of places within the vicinity of that location ordered by the ranking flag. For each place returned, it provides it's name, geo location of latitude and longitude coordinates.

As the Google Places REST API provides a ranking by distance for the places returned for a certain grid point, I initially initiated a query for each point in the grid. This was problematic for two main reasons. First, the Google Places API limits users from initiating too many requests in a given day, and therefore conserving queries was a priority. Second, using direct distance (such as Euclidean or Haversine distance[2]) to find the closest place to each point proved to be a good estimation on the city level. Therefore, the script was modified such as that each grid point is assigned to the closest coffee shop based on the Haversine distance.

5.3.3 Finding Distances to Coffee Shops

To find the walking distance between each grid cell and it's corresponding coffee shop, I used the Google Distance Matrix API. For each grid point, a query is initiated with the origin parameter set to the latitude and longitude of the cell points, and the destination parameter to the coordinates of the coffee shop and transportation mode set to *walking*. The result of the query is a JSON object that is parsed to get the distance and duration taking to walked from the origin to the destination.

[2]Similar to the euclidean distance, the Haversine formula computes the direct distance between two locations with accounting for the curvature of earth.

5.4 Design

The section deals examines the layout and front-end design and interactivity implemented for this visualization, reported in Figure 5-1.

The map consists of three layers: the streets and boundary layer, the data layer, and the interactivity layer. The street and boundary are shown to maximize the readability of the map and provide a contextualized understanding for the city to the user. On the data layer, the HTML5 Canvas element is used for maximal rendering performance to show the 'community' surrounding each coffee shop. For each grid point in the city, the geographic coordinates are projected to pixel coordinates, and a circle with its center being at that point. The opacity of each circle is determined by the walking distance travelled from its location on the map to the nearest coffee shop; more transparent the farther it is from the place, and more opaque the closer it is. To allow the user to visually distinguish coffee shop communities, each place is assigned a random color, and the grid points closest to that place are colored as such. This way, the random color generation makes it less likely that two adjacent coffee shops are colored equally.

On the interactive layer, hovering over any circle highlights the perimeter of the community around the closest coffee shops, thereby allowing the user to see the exact area it serves. The perimeter path of a coffee shop is created using an implementation of the classic convex hull algorithm[3]; the algorithm is applied to all the grid points associated to a the coffee shops and finds the path that covers all the points. Clicking on a coffee shop directs the user to the Google Plus page of the place to get more detailed information about it.

5.5 Discussion

The sections discusses some of the challenges experienced in making this visualization, and potential improvements that can be implemented to make it more effective.

[3]Given a stream of points in 2D, the convex hull algorithm determines the smallest subset of points that contain the whole set.

5.5.1 Performance

In the initial phases of implementation steps of this visualization, the rendering process to the grid points was done using the HTML SVG element[4] due to its inherit and powerful interactivity support. This worked well for small cities like Cambridge, but performance issues were faced when dealing with cities with a much larger scale, such as San Francisco and Chicago. As the SVG implementation maintains the presentation of each element rendered in the DOM[5], trying to render tens of thousands of points created delays and inactivity loss, and in some cases crashed the browser. To account for this, the representation and interactivity were broken down into two layers. First, the rendering for large data is done using the HTML5 Canvas element[6], and the interactivity is offloaded into an SVG layer on top of the Canvas layer. The performance gains from this technique were important to provide a smooth experience to the user.

5.5.2 Dependence on APIs

As we discussed in the implementation section, the process of generating the data for visualization relies on external APIs to find the places in a city, and compute the walking distances for those places. Data querying from those APIs takes time and their usage is rate limited, which can be the bottleneck for scaling up to more cities. This dependence can be removed by two considerations. First, querying for places can be eliminated if the full datasets of registered places in the city can be obtained from city departments. Second, it is possible to compute the walking distance locally by obtaining the roads geometry for each city, transforming it to a network topology, and then applying a shortest path algorithm, such as Dijkstra's algorithm, to find the most optimal two paths between two points.

[4]Scalable Vector Graphics: an XML based representation of graphics in web documents.

[5]Document Object Model: A tree representation for the web page and its components.

[6]The Canvas elements allows the fast and dynamic rendering of two dimensional graphics and bitmaps. The Canvas element lacks awareness to the objects rendered into its context.

5.6 Future Work

The walking sheds visualized in this tool are built on the assumption that people will go to the closest coffee shops. In practice, a person's decision to go to a specific coffee shop is not fully based on minimizing the walking distance, but rather includes other variables such as personal preference or coffee shop quality. That is, a person may decide to walk farther in return for a higher quality product or service. While we cannot easily visualize personal preference, the latter variable is easier to account for. For instance, using user ratings as a metric of coffee shop quality, this tool can be extended to visualize walking sheds that are more tuned to reflect not only the walkability to such places in the city, but also the overall quality of those places.

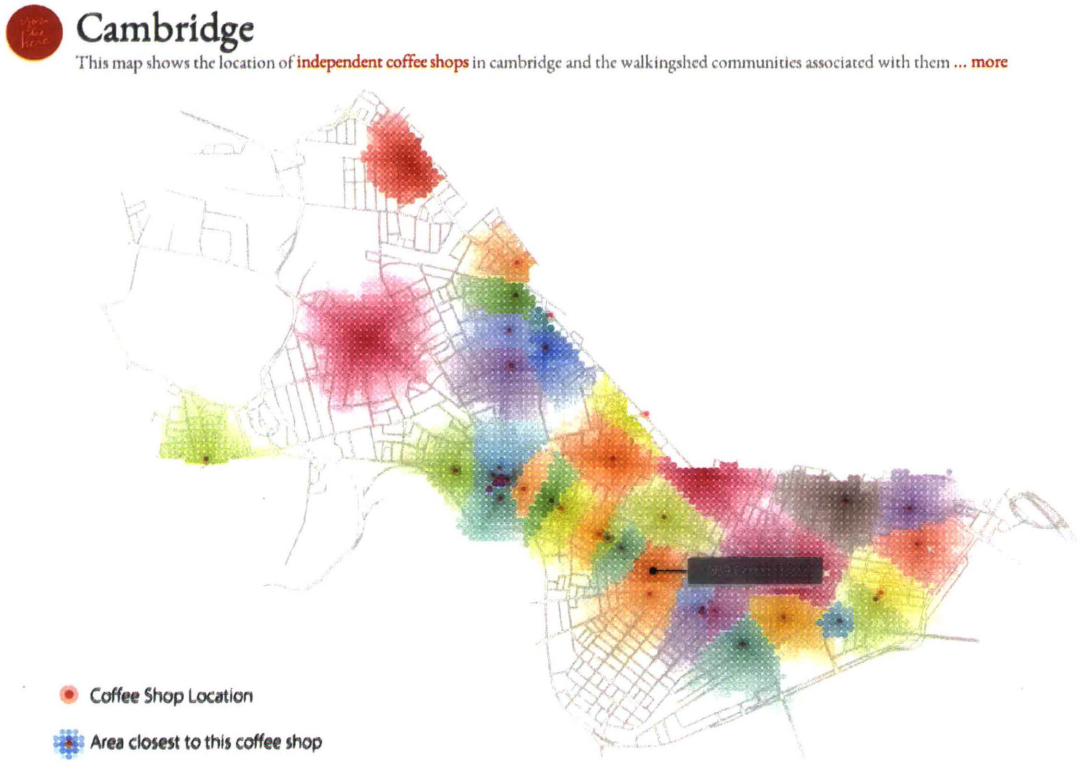


Figure 5-1: Independent coffee shops shown for the city of Cambridge

Chapter 6

Graffiti Incidents

6.1 Abstract

Tracking the incidents of graffiti, from gang signs to intricate art, is a growing interest to many interested parties: city planners, law enforcement, and community members. Graffiti is often visible from the public realm and in some places, dominates a city's street scenery. This chapter discusses a visualization tool that highlights graffiti locations. This tool can be used to analyze how the number of incidents varies over time within specific geographic areas and has been used to visualize data from several major US cities.

6.2 Background

Graffiti is a part of the urban fabric of the city. Ranging from simple scribbles to intricate artwork. Graffiti is often visible from the public realm and in some places, dominates a city's street scenery. While there are hundreds of thousands of reported cases annually, each city presents distinct patterns of distribution of these over time and space.

The tagging of places suggests a distinct need we have as humans to put our mark on the world. The reporting of them suggests a separate need of business owners to not have their property painted on. We might find that addressing this first need will

help to address the second. These maps might suggest places where cities can work to give voices to those without a voice, to create vibrant public spaces for democratized community art, and to explore non-market systems where advertisers don't have a monopoly on street communication. We believe that seeing street art as a need rather than a nuisance, both for its viewers and its creators, could lead to more beautiful cities and a more empowered populace.

On the other hand, of course, some of it is best to be washed off.

6.3 Data and Implementation

The data visualized in this map is obtained from 311 service requests[1] that are related to graffiti removal. For many cities in the United States, 311 service reports are published publicly on city government's data portal, some of which are updated periodically. The availability of this data provided a good basis for implementing this visualization for several major cities such as Boston, Chicago, New York City boroughs, and Seattle. While the data from each city varies with respect to the amount of details provided to each incident, all include the geospatial coordinates and timestamps for which each incident report was opened and closed. The data was downloaded from each source, and standardized to a CSV file format.

After the data is downloaded, the neighborhood of each incident was determined. Using GIS shape file describing the geometry of neighborhoods within the city, a point-in-polygon test is applied on the coordinates of each incident against the list of neighborhood polygons, and a neighborhood id is assigned. Then, incidents are aggregated by the month of their creation time stamp. The final data is then written as a list of objects, each contains a creation time stamp, geographic coordinates, and a neighborhood id.

[1] 311 is a service phone number provided in many cities in the United States where people can inquiry about municipal services and report non-urgent events.

6.4 Design

When mapping graffiti incidents, we are interested in finding patterns that provide some understanding of those incidents within urban contexts either spatially or temporally. To tackle this specification, the interactive visualization presents information on three different levels: micro level information by plotting the location of each incident on the map, macro level information by aggregating incidents by the neighborhoods of the city, and temporally by displaying incidents at different time ranges.

The visualization interface is composed of three main components to achieve this multi-level presentation of data. First, a line chart is located in the upper side of the interface to show the number of graffiti incidents reported over time for each month of the year. The chart itself is interactive: hovering over a marker at shows the number of incidents at its corresponding month, while clicking over a marker updates the map to only show incidents from that month. To the left of the line chart, the user can click the *Play* button to play a time-lapse animation that maps the graffiti incidents from the start to the end of the year. Choosing to visualize the temporal aspect of data using a line chart is effective because it helps the user grasp the month-to-month changes in incidents. Additionally, using the chart as a navigational element simplifies the interface, and emphasizes the data being visualized without having an overwhelming and unnecessary number of buttons and anchors.

The second component of the interface is the map, where every graffiti incident is shown as little circle on top of its exact location. The map consists of two main layers: a layer showing the city boundary and city roads, and a data layer on top. The user is able to pan and zoom through different locations within the city, to see the distribution in a larger resolution. Additionally, street names are shown when the user zooms in to an appropriate zoom level to give a better contextual understanding of the selected area. Initially, the full set of incidents for the whole year is shown to provide a comprehensive view of the overall distribution of those incidents in the city. In areas with high density of graffiti incidents closely located, the data points

are colored red, and purple otherwise. This enriches the visualization by highlighting locations with high graffiti activity.

The final component of the interface is a bar chart where each bar represents the number of graffiti reports for each neighborhood in the city, in descending order. Hovering over each bar updates the map to highlight the incidents for the selected neighborhood, in addition to showing the neighborhood boundary. The reverse interaction is implemented as well where hovering over an incident highlights its corresponding neighborhood on the bar chart. The interface is reported in Figure 6-1.

6.5 Discussion

Due to the large number of incidents that can be reported within one year, the HTML5 Canvas was used for rendering the incidents. This provided improved performance when dealing with large datasets, but per-incident interactivity became very difficult. Unlike SVG, the Canvas element is raster-based. The SVG element, on the other hand, supports more complex types of manipulation and interactivity as each object drawn is also added to the DOM. To improve interactivity, I deployed the use of both elements. The HTML5 Canvas is used to render the data, and an SVG is used to implement neighborhood interactivity when a user hovers over a data point. This way, the interface gets the performance gains while not jeopardizing the interactivity.

6.6 Future Work

This visualization shows a great potential for becoming a crowdsourcing tool, where users can report graffiti incidents in their neighborhood. Additionally, the current representation of incidents does not give an understanding of the graphic nature of the graffiti. If users are enabled to submit photos of graffiti they encounter, the visualization can be much more dynamic and more work can be done to extract and visualize data on the incident level.

Figure 6-1: Graffiti incidents, shown for the city of San Francisco

Chapter 7

Property Values

7.1 Abstract

Detecting changes in property values is an important parameter when studying the economic dynamics within a city. Property values usually change in response to economic booms or stagnation, gentrification, or natural disasters. This chapter explores the changes in property values in New York City over the past five years, from 2010 to 2014. This tool can be used to detect areas with the most increase or decrease in property values over a specific year, and visualize the change dynamics over time.

7.2 Background

On the scale of an individual home, changes in property value of a building tell a financial story. On the scale of the city, changes in property values also tell stories of gentrification, inequality, and climate change, all of which are powerful shaping forces on the city. This map helps us visualize the dynamic of value change over time.

In this chapter, we visualize the changes in residential property values over five years, from 2010 to 2014, in New York City. The goal is provide insights into how property values have changed year by year, and which years has been especially critical to the current assessment of those buildings. In a sense, I aim to provide a reading into those dynamics so that it is easier to reason about what events drive increases

or decreases in those values.

7.3 Data

One of the convenient aspects of this visualization is the availability of extensive data about properties, as cities assess their values periodically for tax collection purposes, and some disclose this information publicly. A city that does a great job in this front is New York City, where the datasets are also downloadable for many years. The primary source of data for the visualization was obtained from the New York City Department of Finance, and includes assessment rolls for the past five years, from 2010 to 2014. The assessment rolls include extensive data about each property, but three fields are central in the implementation of this map:

- **BBLE**: Stands for Borough, Block, Lot, Easement, and can be used as a unique identifier for each property to join the data from multiple years
- **Tax Class**: Determines the type of the establishment, and can be used to identify residential properties from other types (utility, governmental, etc).
- **Full Value**: The assessed market value of the property at the specified time period.

7.4 Implementation

The section explores the implementation steps taken to produce the visualization.

7.4.1 Data Parsing and Aggregation

The initial step of implementation deals with data parsing, preparation and aggregation from multiple datasets. After downloading the raw dataset for each of the five years, they are first converted to a CSV file format. Then, using the *BBLE* field, the assessed market value is obtained for each year, and a single dataset is produced.

properties with zero or invalid assessed values are discarded to avoid divisions-by-zero when visualizing the data.

7.4.2 Geocoding Properties

Much like the rest of visualizations presented by this thesis, all the data visualized was geo-located by having the geographic latitude and longitude coordinates. However, the property assessment data did not have this data and an extra step was needed to find the latitude and longitude of each property. One way to find this data is by using third-party geolocation libraries that convert street addresses to geographic coordinates, such as the Google Geocoding API. While this would be a straightforward method to gather this data, those APIs often impose a limit for the number queries requested in a given period. Instead, the computation is done locally by joining the financial data of a property to the parcel geometry data of the properties provided by the city. For each property, the join key is a concatenation of the borough number, the block number, and the lot number. Once the geometry is joined with the assessed value of the property, the geocoded $\langle latitude, longitude \rangle$ coordinates of the property are computed as the centroid of the geometry.

7.4.3 Neighborhood Assignment

Once the data is geocoded, a neighborhood is assigned to each property. This done by converting the neighborhoods GIS file for the city to the GeoJSON format, loading the file, and performing a point-in-polygon test for each property to find the neighborhood it lies within. The property data is then written in final form where each property is represented by its latitude, longitude, assessed values for a number of periods, and the id of its neighborhood for compactness.

7.5 Design

The core idea of this map is to inform users about the changes in real estate values over time, such that trends can be identified. This is achieved by a thorough presentation of this information on multiple dimensions: spatially by neighborhoods and properties, and temporally by presenting the changes from one year to another. The interface is reported in Figure 7-1.

Spatially, the map initially visualizes the change in property values between the first two years, 2010 and 2011. For each city, there is usually a very large number of properties and showing all them at once can overload the user, and additionally pose interactivity challenges. Instead, the map of the city is divided into a hexagonal grid, where each hexagon covers a set of properties within its proximity. Based on the location of each property, properties are clustered into hexagons using a d3 plugin [3] that implements hexagonal binning. The median change in property values is computed for properties within a hexagon, and the hexagon is colored on a linear scale from dark red (increase in value) to dark blue (decrease in value), and white when no change in value is reported for the current period.

Below the map, a bar chart representing the median change in values is shown for each neighborhood in the city, ordered from the one with the highest increase in value to the one with the highest decrease in value. Each neighborhood is colored red for an increase in value, and blue otherwise. When the user hovers over a neighborhood, the median change in value is shown, and its geometry is highlighted on the map. Likewise, when hovering over a hexagon a tip is shown with the median change for properties in that hexagon, and the neighborhood of that hexagon is highlighted in purple in the map as well as in the bar chart. This way the user is able to grasp changes in the micro level of a hexagon, as well as the macro level using the neighborhood.

Temporally, the user is able to visualize the change from one year to another by clicking any circle between two consecutive years in the top panel of the visualization, or the overall change by clicking the circle between the first and last year. Additionally, the year-to-year change in values can be played as an animation by clicking

the *Play Animation* button, which smoothly updates the map and the neighborhood chart to reflect the change in values period by period. This temporal presentation of data can provide insightful and thought provoking observations, especially when looking at cities where gentrification has been taking place.

7.6 Results

This visualization provides many interested readings that can be associated with recent events happened in New York City. In years from 2010 to 2012 we see a slow increase in the values of properties of some areas in Brooklyn, especially the northern side, perhaps due to gentrification taking place. Yet the most interesting reading comes from the changes between 2013 and 2014. In late 2012, New York was substantially affected by Hurricane Sandy, and flooding impacted many areas in Brooklyn, as seen in Figure 7-3a from the New York Times [20]. We see that the properties that had a large drop in their values between 2013 and 2014 (Figure 7-3b) are precisely the ones that were located near the flooding.

7.7 Discussion

One of the main challenges faced in the making of this visualization was the large scale of the data, and the design iterations made to create an interface that loads a reasonable amount to be visualized from a web application perspective. Initially, the idea of the visualization was to show the geometry of buildings in the city and extrude the height of the building by its estimated value, such that more expensive properties have a higher height than those with less values. An attempt was done in doing this for the city of Cambridge using WebGL, as shown in Figure 7-3. Even though Cambridge is a relatively small city, and has a small number of properties (around 20,000), severe performance issues were faced that prevented the application from being feasible. First, the GeoJSON geometry files for properties were too large to be loaded into a web application. Second, the consumption of processing resources

to render the properties in WebGL in the front-end, created browser inactivity and often crashed it. To prevent this condition, the rendering process was then done in batches, and yet the frame rate was extremely low to have the visualization be smoothly interactive. As a result, I turned to the current implementation where the data is aggregated per hexagon or neighborhood and in a compact form that is reasonably loaded. For a city like a Brooklyn, the current interface is able to load data of around 300,000 properties in only a few second.

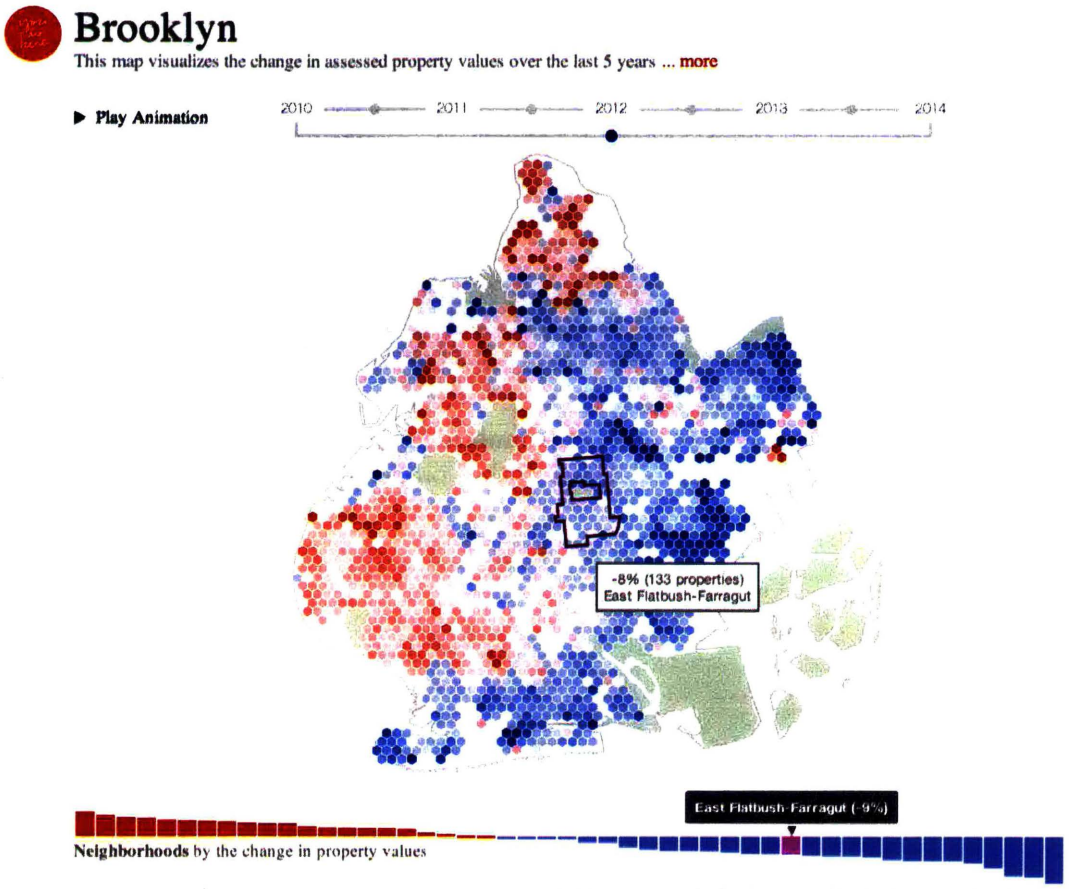


Figure 7-1: Change in property values from 2010 to 2014 in Brooklyn, New York City

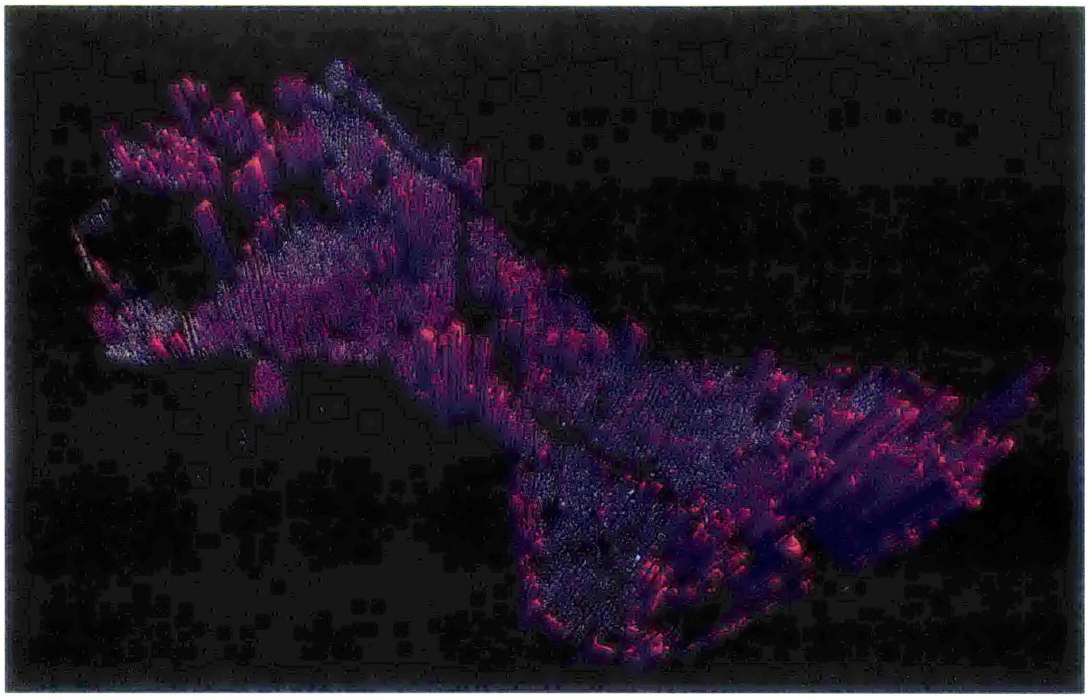
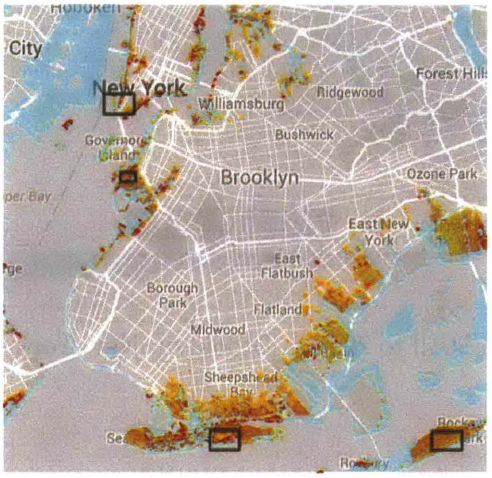
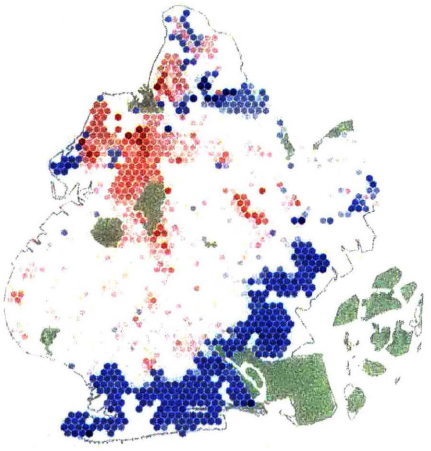


Figure 7-2: Relative property values in Cambridge, MA. Each property is extruded by the estimated market value relative to other properties.



(a) Hurricane Sandy flooded areas in Brooklyn. (Source: New York Times)



(b) Change in property values in Brooklyn from 2013 to 2014

Figure 7-3: Large decreases in property values affected by floods from Hurricane Sandy.

Chapter 8

Chain and Independent Restaurants

8.1 Abstract

Understanding food sources is a critical factor to one's life style. The type of food we eat greatly affects our health, and determines many of the cultural encounter we have with one another. This chapter explores the distribution of restaurants in the city by showing the locations of independent and chain businesses. This visualization can be used to identify the urban areas with a higher ratio of chain to independent restaurants.

8.2 Background

Independent restaurants tend to serve their local communities and repeating clientele; they develop their menus and taste gradually through countless interactions with local clients and suppliers.

National restaurant chains and franchises serve a much larger audience and develop menus that will fit the majority of customers within the territories in which they operate. At the same time, they can bring a sense of familiarity to visitors, and their economies of scale can reduce the costs to eating out.

This visualization was developed to portray the ratio, density and accessibility of chain and independent restaurants across different cities.

Our observation is that chains are often clustered in parts of the city that are throughways – such as streets that are frequented by commuters, squares that attract tourists, and malls. And the presence or absence of fast-food chains tends to be a reflection of the economics of a neighborhood and the prevalence of cars. As a whole, restaurants are a good proxy for businesses in general. The brand familiarity and cost reduction of chains are useful – at the same time, local, independent businesses are a vital component to the economic health of neighborhoods. A policy environment that fosters local entrepreneurship increases a neighborhood's economic autonomy, resilience and cohesiveness.

8.3 Data and Implementation

This section explores the different steps taken in the implementation process of this visualization.

8.3.1 Creating a Grid over the City

Similar to the implementation of the independent coffee shops visualization in Section 5, the first step is generate a uniform grid that spans over the city. Using a GeoJSON file describing the geometry of the city's boundary, the bounding box of the city is computed and a uniform set of $\langle latitude, longitude \rangle$ points are generated within the bounding box. The points that lie outside the boundary are then discarded using a point-in-polygon implementation. The distance between the grid points is selected such that when visualized, the projected distance on the screen between the cells is around three to five pixels, smaller whenever possible.

8.3.2 Finding Restaurants and Distances

Once a grid is created over the city, the Google Places API Nearby service is used to find the set of restaurants within the vicinity of the city boundary. To conserve the number of requests, a configurable sample parameter is chosen such that a fraction of

grid points are used to query for in the city. For each selected grid point, a request is issued with the following parameters:

- **Location**: The latitude and longitude coordinates of the grid point
- **Types**: The place types to be returned, set to *cafe* or *restaurant*
- **Rankby**: The order in which places are returned, set to *distance*

The JSON responses from the service are then parsed, and a dataset is created for the restaurants in each city, consisting of of the restaurant's id, name, and geographic coordinates.

Following the generation of this dataset, the closest restaurant is determined for each grid point using the direct euclidean distance. Then, the Google Distance Matrix API is used to determine the walking distance between each grid point and the restaurant closest to it. Finally, a file is generated consisting of geographic coordinates of each grid point, the id of its closest restaurant, and the walking distance which between them.

8.3.3 Finding Chain Restaurants

After having formed a large dataset containing the restaurants in multiple cities, the next step was to find chain restaurants and label them as such. As with most APIs that return businesses within the perimeter of a location, the data provided lacks this classification. To tackle this challenge, I built a crawler that scraped content from Wikipedia articles containing the keywords *Restaurant* and *Chain*, and extracted restaurant names within those articles. This provided an initial corpus of chain restaurant names which can be checked against when determining whether a specific place is more likely to be a chain or independent restaurant.

Since the list of chain restaurants tended to mostly include national chains, another step was needed to augment the list with local chain names. This was done by adding the names of places that have five or more locations within each city to the chain list.

8.3.4 Restaurant Classification

In this step, restaurants within the city are scored against the corpus of chain names from the previous step, to classify them into either chain or independent. After the list of chains was augmented with the local chains, the names are loaded into a set data structure. The restaurants for a given city are then loaded, and the Levenshtein ratio[1] is computed for each restaurant name and each chain in the chain list. The match is then decided by the chain name with the highest score. If the score is higher than a manually set threshold, for example 90%, the restaurant is labeled a chain, and otherwise independent.

Before conducting the scoring process, each name is sanitized, converted to lower case, and special characters were removed. Additionally, stop words such as *restaurant* and *cafe* were removed to avoid skewing the Levenshtein ratio.

8.3.5 Validation

After classifying each restaurant, matches are ordered from highest to lowest score, and a manual inspection takes place to verify that no restaurant was labeled incorrectly. The list of chain names is then updated to include those escaped by this algorithm. This way, the algorithm's performance improves as more data is classified, and chains are detected more robustly.

8.4 Design

The section deals examines the layout and front-end design and interactivity implemented for this visualization, reported in Figure 8-1. The layout is divided into three closely related panels, explained as follows.

First, the interface shows the map of the city identifying the areas and locations with independent and chain restaurants. The map consists of three layers to maximize

[1]The Levenshtein distance, also known as the edit distance, measures the similarity between two strings. It is generally defined as the smallest number of characters that need to be removed, replaced, or inserted to get from one string to another.

the readability and interactivity: a streets and boundary layer, a data layer, and an interactivity layer with the restaurant locations. In the data layer, an HTML5 Canvas element is used to render a rectangle on top of each grid point in the city. The rectangle is colored orange if it is closest to chain restaurant, and green if it is within proximity of an independent restaurant. The opacity of the rectangle is computed on a linear scale based on the walking distance from the closest restaurant. The lower is the walking distance, the more opaque is the rectangle, and the higher is the distance the more transparency applied, thereby simulating a shed like effect. To eliminate the edge effect from neighboring rectangles and make the opacity shifts smoother, a gaussian blur filter is applied on the Canvas element.

Using the interactivity layer, on the other hand, the user is able to inspect the *area* of a certain restaurant by hovering on its location. This area is computed by running a convex hull on the points closest to this restaurant, and highlighting the path of the perimeter yielded by the algorithm.

To the right of the map is a binary chart that shows the number of chain and independent restaurants for the city. The chart is updated once the user moves the cursor to a specific location on the map, showing the distribution in the neighborhood of that location.

Finally, the interface shows a bar chart of the neighborhoods in the city ordered by their percentage of chain restaurants out of total restaurants. To ensure the visual consistency of information display and improve its readability, each bar in this chart is colored orange if it has a higher percentage of chains than the overall percentage in the city, and green if it is lower. Hovering over a neighborhood highlights the restaurants within its vicinity, and updates the binary chart to show the distribution in that neighborhood.

8.5 Discussion

One of the main components that this visualization uses to communicate information is the classification of the restaurants to chains or independent businesses. What

poses a major challenge to the scalability of this visualization to hundreds of cities is that the manual data validation and verification process can be time consuming, especially when dealing with large scale cities. The validation step is needed to detect incorrect labels produced by the classification algorithm, and augmenting the chains dataset with new chain names, such that the algorithm's labeling performance is improved for future classification tasks. Tackling this challenge is possible by deploying a crowdsourcing engine to make the validation process faster and more robust.

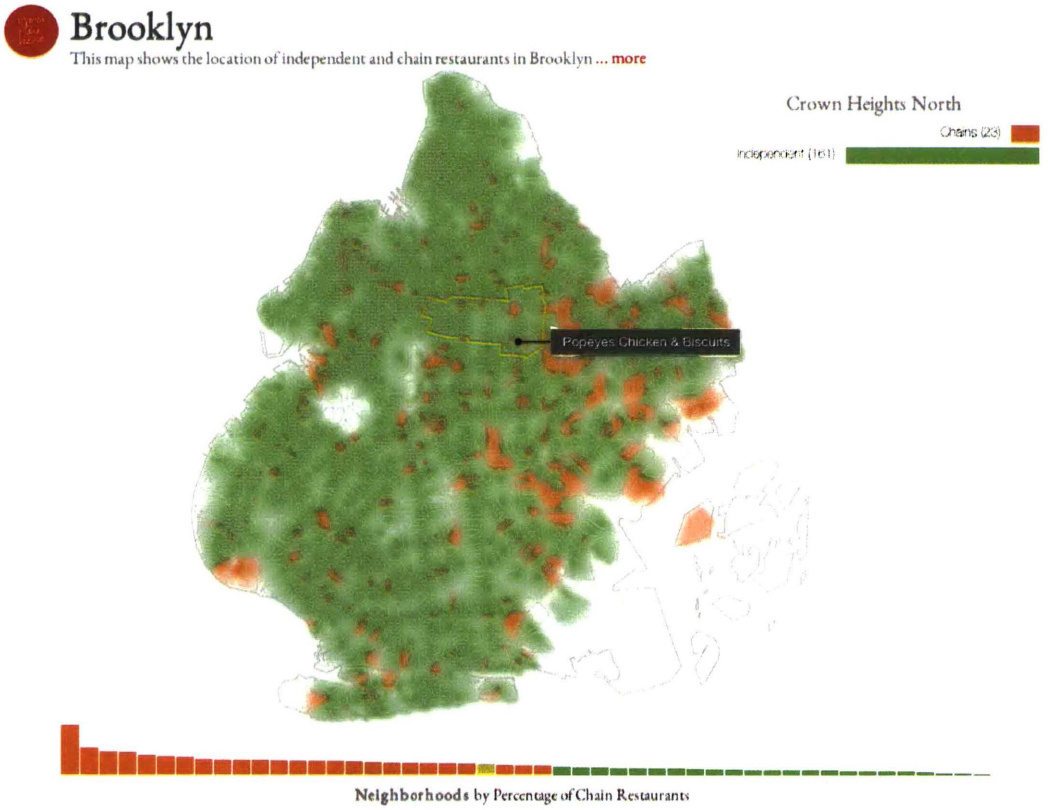


Figure 8-1: Chain and Independent Restaurants, shown for Brooklyn, New York

Chapter 9

Evaluation

9.1 Design Principles

Through the iterative design process to make these visualizations, and by conducting user studies with different individuals within the group, a set of standard design principles started to take place. In this section we list these principles, and describe how they reflected in the implementation of the visualizations presented before.

- **Using Information as Navigation**: Successful and effective design is based on the principle of simplicity, and visualizing information is no different. When a visualization is designed, the data presented is also the interface through which the users interact and engage. Locations on the map, bar charts, and statistics presented all provide access to unique views of the data from different angles.

- **Story Telling**: Every visualization is designed to tell a story. Ranging from the location of coffee shops, the diagnoses of patients, the changes in property values, to where graffiti is reported enables the users to detect patterns and and get insights about different aspects of their living environment.

- **Information Layering and Density**: Everything presented in the visualization conveys meaning, and information is presented in layers. Each layer provides a unique contextual understanding of the data, and exposes a different

reading. The collective experience from all layers provides a thorough understanding of the datasets, and draws focus on the its implications.

- **Interface is Data Agnostic**: Scalability is an important goal for this project. We aim to make visualizations that can be extended to tens and hundreds of cities in the smoothest way possible. Therefore, the layout design and interactions were designed to be agnostic to the source of data. The visualizations are highly configurable such that only a few parameters are changed from city to city, without any effect on the user experience.

- **Visualization State Recovery**: When navigating through a visualization, users are usually interested in saving the state of the visualization such that information in view can be shared with other users. The visualizations presented in this thesis implement this whenever possible by updating the visualization URL to reflect that state currently in view. For example, when a hospital is selected to visualize its most common diagnoses, the browser URL is appended with the hospital identifier. When a user copies this link into the browser, the interface is loaded with the diagnoses for the selected hospital.

9.2 User Reaction

The visualizations presented in this thesis were published under the *You Are Here* project. *You Are Here* is an ongoing project at the Social Computing Group where new visualizations are published daily for a certain urban topic for a number of major cities in the United States. The project was launched in April, 2014 and according to Google Analytics had a total of **501,028** page views, and **125,700** sessions from **108,385** uniques users as of August 17th.

The published visualizations were featured in various local and national news websites, magazines, and blogs such as Gizmodo, Fast Company Design, Architect Magazine, The Washington Post and BostInno. The independent coffee shop maps in particular sparked some discussion on the Gawker blog on how the distribution of coffee shops is related to gentrification. Table 9.1 shows Google Analytics statistics about the published visualizations presented in this thesis.

Table 9.1: Google Analytics Metrics for Published Visualizations.

Visualization	Number of Cities	Release Date	Total Page Views	Unique Page Views	Average Visit Duration (minutes)
Coffee Shops	7	04/07/2014	135,044	37,029	1:20
Health Surveillance	5	05/04/2014	2,418	2,183	1:28
Graffiti Incidents	8	07/08/2014	2,432	2,204	1:30
Restaurants	6	07/28/2014	1,110	955	02:31
Property Values	5	08/05/2014	404	319	01:13

Chapter 10

Entity Placement Optimization Using k-means

10.1 Background

In the second part of this thesis, I utilize the information presented by the geospatial visualizations to empower an intervention. The various datasets visualized in earlier sections provide insights on how individuals can contribute to the improvement of their living environment, and become part of the change.

A particular intervention that comes to mind is the problem of optimally placing entities within the city. In other words, the problem of finding the best location for the next school, health clinic, park or coffee shop. The location of each of these entities is decided using a different set of parameters. A school might be optimally placed in a location where it serves a large population, walkable from most parts of city, and accessible by the different socio-economic spectrum within the city. A coffee shop on the other hand, might optimally be placed near business activity such as downtown or near a square, or within proximity to densely populated streets and neighborhoods.

In the following chapter, I attempt to tackle this problem by providing a modified implementation of the classic k-means clustering algorithm. Using urban parameters such as population, income, or real estate values on the geospatial level, the algorithm

is able to suggest optimal locations for new entities. Additionally, the algorithm is used in a planning tool where users are able to visualize the optimal locations for adding new biking stations in their city.

The algorithm presented in this part is purely suggestive, and the optimal locations found are not where the final locations for new entities is decided. The purpose of this algorithm is to give urban planners, and citizens alike, a better understanding of the general locations where a certain parameter is maximized within the geometric constraints. The tool aims to mainly provide a more complete data-driven understanding of the environment for users, and help them make informed decisions with respect to the final placement locations.

10.2 Overview of K-Means

The k-means clustering algorithm is an unsupervised learning method that aims to solve the following problem: given a stream of n data points, and given a number k, the goal is to assign the n data points to k distinct clusters. Then, for each cluster, the algorithm finds a centroid point (or mean) such that the sum of distances between the cluster point and the centroid is at minimum.

The k-means algorithm is iterative. At each iteration, two steps are undertaken: the assignment of each data point into a cluster, and then the recomputation of the new means (or cluster centroids). In the initial step, k centroids are selected (randomly, or uniformly) and each data point is assigned to one of those centers by minimizing a distance or an objective function. After the initial clusters are formed, a more accurate mean is computed for each cluster, and becomes the new cluster centroid. As the number of iterations increase, the change in clusters' centroid locations decreases until the algorithm converges to a solution and a grouping is reached. The pseudocode for the algorithm is shown in Algorithm 1.

In this algorithm, $\langle x_1, x_2, \ldots, x_n \rangle$ is the list of data points, $\langle \mu_1, \mu_2, \ldots, \mu_k \rangle$ is the list of cluster means, C_i is the index of the cluster (or mean) that is assigned to the data point x_i. SS is the total sum of squares defined as $SS = \sum_{i=1}^{k} \sum_{C_j=1} (x_j - \mu_i)^2$. ϵ is

Algorithm 1 The K-Means Algorithm

function K-MEANS($\langle x_1, x_2, \ldots, x_n \rangle, K$)
  $\langle \mu_1, \mu_2, \ldots, \mu_k \rangle \leftarrow$ SELECT-RANDOM-POINTS(K) ▷ Seeding Step
  while $|SS - SS_{old}| > \epsilon$ **do**
    for $i \leftarrow 1, n$ **do**
      $C_i \leftarrow argmin_c$ DISTANCE(x_i, μ_c) ▷ Assignment Step
    for $i \leftarrow 1, K$ **do**
      $\vec{\mu_i} \leftarrow \frac{1}{\sum_{C_j=i} |x_j|} \sum_{C_j=i} \vec{x_j}$ ▷ Recomputation Step
  return $\langle \mu_1, \mu_2, \ldots, \mu_k \rangle$

a manually set threshold, for which the algorithm terminates if there is no significant decrease in the sum of the squares at the current iteration.

The goal is to minimize the total sum of squared distances SS between the the mean of each cluster and its assigned data points. This term is also known as the Residual Sum of Squares or *RSS*.

$$\text{Minimize RSS} = \text{Minimize} \sum_{i=1}^{k} \sum_{C_j=1} (x_j - \mu_i)^2 \tag{10.1}$$

Where $\sum_{C_j=1} (x_j - \mu_i)^2 = \sum_{C_j=1} DISTANCE(x_j, \mu_i)$ is the cluster's sum of square distances from the mean, or the optimization function that is desired to be at minimum.

10.3 Spatial Optimizations Using K-Means

In this section, I explain a modified implementation of the k-means algorithm to optimize the placement of entities geometrically. The k-means algorithm is well suited in this context because often, geospatial data is naturally clustered and hubs of activity exist within certain perimeters in a given city. For instance, certain areas of the city are more populated, while other may have a larger concentration of businesses. With this natural observation, one is able to use k-means to identify the optimal locations to place a new entity such that it is within proximity to these naturally occurring clusters. In other words, using real urban datasets we aim to use the k-means algo-

rithm to select the optimal locations within the city where a new entity can effectively serve the population or other parameters. For instance, this can be the selection of location for a new school, a new coffee shop, or a new hospital. Two modifications are implement to tackle this goal. First, the concept of fixed means is introduced to account for already existing entities. Second, the algorithm is modified to optimize for a metric that is given to each input point indicating its population, income, etc. The modified pseudocode is reported in Algorithm 2.

Algorithm 2 The K-Means Algorithm with Fixed Means

function K-MEANS($\langle x_1, x_2, \ldots, x_n \rangle, K, \langle u_1, u_2, \ldots, u_m \rangle$)
  $\langle \mu_1, \mu_2, \ldots, \mu_k \rangle \leftarrow$ SELECT-RANDOM-POINTS(K)
  $means \leftarrow \langle \mu_1, \mu_2, \ldots, \mu_k \rangle \cup \langle u_1, u_2, \ldots, u_m \rangle$ ▷ Seeding Step

  while $|SS - SS_{old}| > \epsilon$ **do**
    for $i \leftarrow 1, n$ **do**
      $C_i \leftarrow argmin_c$ DISTANCE$(x_i, \mu_c \in means)$ ▷ Assignment Step

    for $i \leftarrow 1, K$ **do**
      $\vec{\mu_i} \leftarrow \frac{1}{\sum_{C_j=i} |x_j|} \sum_{C_j=i} \vec{x_j}$ ▷ Recomputation Step

  return $\langle \mu_1, \mu_2, \ldots, \mu_k \rangle$

function COMPUTE-CLUSTER-MEAN(i, $\langle x_1, x_2, \ldots, x_n \rangle$)
  $\vec{\mu} \leftarrow \frac{1}{\sum_{C_j=i} \alpha_j \cdot |x_j|} \sum_{C_j=i} \alpha_j \cdot \vec{x_j}$ ▷ α_j is the geospatial data value for x_j
  return $\vec{\mu}$

10.3.1 Fixed Means

A realistic solution for placements of an entity should also account for where other instances of the entity are currently located in the city and then optimize given these locations. For instance when investigating locations to place a new school, the solution should avoid locations for current schools in the city and then optimize with this added knowledge. Fixed means are essentially locations of existing entities that do not change over iterations of the algorithm. The implemented k-means algorithm attempts to perform clustering around these fixed means. Alternatively, it could

be viewed as the usual k-means algorithm except for the fixed means we skip the assignment step.

10.3.2 Integrating Geospatial Data

In the modified algorithm, the factor α_j used when computing the mean of a cluster represents the weight for a certain geographic coordinate x_j determined by a given geospatial dataset. To simplify analysis, I apply this algorithm using α to signify the population estimate at each given point. This way, we would expect k-means to optimally select locations that are closer to more populated areas in the city.

To implement this application, I used population census blocks for the city of Cambridge. Block data includes the geometric coordinates of the block, and the population count taken during the most recent US CENSUS survey [13]. Block centroids are then used as the candidate points $\langle x_1, x_2, \ldots, x_n \rangle$. In the classical version of the k-means algorithm, we assign points and recompute means by only considering distances. However, in this version of the algorithm, the spatial data distribution at each point (population count in this case) will determine the degree to which a given mean is recomputed to be near a certain location. In effect, this aims to change the optimization function to minimize distance as well as maximize the population served by the mean. In a scenario where population is constant, for example, the algorithm will work exactly like classic k-means. When population is higher at certain blocks, the solution will tend to be closer to those areas.

10.4 Analysis

In this section I analyze the performance of the algorithm as well as reason about its robustness.

10.4.1 Convergence and Correctness

While the K-means algorithm ought to terminate after a sufficient number of iterations, and produce a stable clustering solution, it is not guaranteed that the solution produced is the most optimal one (k-means is NP-Hard). That is, depending on the initial seeding of means in the algorithm, the final computed cluster means is always guaranteed to be a configuration at a local minimum, but not necessarily the global minimum for the RSS term. Since the initial selection of means in the first iteration influences the final configuration, it is important to investigate how the seeding step can be used to improve the outcome of the algorithm.

There many ways that one can select the means in the seeding step. One way is to randomly select a vector of k positions in the space of data points using a uniform random distribution. Another is to evenly sample k positions in the space of data points such that each mean is as far away as possible from all other means. A third way would be to have the user identify the initial locations of the means. Each of these selection methods affects the final outcome of the algorithm, but some can be more helpful than others in different contexts. For instance, even sampling can be beneficial in contexts where the data points are sparse. However, since no assumptions can be made to the data sets used by this algorithm, this seeding method might not be the best candidate.

The random sampling method, on the other hand, can be made more robust by running the algorithm multiple times for each random starting positions, and then selecting the solution with that minimizes the distance function (or the one that yields the most optimal value for the objective function). The effectiveness of this technique is evident from Figure 10-1 where the final RSS score is shown for different runs of the algorithm at random seeds, for six clusters using population data. Looking at the figure, one can easily identify the different minima of the RSS term that the algorithm hits for a thousand separate runs, and see that only a few number of runs are needed to reach the global minima where the clustering is most optional.

The last method, user input, can be very helpful in contexts where the user has

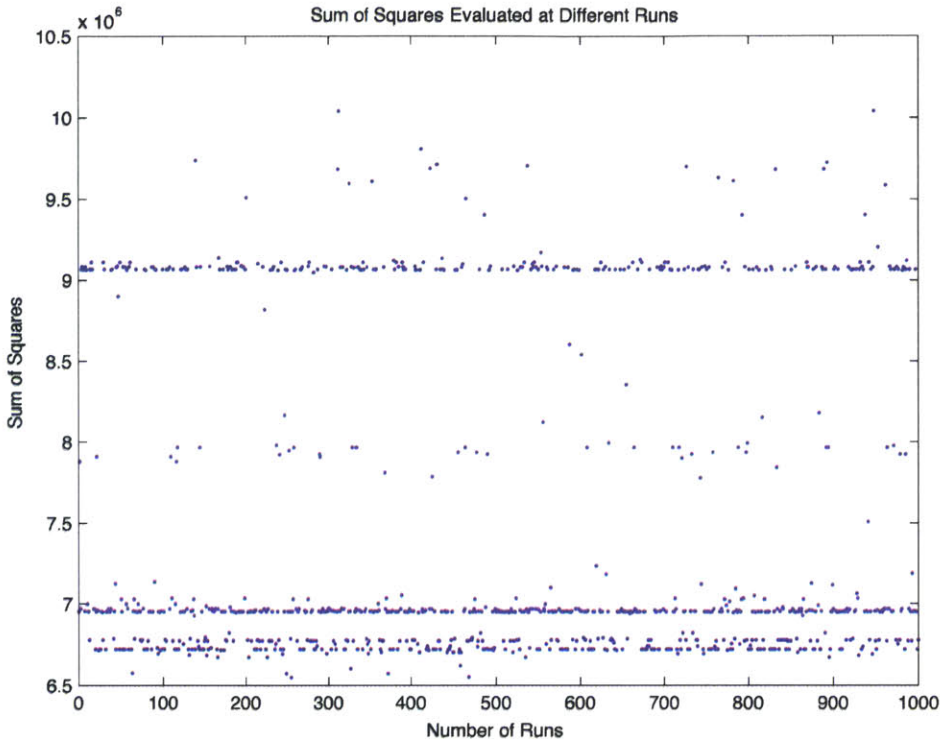


Figure 10-1: The converged RSS value at different runs of the k-Means algorithm using random seeds for six means. The clustering configuration that minimizes the RSS globally is reached every few runs

initial guesses on the locations where urban desired to be places would be in close proximity. For instance, these guesses can be the centers of neighborhoods in the city where the user, say an urban planner, would like to introduce schools.

10.4.2 Time Complexity

The runtime of the k-Means algorithm depends on four main variables: the number of data points N, number of clusters K, dimensionality of the space M, and number of iterations needed for the algorithm to converge I. For each iteration, the assignment step dominates the runtime as KN distances are calculated to assign points to clusters, each of which requires $O(M)$ floating point operations. In the recomputation step, on the other hand, each mean is computed by iterating over the points in each

cluster. This operation takes NM computations. The runtime of the algorithm per iteration then is $O(KNM)$, and thereby a total runtime of $O(IKNM)$. Since we use this algorithm in two dimensional space, only two distance computations are needed for each data point, and thus $M = 2$. Additionally, the number of iterations needed for the algorithm to converge are usually very few in practice. For instance, as shown in Figure 10-2, the algorithm converges in around 6 iterations when running it on CENSUS population data per blocks for six means. Therefore, the realistic runtime of the algorithm is $O(KN)$. The computation time spent over iterations is reported in Figure 10-3.

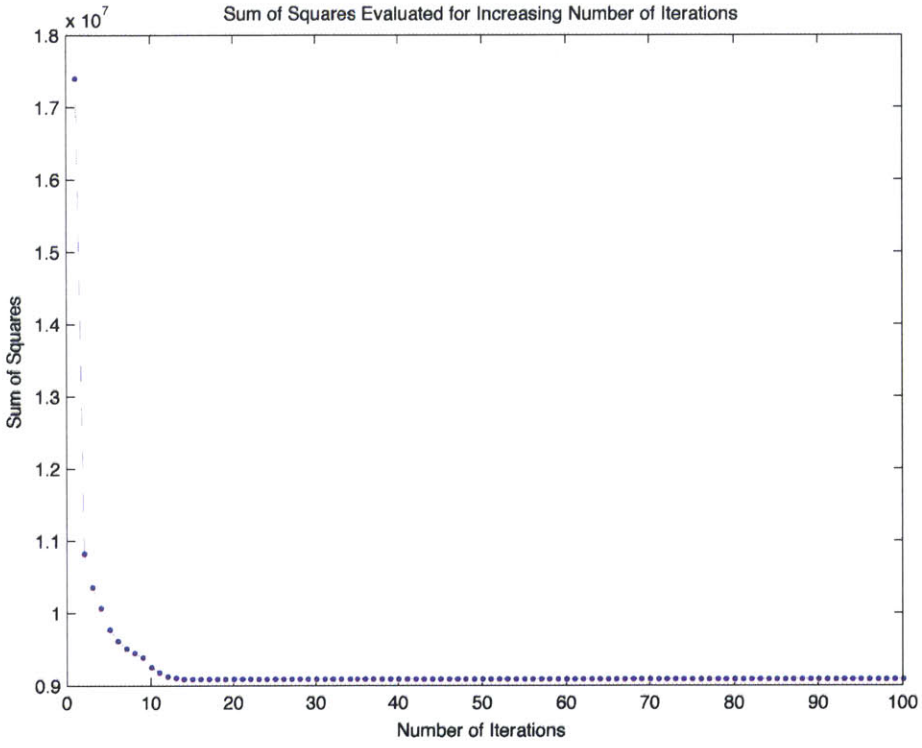


Figure 10-2: The convergence of the k-Means algorithm on CENSUS population data for six means after approximately 15 iterations.

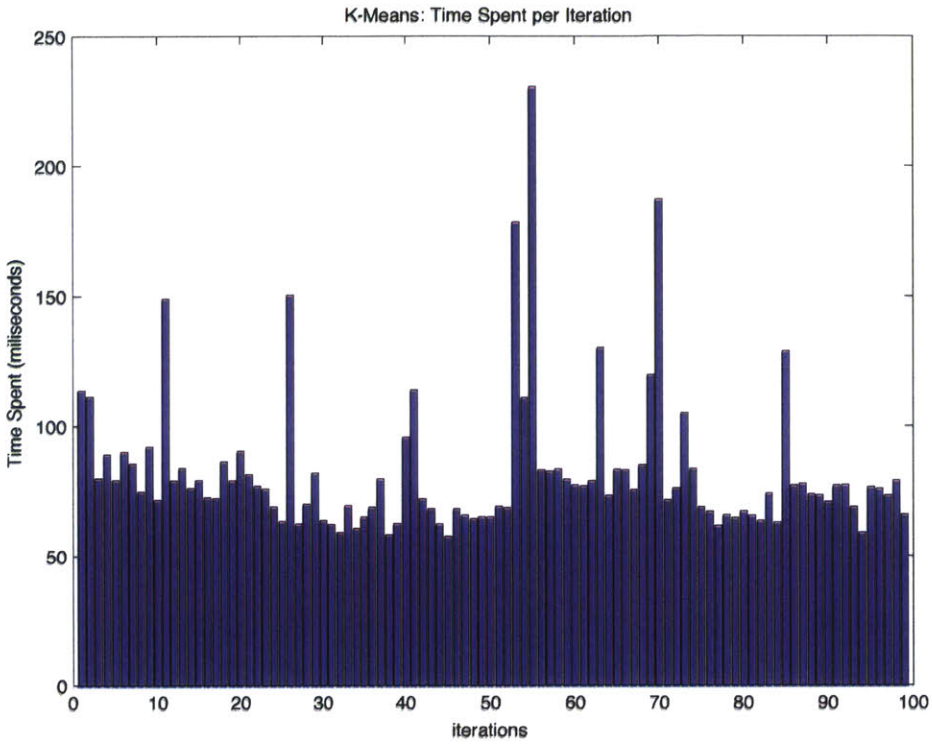


Figure 10-3: The average time spent in computation per each iteration in milliseconds for six means.

10.4.3 Number of Entities

When deciding for placing new entities within a city, we are usually interested to see how the benefit added by one entity can be compared with the cost to place that entity (building or maintenance costs, for instance). More precisely, we are interested to find the number of entities after which the return to scale is zero. That is, when the cost of adding a new entity exceeds the benefit generated by its service.

In the k-means algorithm, when adding more clusters, the algorithm will always yield a lower *RSS*. That is, for increasing values of K, the total distance between cluster points and their means decreases, to the point where $K = n$, after which overfitting takes place. To integrate cost, we need to instead evaluate the function $f(K) = RSS + \alpha K$, where alpha is a parameter proportional to the cost introduced by adding an extra entity in the city. Once this value is determine, we are able to

find the optimal number of entities that can be added while the benefit exceeds the cost. Figure 10-4 hypothetically shows $f(n)$, where the optimal number of entities is 7.

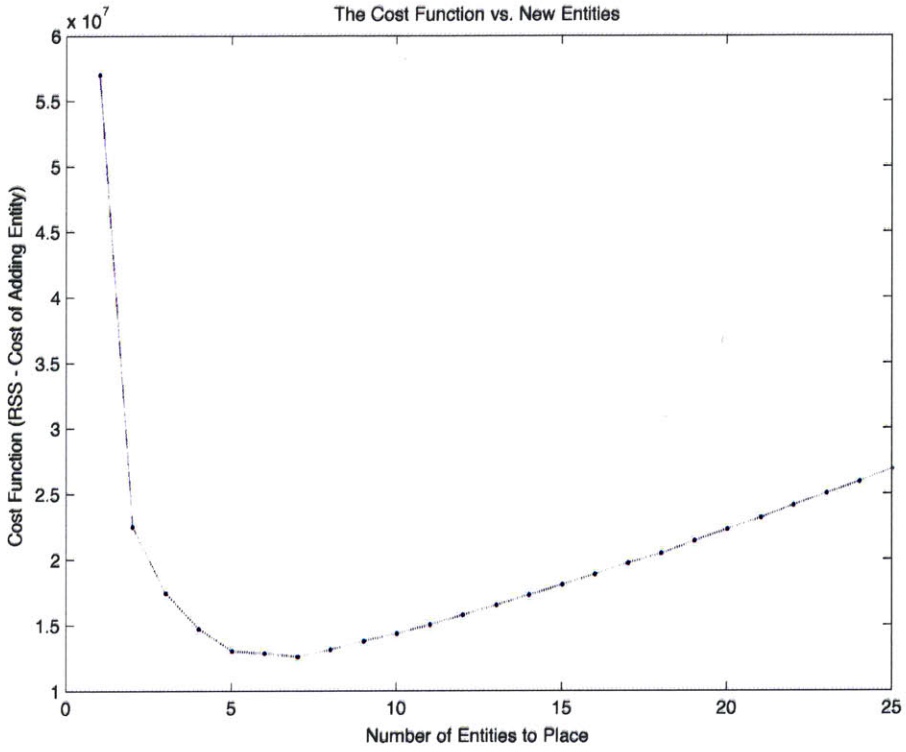


Figure 10-4: Net benefit for adding new entities.

10.5 Use Case: Biking Stations

With the ever increasing population densities in cities in the United States and the relatively high costs of automobiles, bike sharing programs are particularly growing in popularity recently. Additionally, specialized companies such as Hubway and Citibikes have put a lot of resources into putting these programs into larger scales in major cities, and biking stations are continuously added to new areas.

The problem of placing a biking station is a perfect use case for the modified k-means algorithm presented in earlier sections. It is particularity relevant because the

dynamics of adding a new station is usually controlled by two factors: the population it can server, and whether it can be placed closest to activity centers within the city.

To simplify the application, I tackle the first factor of which the locations new bike stations are optimized to serve the most population. To start, the list of already existing Hubway stations is obtained for the City of Cambridge [8]. Using the CENSUS population blocks used from the before, we can find the most optimal locations for six new biking stations.

The optimal configuration is reported in Figure 10-5. Dark gray markers with white outlines represent the locations of the already existing Hubway stations, while the yellow markers with black outline represent the optimal locations for the six new biking stations. Population blocks are shown, colored from white (zero population) to red (largest population per block area).

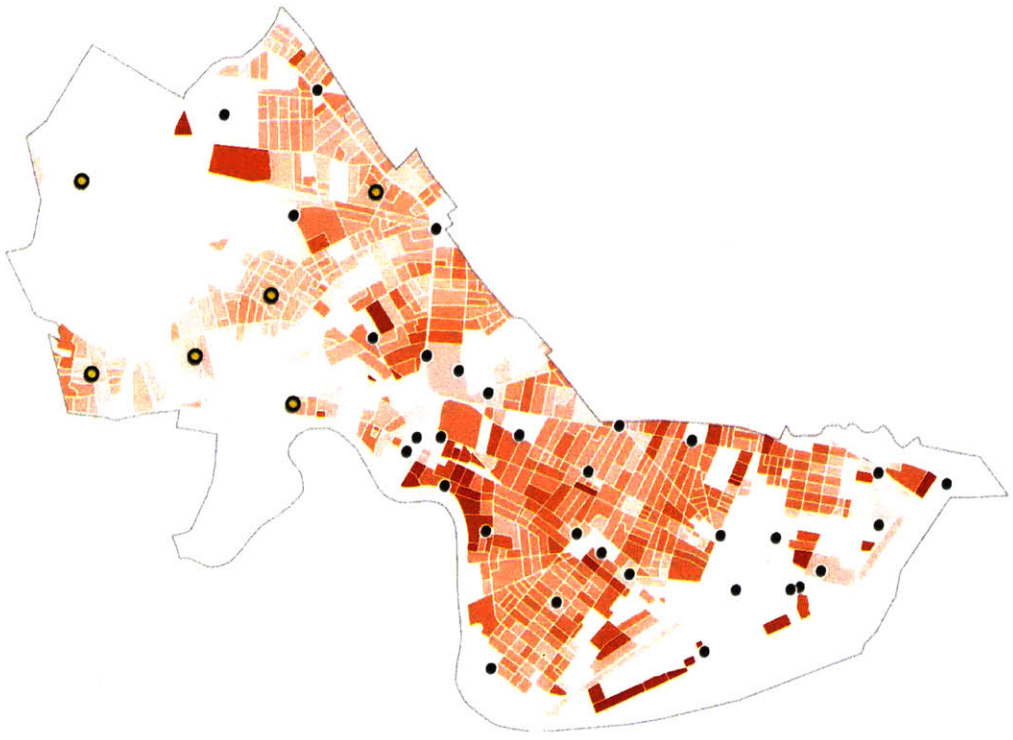


Figure 10-5: The optimal locations for adding six new biking stations (shown in yellow) for the city of Cambridge

10.6 Conclusion and Future Work

Recall that the problem introduced by this chapter was to evaluate where to place entities within cities. In particular, to find the most optimal places to build or set up entities given geospatial parameters and constraints. The goal was to create a tool that could be used to effectively address this problem. We modified the K-means clustering algorithm as an initial approach to solving the placements problem. We modeled existing instances of the entity as fixed means, while attempting to place new instances (regular means) in locations that optimize for various parameters. The K-means approach was effective and efficient, but yet could be extended for better performance. The following subsections discuss potential future steps that can be taken to improve this algorithm.

10.6.1 Using Travel Distances

Throughout the implementation of this algorithm, euclidean distances is computed between the means and the input points. While this serves as a good approximation for some cities, it fails to account for the geometry of roads within the city, and thereby solutions done by the algorithm might be suboptimal. Accounting for topology is possible using the functional distance concept introduced in earlier chapters. The map can be morphed by the real distance travelled by a selected method of transportation, and the locations can be selected on the morphed map and reprojected back to the original map.

10.6.2 Detecting for Validity of Place

One of the shortcomings of this algorithm is its inability to detect if an optimal looking solution is inherently bad if some of the locations suggested are placed on top of waterbodies, or in private perimeters. That is, the algorithm has no constraints on the locations it produces, as long as they are optimal with respect to the dataset. To make this algorithm more robust, the notion of output constraints needs to be added such that the selected locations are optimal only if the constraints are satisfied. For

instance, a constraint can be added to the bike station placement application such that the suggested locations are only limited to the side-paths.

Chapter 11

Conclusion

The goal of this thesis was to utilize web technologies and data analysis concepts by tools that enhance our understanding to the surrounding urban environment, and provide insights to the planning of interventions on the city level. By presenting large scale urban data using intuitive and interactive visualizations, the public can become more aware of the issues and challenges facing their cities. By providing planning tools, planners are better informed to implement more effective policies. Additionally, residents are able to contribute to the planning of their neighborhoods, and to initiate change within their communities.

The work in this thesis approached this goal in two ways. First, I presented a portfolio of web-based visualizations to present urban information effectively, and highlight patterns and trends. Each visualization shows and analyzes a different angle of the urban environment, from monitoring diagnoses and births to visualizing the changes in property values over time for a number of cities in the United States. Second, I presented a tool to suggest the most optimal locations for placing urban entities and establishment with respect to selected urban parameters. Using a modified k-means algorithm, the tool was successful in effectively and efficiently suggesting most optimal location as well as visualizing the results on a map, in parallel to the earlier visualization interfaces.

I believe that this project is exploring a small subset of a large space, and there is more work to be done in creating effective tools for studying and analyzing complex

urban processes using data. As cities grow lager and more urban data is collected, data visualizations will become an essential component in the urban planning process. Moreover, the more open cities become about the data they collect, the more data-driven development and progress will take place to create positive change within our environments.

Bibliography

[1] Gapminder: Unveiling the beauty of statistics for a fact based world view. `http://www.gapminder.org/`. Last accessed: 08/14/2014.

[2] Shawn Allen. Cartograms with d3 and topojson. `http://prag.ma/code/d3-cartogram/`. Last accessed: 08/16/2014.

[3] Michael Bostock. Hexagonal binning - a d3 plugin. `https://github.com/d3/d3-plugins/tree/master/hexbin`, 2013. A Github Repository.

[4] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011.

[5] James A Dougenik, Nicholas R Chrisman, and Duane R Niemeyer. An algorithm to construct continuous area cartogramsâĹŮ. 1985.

[6] The Guardian. A global guide to the first world war - interactive documentary. `http://www.theguardian.com/world/ng-interactive/2014/jul/23/a-global-guide-to-the-first-world-war-interactive-documentary`. Last accessed: 08/14/2014.

[7] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[8] Hubway. Hubway stations in cambridge. `https://www.thehubway.com/data/stations/bikeStations.xml`, 2014. Last accessed: 08/17/2014.

[9] Sepandar D Kamvar and Jonathan Harris. We feel fine and searching the emotional web. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 117–126. ACM, 2011.

[10] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[11] New York State Department of Health. Hospital Inpatient Discharges (SPARCS De-Identified) 2012. `https://health.data.ny.gov/d/3m9u-ws8e?category=Health&view_name=Hospital-Inpatient-Discharges-SPARCS-De-Identified`. Last accessed: 08/01/2014.

[12] New York State Department of Health. "sparcs overview", 2014. Last accessed: 08/11/2014.

[13] The Commonwealth of Massachusetts. Massgis data - datalayers from the 2010 u.s. census. http://www.mass.gov/anf/research-and-tech/it-serv-and-support/application-serv/office-of-geographic-information-massgis/datalayers/census2010.html, 2012. Last accessed: 08/17/2014.

[14] World Health Organization. 7 million premature deaths annually linked to air pollution. http://www.who.int/mediacentre/news/releases/2014/air-pollution/en/, 2014. Last accessed: 08/14/2014.

[15] David E. Persse, Craig B. Key, Richard N. Bradley, Charles C. Miller, and Atul Dhingra. Cardiac arrest survival as a function of ambulance deployment strategy in a large urban emergency medical services system. *Resuscitation*, 59(1):97 – 104, 2003.

[16] David Pimentel, Maria Tort, Linda D'Anna, Anne Krawic, Joshua Berger, Jessica Rossman, Fridah Mugo, Nancy Doon, Michael Shriberg, Erica Howard, et al. Ecology of increasing disease. *Bioscience*, pages 817–826, 1998.

[17] Edward Segel and Jeffrey Heer. Narrative visualization: Telling stories with data. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2010.

[18] The New York Times. Reshaping new york - interactive feature. http://www.nytimes.com/newsgraphics/2013/08/18/reshaping-new-york/. Last accessed: 08/14/2014.

[19] The New York Times. The voting blocs of new york city. http://www.nytimes.com/newsgraphics/2013/09/06/voting-blocs/. Last accessed: 08/14/2014.

[20] The New York Times. A survey of the flooding in n.y.c. after the hurricane - nytimes.com. http://www.nytimes.com/newsgraphics/2012/1120-sandy/survey-of-the-flooding-in-new-york-after-the-hurricane.html, 2012. Last accessed: 08/17/2014.