Unsupervised Learning for Submarket Modeling: A Proxy for Neighborhood Change

A Thesis Presented to the Faculty of Architecture, Planning, and Preservation
COLUMBIA UNIVERSITY

In Partial Fulfillment of the Requirements for the Degree
Master of Science in Urban Planning, Urban Analytics

by
Nicholas Hyeong Kunz
May 2019

Advisor: Prof. Lance Freeman
Reader: Kazuki Sakamoto

Abstract

This study focused on submarket modeling with unsupervised learning and geographic information system fundamentals to better understand urbanism at the neighborhood scale. A Spatially Constrained Weighted-Multivariate Hierarchical Clustering algorithm was trained to identify the optimal number of Multifamily Residential Commercial Real Estate submarkets in Manhattan, New York. The methodology explored Non-Negative Matrix Factorization for predicting the annual normalized values of every Multifamily Residential Commercial Real Estate property in Manhattan, which had been transacted from and including 2004 to 2018. Several extensive data transformations were applied prior to model fitting. A novel conditional random sampling technique was introduced to train and test set split sparse matrices for validating the prediction results. The study utilized a series of optimization techniques, including Leave One Out Cross Validation for estimating the optimal low rank matrix. The results from Non-Negative Matrix Factorization were compared with other imputation methods for sparse matrices, including Simon Funk's Singular Value Decomposition. Both the observed and imputed values were then clustered on a weighted basis with the Spatially Constrained Weighted-Multivariate Hierarchical Clustering algorithm, using five different Agglomerative Hierarchical Clustering linkage methods: Average Linkage, Median Linkage, Centroid Linkage, Complete Linkage, and Ward's Method. Ward's Method was found to be the superior linkage method for determining the optimal number of submarkets, when measured by the maximum absolute value difference between the mean intra-cluster similarity and the maximum inter-cluster dissimilarity. The clustering results indicated that the optimal number of Multifamily Residential Commercial Real Estate submarkets in Manhattan from 2004 to 2018 was 43. The final results were mapped by spatial joining to the intersecting land lot polygons with their respective submarket identifications. This study found that in several cases, there was a strong and obvious presence of multiple submarkets contained within a discrete neighborhood boundary. A speculative discussion was introduced regarding what that might mean for better understanding neighborhood change, possible policy applications, and the future of urbanism.

Keywords: neighborhood planning, urban real estate economics, matrix factorization, spatially constrained clustering, conditional random sampling sparse matrices

Acknowledgements

Thank you Prof. Freeman and Kaz for your guidance and motivation to purse quantitative methods of inquire throughout my time at Columbia. Also, thank you to my classmates for your friendship and support. Thank you Cheryl at the VA in Manhattan for your guidance in this transition. Thank you Dr. Badr for your correspondence and help implementing the Hierarchical Climate Regionalization algorithm, SmartyStreets for the educational license, and Google Cloud Engine for your flexibility with my virtual machine usage.

I would also like to thank my family, my fiancé, and my friends for your limitless support while pursuing this degree. In addition, I would like to dedicate this thesis to the following. In the time I was pursuing this degree, they made the ultimate sacrifice. I am humbled and honored to have had the privilege of serving with them. We truly stand on the shoulders of giants. Please remember:

SGT Jonathan Hunter, US Army, 1st Brigade Combat Team, 82nd Airborne Division
SPC Christopher Harris, US Army, 1st Brigade Combat Team, 82nd Airborne Division
SPC Devin Kuhn, US Army, 2nd Ranger Battalion, 75th Ranger Regiment
SGT Leandro Jasso, US Army, 2nd Ranger Battalion, 75th Ranger Regiment
SGT Cameron Meddock, US Army, 2nd Ranger Battalion, 75th Ranger Regiment

Rangers Lead the Way

Contents

List of Tables

List of Figures

List of Abbreviations

AHC: Agglomerative Hierarchical Clustering
CA: Cluster Analysis
CBSA: Core-Based Statistical Areas
Funk SVD: Simon Funk's Singular Value Decomposition
GIS: Geographic Information Systems
GSF: Gross Square Footage
FMR: Fair Market Rents
HiClimR: Hierarchical Climate Regionalization
HUD: Department of Housing & Urban Development
Lee: Lee & Seung's Multiplicative Update
LOOCV: Leave One Out Cross Validation
LSF: Land Square Footage
MFR-CRE: Multifamily Residential Commercial Real Estate
MICE: Multiple Imputation by Chained Equations
MKL: Mean Kullback-Leibler Divergence
MSE: Mean Squared Error
MVC: Multivariate Weighted Clustering
NNMF: Non-Negative Matrix Factorization
NTA: Neighborhood Tabulation Area
PCA: Principal Component Analysis
PLUTO: Primary Land Use Tax Lot Output
SAFMR: Small Area Fair Market Rent
SCD: Sequential Coordinate Wise Decent
SCWMHC: Spatially Constrained Weighted-Multivariate Hierarchical Clustering
SVD: Singular Value Decomposition
USPS: United States Postal Service
ZCTA: Zip Code Tabulation Areas

1 Introduction

Submarket Conceptualization, Purpose, and Inquire

1.1 Submarket Conceptualization

Location is perhaps the single most important and unique feature of real estate when compared to other asset classes. Although the location theory is a widely accepted one, there lacks a firm basis on how it is understood empirically. Of the existing empirical methods used to make sense of the phenomena, perhaps one of the most widely accepted urban economic ontologies is the notion of a submarket (Galster 1996; Whitehead 1999). Although there does exist some contention in this regard - mainly on whether or not submarkets theoretically exist, the inconsistency in which the term is sometimes used, and how to operationalize the concept - there is clear evidence that points to the need for a reliable analytical framework for making sense of the spatial complexity of large urban real estate economies, which are necessarily spatially dependent (Galster 1996; Whitehead 1999).

1.2 Purpose Statement

The purpose of this study was to attempt to reduce the spatial complexity of real estate market phenomena in a way that could be useful for planning & policy at the neighborhood scale. It sought to advance this effort through an approach that did not assume or impose the subject's geographic form or boundary *a priori*, as some previous studies have. This study sought to find ways to depart from relying on existing administrative areal unit aggregation (census tracts, zip codes, school districts etc.) for the basis of spatial analysis and geographic representation.

It was thought that the existing literature relied too heavily on spatial information, which was not theoretically consistent with the underlying phenomena. Therefore, it was believed that the findings of some studies suffered from being too distorted or misrepresented by the modifiable areal unit problem (MAUP) in order to be useful for planning & policy intervention at the neighborhood scale. In distancing itself from areal unit aggregation and the MAUP, this study instead relied on unsupervised learning and geographic information system fundamentals, as the basis for submarket modeling.

1.3 Thesis Statement

This study argued that submarket models were best approached as a spatially dependent classification problem, rather than an areal unit aggregation optimization one. It argued that classifying individual properties into spatially dependent clusters, were better submarket models for understanding neighborhood change than areal unit aggregation. It was also suggested that this approached generalized enough of the information for it to make sense. However, it was still detailed enough to maintain the integrity of the information at the spatial resolution of interest. Furthermore, that in order to effectively and responsibly proceed in implementing planning & policy interventions at the neighborhood scale, it is useful to consider them.

1.4 Research Questions

This study proceeded by asking the following questions: If we cannot rely on areal unit aggregation for describing real estate market geography, how do we know how many different submarkets there are in a given location and where exactly are they located? If we relied on machine intelligence to help us answer that question, how might we think about it? Would we interpret the findings differently than if they were determined with existing methods of inquire? Furthermore, how could we responsibly include this information revealed by machine intelligence to advance broader theories in urbanism to better serve communities at the neighborhood scale?

2 Background & Theory

Submarket Theory & Existing Methods of Inquire

2.1 Theory & Definition

2.1.1 Beyond the Law of One Price

The concept of a submarket can generally be thought of as a departure from the *"law of one price"* theory, which states that identical goods command identical prices (Baffes 2006). Sometimes the theory is more softly interpreted, as similar goods warranting similar prices (Handbury and Weinstein 2014). This idea works well in the context of markets such as commodities, securities, and currencies. However, within the context of real estate, it is not obvious that the theory is still valid (Galster1996; Whitehead 1999).

One of the reasons that the *"law of one price"* theory comes into question with regard to real estate, is the structural heterogeneity of the underlying assets (Galster1996; Whitehead 1999). In the architecture and planning lexicon, this is sometimes described as different building typologies (Montalbán Pozas and Neila González 2018). Meaning, the comparability of real estate assets is difficult for determining their market prices because land and buildings are often times poor substitutes of one another. This continues to remains a long standing issue (Galster1996; Whitehead 1999).

This is further complicated by the spatial dependency of real estate. In contrast to other markets such as commodities, securities, and currencies, real estate is by definition permanently fixated to a specific geography. This idea is well understood and even directly captured in the Italian translation for real estate as *"Immobiliare"*. In common parlance, this is often referred to as the *"3L's"* - location, location, location.

A new theory began to appear in the literature in the mid 20th century, which challenged the *"law of one price"* (Fisher and Fisher 1954). It evolved to suggest that there is not a single unitary real estate market, but perhaps several "market-subdivisions" or what is now referred to in the literature as "submarkets" (Goodman 1981). The theory recognized that the structural heterogeneity and spatial dependency of real estate prevented it from fitting unitary market theory. It was thought that submarkets were a more appropriate theoretical basis for understanding real estate economics than was the unitary market theory (Goodman 1981; Schnare and Struyk 1976).

2.1.2 Submarkets Defined

What these researchers were describing was not just a more nuanced view of real estate market segmentation by building typology, but in fact introducing the idea that perhaps there are multiple different real estate markets given a location. Bourassa et al. (1999) had succinctly summarized this notion in his definition of submarkets as:

> *"...a set of dwellings that are reasonably close substitutes for one another, but relatively poor substitutes for dwellings in other submarkets"* (Bourassa et al. 1999).

By the 1990's a substantial body of empirical research had been developed, which supported this definition and the submarket theory over the unitary market theory (Adair1996; Galster 1996; Goodman and Thibodeau 1998; Bourassa et al. 1999). Several studies demonstrated that by controlling for the spatial dependency of real estate markets through submarket boundaries, hedonic models increased predictive performance when estimating prices (Adair, Berry, and McGreal 1996; Goodman and Thibodeau 2003, 2007). These studies were important because they provided the empirical evidence for earlier arguments supporting submarket theory.

2.2 Existing Studies

2.2.1 Areal Unit Aggregation

Perhaps one of the most common themes underlying the existing empirical research on submarkets was their reliance on assumed areal units of geography for submarket identification. One such example, was a series of influential studies conducted by Goodman and Thibodeau (1998) Goodman and Thibodeau (2003). The studies identified submarkets on the basis of school districts in Dallas, Texas. Another such study that relied on assumed areal units was conducted by Watkins (2001). The study analyzed the housing market in Glasgow, Scotland using postal code aggregation as the geographic basis of analysis. Although there are a number of studies that engage in areal unit aggregation *a priori*, these methods have seem to fallen out of favor in the literature and largely been supplanted by other more nuanced ways of identifying them.

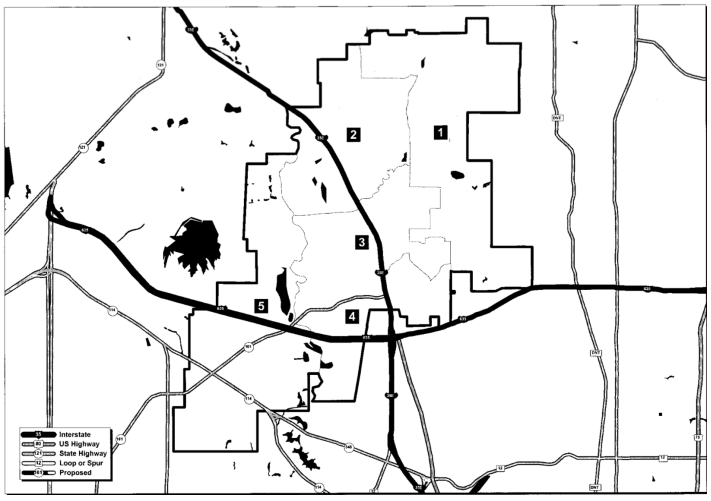

Figure 1: Dallas Carrollton–Farmers Branch School District Submarkets (Goodman et al. 1998)

2.2.2 Nested Classification

Underlying some of the submarket research was the idea that submarkets might be thought of as spatially nesting classifications. A study conducted by Adair, Berry, and McGreal (1996) investigated housing market areas in Belfast, Northern Ireland based on three geographic scales ranging from inner, middle, and outer (the inner nested within middle, and the middle nested within outer). Although one of the motivation of the study was to identify the optimal geographic scale to analyze the Belfast housing market, an ancillary result was that it opened the idea that submarkets might be spatially nested (Adair, Berry, and McGreal 1996).

The notion of spatially nested submarkets was later explicitly investigated in the previously mentioned series of studies (Goodman and Thibodeau 1998, 2003, 2007). They conducted a number of studies on the theoretical basis of spatial nesting by using hierarchical methods to identify submarkets on the basis of school districts in Dallas, Texas. These series of studies strengthened the notion that submarkets might be bettered described as a telescoping spatial compartmentalization that ranges in variability relative to geographic scale (Goodman and Thibodeau 1998, 2003, 2007).

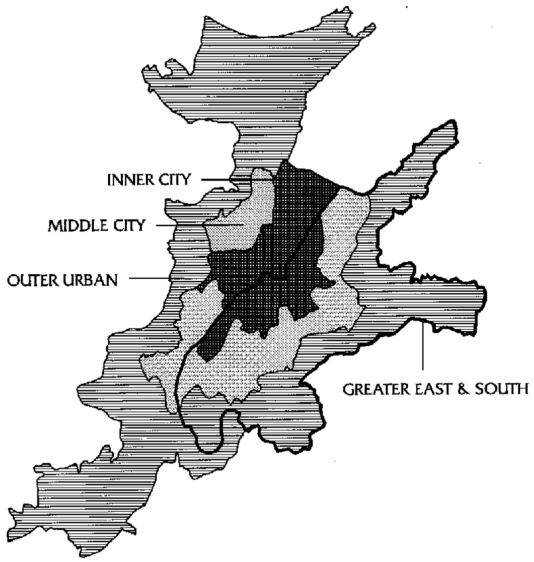

Figure 2: Spatial Sub-Divisions Within The Belfast Urban Area (Adair et al. 1996)

2.2.3 Statistical Clustering

In advancing the technical rigor of submarket research, some studies focused on statistical methods of submarket identification to include those commonly found in the machine learning literature (Gareth et al. 2000). One such study conducted by Wu and Sharma (2012) used Principal Component Analysis (PCA) and Cluster Analysis (CA) to identify submarkets in Milwaukee, Wisconsin. The study introduced a spatially constrained PCA and CA methodology for submarket identification and compared them with a spatially unconstrained PCA and CA model, as well as school districts and alderman districts (city council districts). The study found that the spatially constrained PCA and CA method was a superior method in identifying submarkets when validated by substitutability, spatial contiguity and similarity (Wu and Sharma 2012).

An earlier study, which focused on statistical methods, was conducted by Bourassa et al. (1999). The study used PCA and K-Means clustering to identify submarkets in both Sydney and Melbourne, Australia. It was found that by clustering individual properties by K-Means, the model identified submarkets with only marginally better improvement than assumed submarket boundaries. Although at closer examination of the study, still casually assumed administrative areal unit boundaries as the basis of spatial analysis (Bourassa et al. 1999).

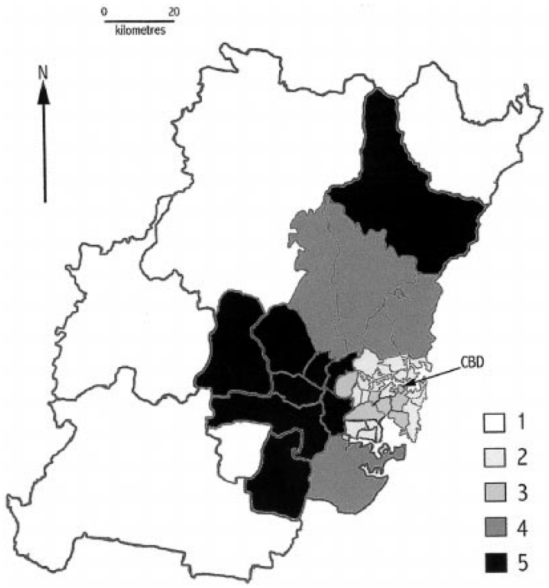

Figure 3: Sydney LGA K-Means Submarkets (Bourassa et al. 1999)

2.2.4 Expert Opinion

Other studies departed from advanced quantitative methods of research. Rather, they explored the possibility that expert opinion might be just as effective in identifying submarkets as their statistically driven counterparts. One such study conducted by Keskin and Watkins (2017) focused on comparing submarket methodologies in Istanbul, Turkey. The study compared those identified by real estate agents and those identified by PCA. They demonstrated that expert opinion, generally performed at least as well as PCA at identifying submarkets, when measured by the predictive performance of their price estimations (Keskin and Watkins 2017).

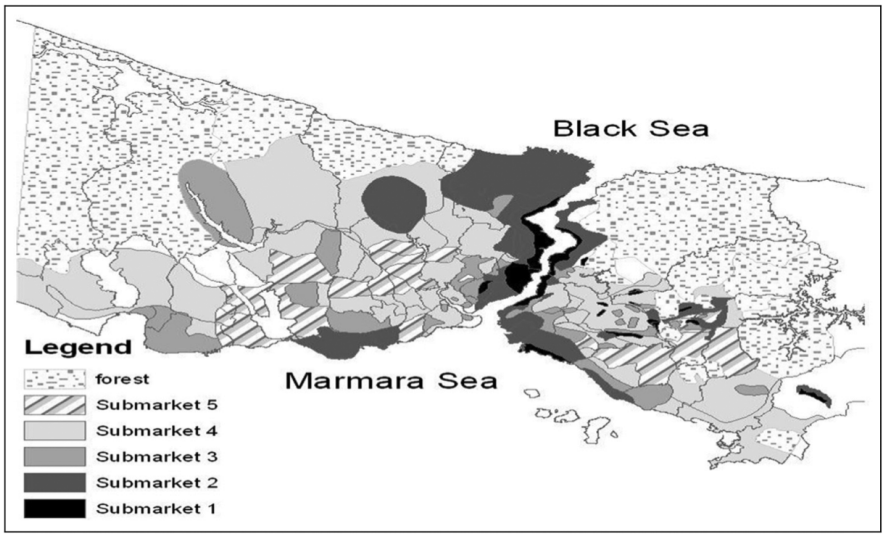

Figure 4: Submarkets in Istanbul (Keskin et al. 2017)

2.3 Need for Investigation

Although the theory and methods of existing submarket studies have advanced in depth and rigor, many still rely on some degree of areal unit aggregation for explaining real estate market geography. In addition, they generally require this information to analyze their subject *a priori*. Meaning, they typically do not attempt to model their geography explicitly, but rather impose it. Furthermore, the results of the maps produced in the existing literature are usually too spatially generalized to be useful at the neighborhood scale. Of the existing methods for submarket modeling that might be useful at the the neighborhood scale, few if any have accounted for the way in which real estate geography is manifested at the moment of market exchange (individual property boundaries). Additionally, it appears that no previous study has focused specifically on Multifamily Residential Commercial Real Estate (MFR-CRE), analyzed an extremely dense study area, or discussed the subject within the context of urbanism. Because of this void in the literature, this study was conducted.

3 Study Area & Data

Real Estate Sales & Land Lot Polygons in Manhattan, New York

3.1 Study Area

3.1.1 Geography: Manhattan

The study area of this research was the borough of Manhattan in New York, New York. Manhattan was selected as the study area because of the heterogeneity and high density of its building typology. The dissimilarity of the structural characteristics underlying Manhattan's real estate was of interest, as it posed additional challenges to asset and price comparability that this study sought to address. This is in contrast to suburban single-family detached residential markets, where properties can generally be thought of as having a higher degree of typological or structural similarity. Therefore, are generally considered easier to assess market prices by comparability.

3.1.2 Market Segment: Multifamily Residential Commercial Real Estate

The scope of this research was specifically focused on Multifamily Residential Commercial Real Estate (MFR-CRE). MFR-CRE was selected as the market segment for this study for three reasons. First, it is abundant given the study area. Second, it necessarily contains leasehold tenants by the definition of its broader categorical association within commercial real estate. Leasehold tenants were of interest, as it is thought that they are generally more sensitive to neighborhood change at low income levels than their freehold counterparts. This was briefly discussed near the end of the study in *Chapter 6 - Implications*. Third, as previously mentioned, the varying degree of building typology or structural characteristics of MFR-CRE poses additional challenges to asset comparability. When operationalized, the issue presented itself as a combined multivariate classification and spatial regionalization problem.

3.1.3 Additional Considerations: Land Rights

Also, this study only focused on MFR-CRE sales containing land rights. Transactions containing land rights were important because their geography could be explicitly modeled using their land lot parcel boundaries. The land lot parcel boundaries were used as the basis of spatial representation, as their areal units were considered the most appropriate basis of analysis at the highest resolution of geography. This is in contrast to being presented as merely dots on a map or a theoretically dislocated representation of submarket phenomena, such as administrative areal unit aggregation.

3.2 Data

There were four data sources used in this study. The first and primary data source was the Rolling Real Estate Sales data provided by the New York City Department of Finance. The second data source was the Primary Land Use Tax Lot Output (PLUTO) data provided by the New York City Department of City Planning. The third data source was the Zip Code Tabulation Areas (ZCTA) data retrieved from the City University of New York, originally published by the United States Census Bureau. The fourth data source was the Neighborhood Tabulation Areas (NTA) provided by the New York City Department of City Planning. All data sources used in this study were open for public distribution and retrieved on a rolling basis from October 2018 to December 2018. All basemaps were open source generated with Leaflet, OpenStreetMap, and CartoDB. The R implementation of all data loading procedures described in this section were made available for reproduction in *Appendix A - Data*.

3.2.1 Real Estate Sales

The Rolling Real Estate Sales data originally contained 349,777 observations and 21 variables. The observations were the number of total real estate transactions in Manhattan, New York from 2004 to 2018. Each observation represented a single recorded real estate sale. The temporal resolution of the data was detailed to the a day, but was later aggregated per annualized basis (NYC Department of Finance 2018). After fully pre-processing the data, but before the matrix transformation, the data contained 8,659 observations and 18 variables. The details regarding the data handling procedures are detailed later in this chapter.

Dimensions

```
## [1] 349777     21
```

List of Variables

```
##  [1] "BOROUGH"                       "NEIGHBORHOOD"
##  [3] "BUILDING CLASS CATEGORY"       "TAX CLASS AT PRESENT"
##  [5] "BLOCK"                         "LOT"
##  [7] "EASEMENT"                      "BUILDING CLASS AT PRESENT"
##  [9] "ADDRESS"                       "APARTMENT NUMBER"
## [11] "ZIP CODE"                      "RESIDENTIAL UNITS"
## [13] "COMMERCIAL UNITS"              "TOTAL UNITS"
## [15] "LAND SQUARE FEET"              "GROSS SQUARE FEET"
## [17] "YEAR BUILT"                    "TAX CLASS AT TIME OF SALE"
## [19] "BUILDING CLASS AT TIME OF SALE" "SALE PRICE"
## [21] "SALE DATE"
```

3.2.2 Primary Land Use Tax Lot Output

The Primary Land Use Tax Lot Output (PLUTO) data originally contained 42,556 observations and 90 variables, plus the polygon information. The observations were the total number of land lots in Manhattan, New York and 90 of their attributes. Each observation represented an individual land lot polygon (NYC Department of City Planning 2018b). After fully pre-processing the PLUTO data, the data contained 42,556 observations and 2 variables, plus the polygon information. The Coordinate Reference System (CRS) was projected using the World Geodetic System 1984 Web (WGS84). The details regarding the data handling procedures of this data are described later in this chapter.

Dimensions

```
## [1] 42556    90
```

List of Variables

```
##  [1] "Borough"    "Block"      "Lot"        "CD"         "CT2010"
##  [6] "CB2010"     "SchoolDist" "Council"    "ZipCode"    "FireComp"
## [11] "PolicePrct" "HealthCent" "HealthArea" "SanitBoro"  "SanitDistr"
## [16] "SanitSub"   "Address"    "ZoneDist1"  "ZoneDist2"  "ZoneDist3"
## [21] "ZoneDist4"  "Overlay1"   "Overlay2"   "SPDist1"    "SPDist2"
## [26] "SPDist3"    "LtdHeight"  "SplitZone"  "BldgClass"  "LandUse"
## [31] "Easements"  "OwnerType"  "OwnerName"  "LotArea"    "BldgArea"
## [36] "ComArea"    "ResArea"    "OfficeArea" "RetailArea" "GarageArea"
## [41] "StrgeArea"  "FactryArea" "OtherArea"  "AreaSource" "NumBldgs"
## [46] "NumFloors"  "UnitsRes"   "UnitsTotal" "LotFront"   "LotDepth"
## [51] "BldgFront"  "BldgDepth"  "Ext"        "ProxCode"   "IrrLotCode"
## [56] "LotType"    "BsmtCode"   "AssessLand" "AssessTot"  "ExemptLand"
## [61] "ExemptTot"  "YearBuilt"  "YearAlter1" "YearAlter2" "HistDist"
## [66] "Landmark"   "BuiltFAR"   "ResidFAR"   "CommFAR"    "FacilFAR"
## [71] "BoroCode"   "BBL"        "CondoNo"    "Tract2010"  "XCoord"
## [76] "YCoord"     "ZoneMap"    "ZMCode"     "Sanborn"    "TaxMap"
## [81] "EDesigNum"  "APPBBL"     "APPDate"    "PLUTOMapID" "FIRM07_FLA"
## [86] "PFIRM15_FL" "Version"    "MAPPLUTO_F" "SHAPE_area" "SHAPE_len"
```

Primary Land Use Tax Lot Output Map

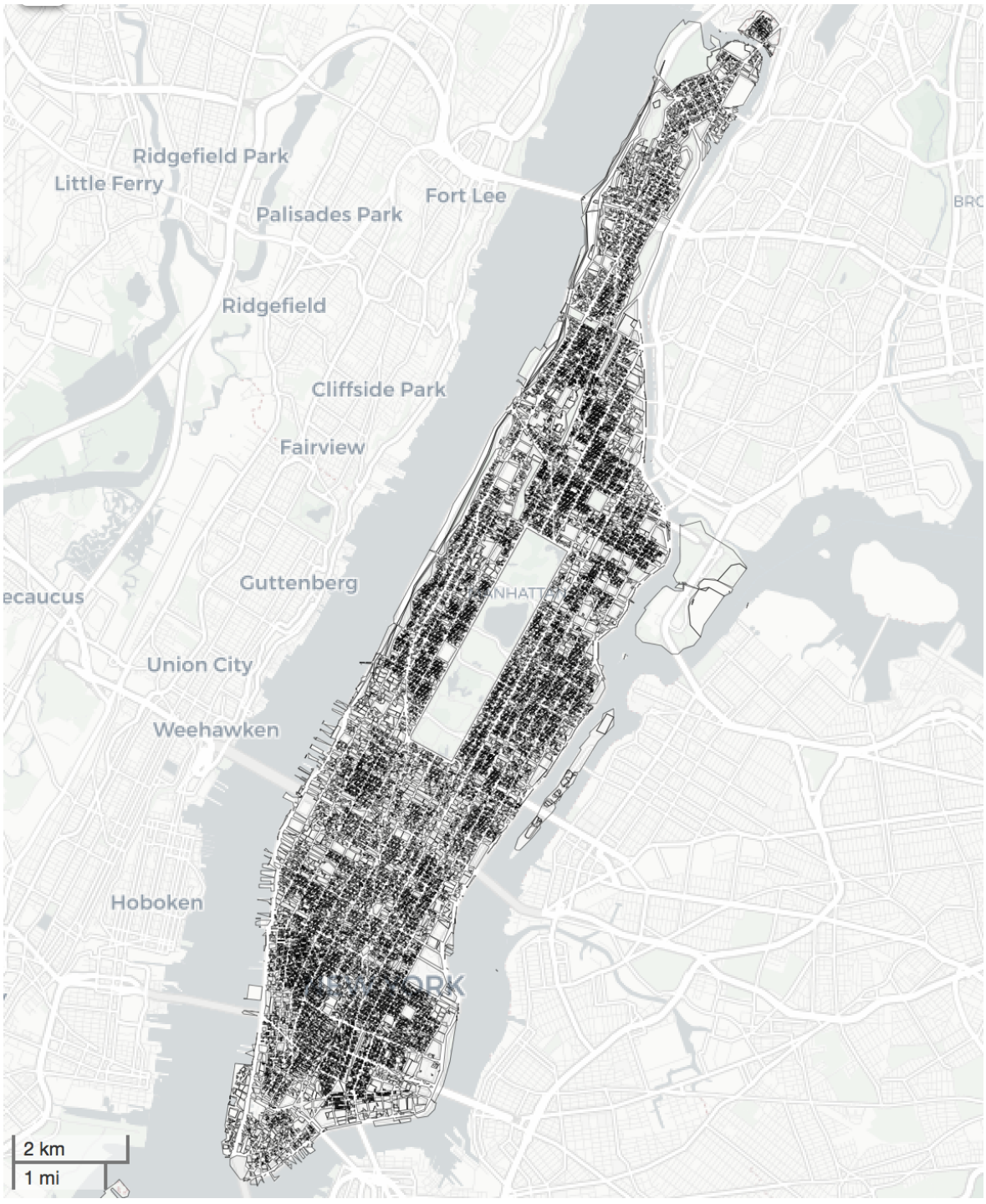

Figure 5: Primary Land Use Tax Lot Output Map

3.2.3 Zip Code Tabulation Areas

The Zip Code Tabulation Areas (ZCTA) data contained 69 observations and 3 variables, plus the polygon information. The observations were the total number of ZCTA's in Manhattan, New York (Baruch College City University of New York 2016). No pre-processing or transformations were conducted. The Coordinate Reference System (CRS) was projected using the World Geodetic System 1984 Web (WGS84), consistent with the PLUTO data. The ZCTA data was used for geographic context and to serve as the analytical basis of comparison to the main results of this study. Specifically, the ZCTA data was used to determine the geography of the Small Area Fair Market Rent (SAFMR) areas determined by the United States Department of Housing & Urban Development (HUD) (HUD Office of Policy Development and Research 2019b). The details regarding the use of this data is described later in this study in *Chapter 6 - Implications*.

Dimensions

```
## [1] 69  3
```

List of Variables

```
## [1] "zcta"  "bcode" "note"
```

3.2.4 Neighborhood Tabulation Areas

The Neighborhood Tabulation Areas (NTA) data contained 29 observations and 7 variables, plus the polygon information. The observations were the number of neighborhoods in Manhattan recognized by the New York City Department of City Planning. Each observation represented a single neighborhood (NYC Department of City Planning 2018b). No pre-processing or transformations were conducted. The Coordinate Reference System (CRS) was projected using the World Geodetic System 1984 Web (WGS84), consistent with the PLUTO data. The data was used for providing geographical context and to serve as the basis of comparison for the results of this study.

Dimensions

```
## [1] 29  7
```

List of Variables

```
## [1] "boro_code"  "boro_name"  "county_fip" "ntacode"    "ntaname"
## [6] "shape_area" "shape_leng"
```

Zip Code Tabulation Areas Map

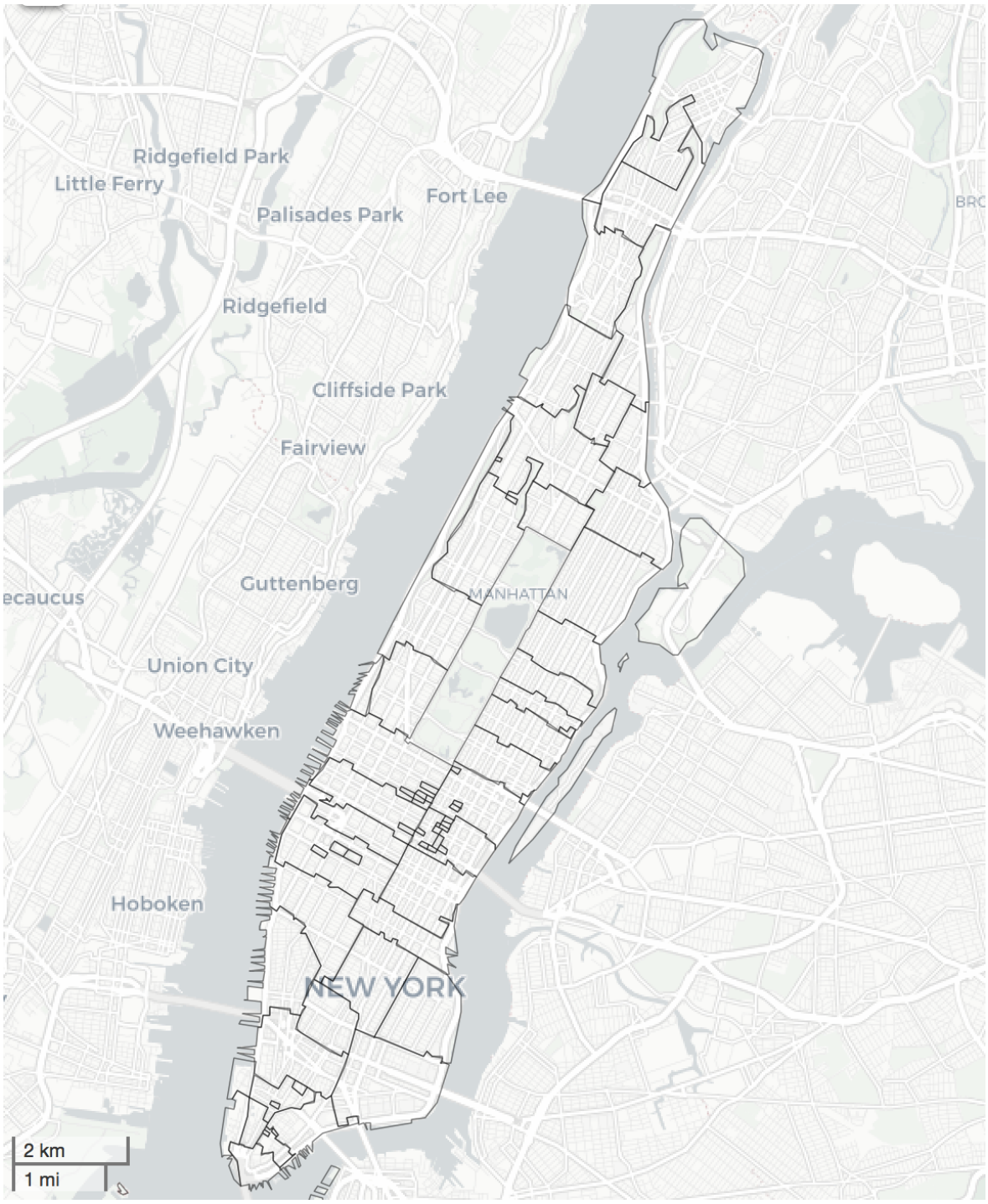

Figure 6: Zip Code Tabulation Areas Map

Neighborhood Tabulation Areas Map

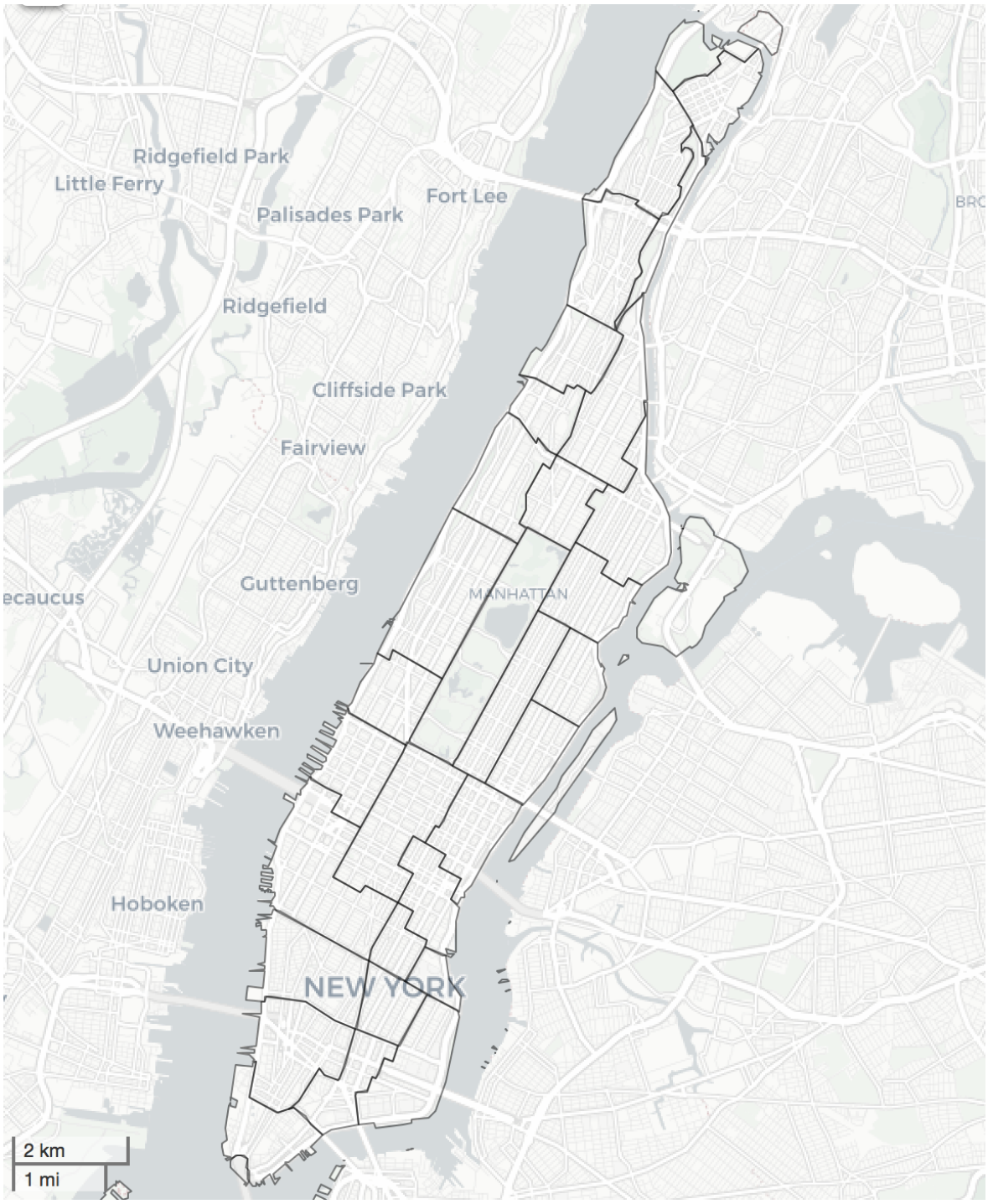

Figure 7: Neighborhood Tabulation Areas Map

3.3 Standard Data Handling Procedures

Ten data handling procedures were conducted in this study. They were categorized into two sections. The first section was *Standard Data Handling Procedures*, which detailed the data handling procedures ranging from one to six. The second section was *Advanced Data Handling Procedures*, which detailed the data handling procedures from seven to ten. Each procedure was carefully selected to ensure the minimum possible information loss. Omitting variables or observations for sake of practical convenience where missing values were discovered was not considered. The R implementation of all data handling procedures described in this section were made available for reproduction in *Appendix B - Standard Data Handling Procedures* and in *Appendix C - Advanced Data Handling Procedures*.

3.3.1 Data Cleaning

The first of the *Standard Data Handling Procedures* was a general cleaning operation on the Real Estate Sales data. All duplicate observations were removed, the variable names (headers) were corrected and operationalized. Any special characters (currency denominations, punctuation, etc.) and number ordinals (1 *'st'*, 2 *'nd'*, 3 *'rd'*) contained within character strings were removed. Any extra white space (within, leading, or training) within character strings was also removed.

3.3.2 Feature Selection

The second procedure was variable selection. The variables in the Real Estate Sales data were removed that were not applicable to this study. The remaining variables were: Building Classification Category, Address, Zip Code, Residential Unit Count, Commercial Unit Count, Total Unit Count, Gross Square Footage, Land Square Footage, Sales Price, and Sales Date. The same procedure was repeated with the attribute information in the Land Lot Polygon PLUTO data. The remaining variables in the Land Lot Polygon PLUTO data were: Address and Zip Code, plus all polygon information.

3.3.3 Data Type Correction

The third procedure corrected all the data types contained Real Estate Sales data. Building Classification Category, Address, and Zip Code were all converted to character strings. Residential Unit Count, Commercial Unit Count, Total Unit Count, Gross Square Footage, Land Square Footage, and Sales Price were all converted to numeric data. The Sales Date variable was aggregated from days to years, and converted to date data.

3.3.4 Recalculation

The fourth procedure corrected erroneous data entry errors in the total number of units per MFR-CRE property (Residential Units plus Commercial Units) in the Real Estate Sales data. All missing values in the Commercial Units variable were assumed to be missing because there were no commercial units contained in the property and were imputed with a zero entry. The Total Unit count was then recalculated by summing the Residential Units variable with the Commercial Units variable. In a manual inspection, it was observed that there were no other variables that contributed to the Total Unit count. Also, sales that included properties containing zero values in the GSF variable were assumed to be data entry errors and converted to missing values to be statistically imputed later.

3.3.5 Subsetting

The fifth procedure subsetted the data to only include MFR-CRE sales containing land rights in the Real Estate Sales data. The property types that were included were limited to Walk-Up Apartments, Elevator-Apartments, Condo Rentals, and 4-10 Residential Unit Rentals (only MFR-CRE). MFR-CRE sales, which contained Land Square Footage (LSF), not one or zero (contained land rights), and also had more than 4 housing units (multifamily residential) were retained. All property sales below $1,000 were truncated to avoid including obvious data input errors and non-market transactions (not at arm's length). Where to truncate price or even if to price truncate is debatable and a limitation of this study. However, truncating observations to avoid obvious noise was thought to be a prudent way to proceed with precedent (Wu and Sharma 2012).

3.3.6 Address Standardization & Geocoding

The sixth procedure cleaned the addresses utilizing the United States Postal Service (USPS) standard. This step was conducted externally with the educational license provided by SmartyStreets. The list of USPS addresses in the Real Estate Sales data were table merged with the USPS addresses cleaned by the same standard in the PLUTO data. The land lot polygon centroids were then calculated from the PLUTO data. The geocoordinantes from land lot polygon centroids were then table merged with the Real Estate Sales data by common USPS address. The geocoordinates provided by SmartyStreets were then used secondarily only when the geocoordinates from the land lot polygon centroids could not be merged with the Real Estate Sales data by shared USPS address.

List of Address Standardization Results

```
##  [1] "1592 2nd Ave"                "38 W 75th St"
##  [3] "2051 2nd Ave"                "11 Saint Marks Pl"
##  [5] "42 W 76th St"                "601 10th Ave"
##  [7] "52 E Broadway"               "52 Hamilton Ter"
##  [9] "117 Henry St"                "530 W 178th St"
## [11] "1431a York Avenue"           "517 W 180th St"
## [13] "354 W 39th St"               "223 E 2nd St"
## [15] "235 E 87th St"               "7 Avenue A"
## [17] "20 Greene St"                "242 E 124th St"
## [19] "420 E 10th St"               "St Nicholas Avenue"
## [21] "38 E 38th St"                "42 W 48th St"
## [23] "427 W 47th St"               "697 10th Ave"
## [25] "149 W 93rd St"               "218 W 14th St"
## [27] "112 W 34th St"               "1190 Lexington Ave"
## [29] "247 E 48th St"               "1484 Saint Nicholas Ave"
## [31] "165 E 103rd St"              "522 E 11th St"
## [33] "444 W 19th St"               "108 W 69th St"
## [35] "62 Hamilton Ter"             "2455 Frederick Douglass Blvd"
## [37] "156 W 75th St"               "1637 1st Ave"
## [39] "1264 Amsterdam Ave"          "23 Ludlow St"
```

Address Standardization & Geocoding Map

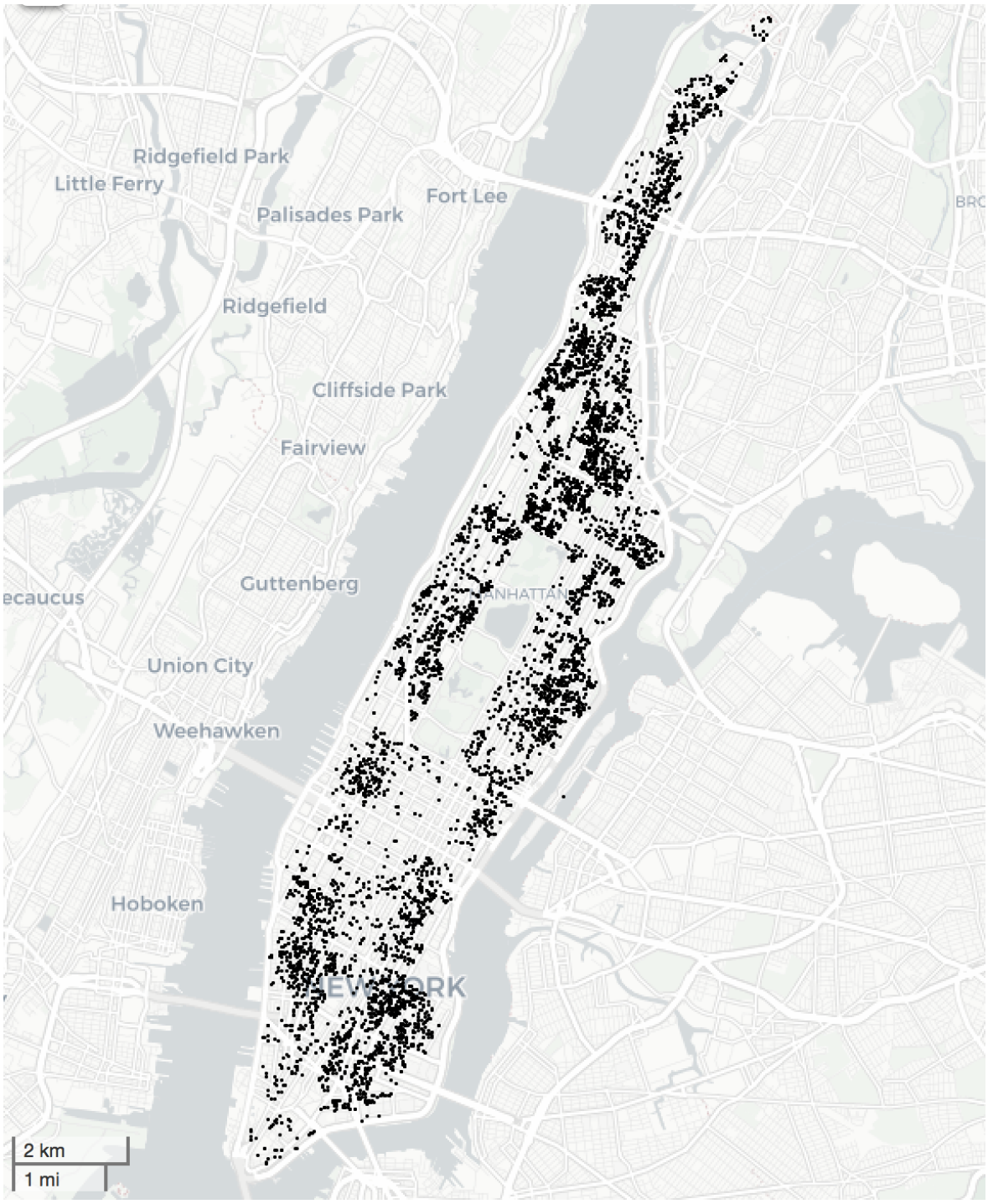

Figure 8: Real Estate Sales Geocoding Results Map

3.3.7 Data Exploration

Frequency of Transactions

Exhibited here were the 100 most frequently transacted MFR-CRE properties in Manhattan from 2004 to 2018. Notice the nominally small number of properties transacted over 5 times throughout the course of 15 years. This seems to suggest that MFR-CRE sales were generally not common. The average number of times a MFR-CRE property was transacted in Manhattan from 2004 to 2018 was roughly 1.5. Meaning, the occurrence of a MFR-CRE transaction is significant and should not be overlooked.

100 Most Transacted Properties Plot

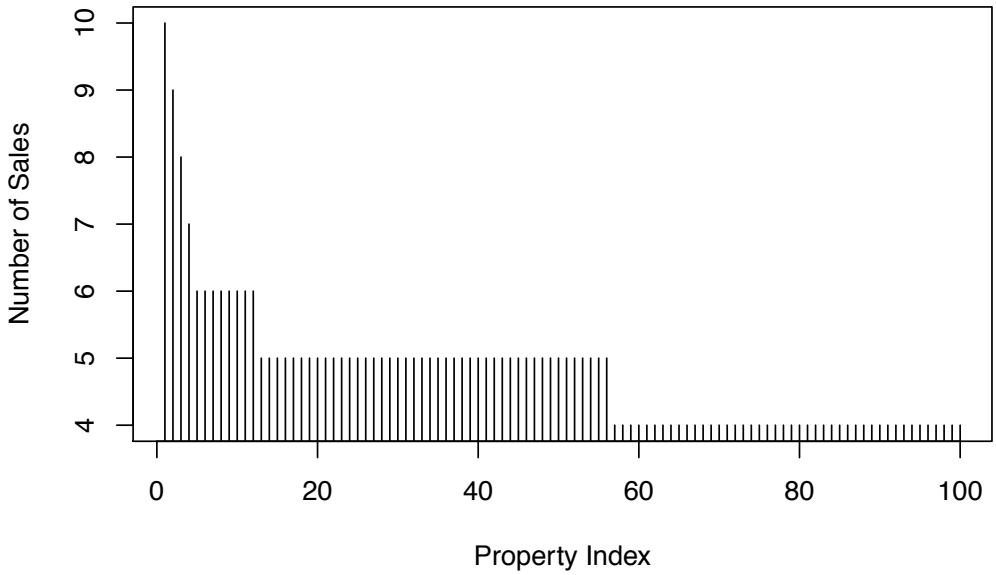

Figure 9: 100 Most Transacted Properties Plot

Average Number of Transactions Results

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.445   2.000  10.000
```

3.4 Advanced Data Handling Procedures

3.4.1 Multiple Imputation by Chained Equations

The seventh procedure overall data handling procedure and first of the *Advanced Data Handling Procedures*, imputed the 21 missing values contained in the Gross Square Footage (GSF) variable in the Real Estate Sales data. Rather than omitting the observations, the missing values were imputed using Multiple Imputation by Chained Equations (MICE). More information regarding MICE can be found in *"MICE: Multivariate Imputation by Chained Equations in R"* by Buuren and Groothuis-Oudshoorn (2011).

There were limitations to this method, as no rigorous tests were involved to validate the predictions. However, the 10 imputation iterations exhibited generally similar distributions as the observed values. It is also important to mention that 21 imputed observations were not believed to be enough to distort the integrity of the data. However, it was suspected that removing them might cause more damage than predicting them.

Multiple Imputation by Chained Equations Density Plot

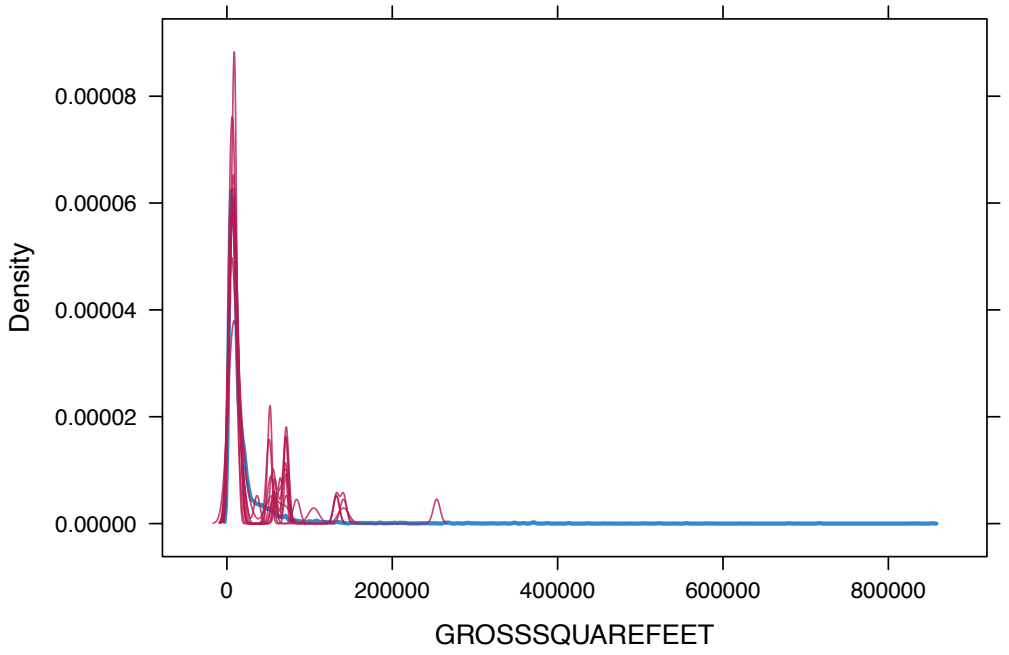

Figure 10: Multiple Imputation Density Plot

3.4.2 Feature Engineering

The eighth procedure feature engineered four variables used in model specification in the Real Estate Sales data. These four feature engineered variables were specified to capture the relationship between the observed price and the underlying structural characteristics of each MFR-CRE property. The model could then be specified for both price similarity, as well as similarity in the interaction between price and asset comparability or building typological substitutability. The four feature engineered variables were:

List of Feature Engineered Variables

- *Sale Price Ratio of Residential Units / Number of Residential Units*
- *Sale Price / Total Number of Units (Including Commercial Units)*
- *Sale Price / Gross Square Footage*
- *Sale Price / Land Square Footage*

List of Feature Engineered Variables Result

```
##  [1] "ADDRESS.USPS"                  "RESIDENTIALUNITS"
##  [3] "TOTALUNITS"                    "GROSSSQUAREFEET"
##  [5] "LANDSQUAREFEET"                "SALEPRICE"
##  [7] "SALEYEAR"                      "LAT"
##  [9] "LON"                           "SALEPRICE.PER.RES.UNITS.RATIO"
## [11] "SALEPRICE.PER.TOTALUNITS"      "SALEPRICE.PER.GROSSSQUAREFEET"
## [13] "SALEPRICE.PER.LANDSQUAREFEET"
```

3.4.3 Normalization

The ninth procedure normalized the five variables specified in the model. Normalization was conducted on a non-negative scale from *0 to 100*. This was step was taken in an effort to preserve the non-negative quality of the observed values for interpretability, as gross asset values of real estate can theoretically only be positive. Also, positive normalization was conducted to ensure that the matrix factorization conducted later would guarantee non-negative predictions. In addition, it was believed that the 0 to 100 scale was easy to interpret.

3.4.4 Matrix Transformation

The tenth procedure transposed each of the five variables into separate sparse matrices ($A = n \text{ x } m$). The matrices contained n properties and their geolocations (in a separate corresponding list) and m years. That is to say, the rows n contained the index of MFR-CRE property sales, the columns m contained years, and the values A_{ij} contained observed sale prices. The sparsity of the matrix was roughly 91% missing. There were five variables selected for model specification; four of which were feature engineered variables covered in the previous steps, *Feature Engineering* and *Normalization*. The five variables were:

List of Matrice Variables

- X_1 = *Normalized Sale Price*
- X_2 = *Normalized Sale Price Ratio of Residential Units / Number of Residential Units*
- X_3 = *Normalized Sale Price / Total Number of Units (Including Commercial Units)*
- X_4 = *Normalized Sale Price / Gross Square Footage*
- X_5 = *Normalized Sale Price / Land Square Footage*

Matrix Missingness Plot: 91% Missing

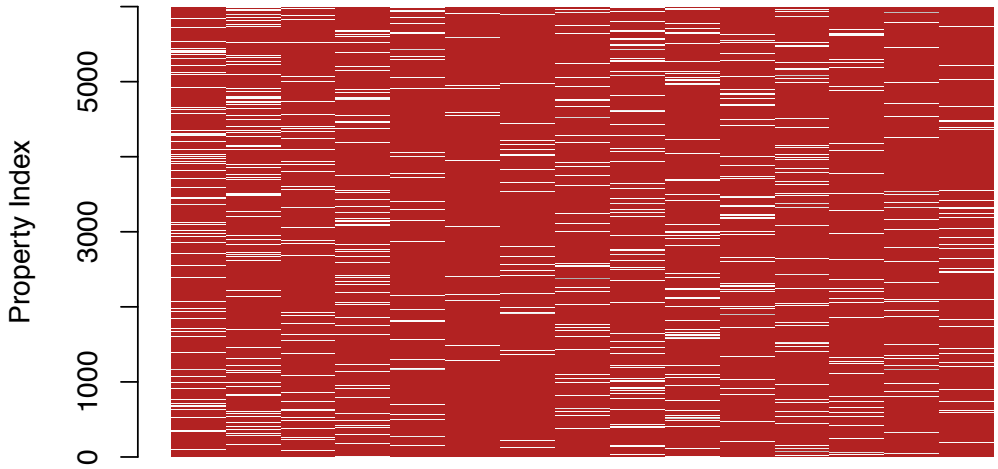

Figure 11: Matrix Missingness Plot

3.5 Remarks on Data

As a concluding remark regarding the data and data handling procedures in this study, data preservation was taken seriously. The data handling procedures were carefully designed to minimize information loss. In fact, much of the methodology itself was designed to extend the limits on urban data handling in this regard. The R implementation of all data handling procedures described in this chapter were made available for reproduction in *Appendix B - Data Handling Procedures* and in *Appendix C - Advanced Data Handling Procedures*.

3.6 Descriptive Statistics Summary

Table 1: Descriptive Statistics Summary of Original Variables

	Residential Units	Total Units	GSF	LSF	Price
nbr.val	8659.00	8659.00	8638.00	8659.00	8659.00
nbr.null	0.00	0.00	0.00	0.00	0.00
nbr.na	0.00	0.00	21.00	0.00	0.00
min	4.00	4.00	1614.00	469.00	1000.00
max	914.00	915.00	853569.00	248323.00	620000000.00
range	910.00	911.00	851955.00	247854.00	619999000.00
sum	221901.00	230045.00	195208940.00	41469854.00	77721001793.00
median	14.00	15.00	9625.00	2517.00	4075000.00
mean	25.63	26.57	22598.86	4789.22	8975747.98
SE.mean	0.53	0.53	554.31	96.33	253221.60
CI.mean.0.95	1.03	1.04	1086.57	188.83	496374.60
var	2399.26	2450.75	2654069830.60	80348403.43	555225274124526.62
std.dev	48.98	49.51	51517.67	8963.73	23563218.67
coef.var	1.91	1.86	2.28	1.87	2.63

Table 2: Descriptive Statistics Summary of Feature Engineered Variables

	Price Ratio / Residental Units	Price / Total Units	Price / GSF	Price / LSF
median	239071.00	252941.00	381.00	1308.00
mean	391720.49	422125.74	548.09	2019.93
SE.mean	7845.28	8383.49	9.22	32.14
CI.mean.0.95	15378.61	16433.64	18.08	63.01
var	532947557672.17	608580341917.01	736827.72	8946310.10
std.dev	730032.57	780115.60	858.39	2991.04
coef.var	1.86	1.85	1.57	1.48

Table 3: Descriptive Statistics Summary of Normalized Feature Engineered Variables

	Price Ratio / Residential Units	Price / Total Units	Price / GSF	Price / LSF
median	0.52	0.55	0.83	1.36
mean	0.85	0.91	1.19	2.10
SE.mean	0.02	0.02	0.02	0.03
CI.mean.0.95	0.03	0.04	0.04	0.07
var	2.49	2.85	3.50	9.67
std.dev	1.58	1.69	1.87	3.11
coef.var	1.86	1.85	1.57	1.48

4 Methodology

Prediction, Clustering, Geoprocessing

Introduction

The methodology contained a three part procedure. The first procedure focused on Non-Negative Matrix Factorization (NNMF) for predicting the normalized values of each MFR-CRE property for every year from and including 2004 to 2018. The second procedure focused on Spatially Constrained Weighted-Multivariate Hierarchical Clustering (SCWMHC) for submarket identification. The third procedure focused on Geographic Information Systems (GIS) fundamentals for spatial modeling and geographic representation. The R implementation of the methodology described in this chapter was made available for reproduction and can be found in *Appendix E - Methodology*.

4.1 Non-Negative Matrix Factorization

4.1.1 Training & Test Split: Conditional Random Sampling Sparse Matrices

The first methodological procedure of this study focused on NNMF. It began with a novel conditional random sampling technique designed to sample out data from sparse matrices for train and test set splitting. One of the reasons this technique was conducted, was to reformulate a traditional unsupervised learning problem, into a supervised one (Hastie, Tibshirani, and Friedman 2009). This was inspired from the study *"Fast Nonnegative Matrix Factorization and Applications to Pattern Extraction, Deconvolution and Imputation"* by Lin and Boutros (2018). However, it was adapted for this study on a technical and theoretical basis.

The conditional random sampling technique developed for this study iterated through the $_{i}th$ observation in matrix A_{ij} and sampled a single $_{j}th$ value on the condition that number of $_{j}th$ values within the $_{i}th$ observation was > 1. It repeated this process until the specified sampling threshold was met. That is to say, the conditional random sampling algorithm sampled out a single observed value from each row that contained more than one observed value. It repeated this process for every row in the matrix until the sampling threshold was reached.

There were strong theoretical motivations for developing this conditional random sampling technique. In this study, completely sampling out values from observations, would theoretically mean that there was no sale observed for that particular MFR-CRE property. It was believed that it would be theoretically inappropriate to fit a model to training data that contained properties without a market, when measuring submarkets. In order to maintain consistency with submarket theory and the phenomena being measured in this study, it would have been theoretically difficult to proceed. For this reason, conditional random sampling for sparse matrices was introduced.

There were limitations to this conditional random sampling technique. If the sampling threshold was set too high, the algorithm would not be able to reach the stopping criteria and would iterate through the matrix until infinity. Recall that the conditional statement required every observation to contain at least one observed value. If every observation had only one observed value before the stopping criteria was reached, there was no way to proceed. As stated before, the algorithm would iterate forever. Therefore, the maximum sampling limit achieved here, was a roughly *80%* to *20%*, training to test set split, respectively.

4.1.2 Algorithm Selection: Iteration Maximum & Convergence Speed

Two NNMF algorithms were explored: Sequential Coordinate Wise Decent (SCD), and Lee & Seung's Multiplicative Update (Lee). Each algorithm was tested for a minimally sufficient number of iterations to reach sufficient convergence. *50*, *500*, and *5000* iterations were tested for both SCD and Lee. In testing for a minimally sufficient number of iterations, the k rank tuning parameter was set at *2*. The *alpha-beta* regularization was set at *0.0004 alpha*, *0.02 beta*, as a slightly higher adjustment to what has been observed in practice (Yeung 2010). There are limitations to this method. The *alpha* and *beta* regularization parameters could be optimized by virtue of cross validation. However, precisely tuning the NNMF model was beyond the scope of this study.

Detailed information regarding NNMF with SCD and Lee, and its application can be found in *"Fast Nonnegative Matrix Factorization and Applications to Pattern Extraction, Deconvolution and Imputation"* by Lin and Boutros (2018). More detailed information specifically regarding Sequential Coordinate Wise Decent can be found in *"Sequential Coordinate-Wise Algorithm for the Non-Negative Least Squares Problem"* by Franc, Hlaváč, and Navara (2005). For Lee & Seung's Multiplicative Update, more detailed information can be found in *"Learning the Parts of Objects by Non-Negative Matrix Factorization"* by Lee and Seung (1999).

The convergence speed results indicated that *50* iterations was a minimally sufficient number of iterations. While *50* iterations sufficed, the difference been SCD and Lee was such that SCD converged quicker and reached a lower Training MSE. Computational costs were also considered and were by in large regarded as a limitation for the *5000* iteration maximum. Therefore, the study proceeded with SCD and a *50* iteration maximum.

Iteration Maximum & Convergence Speed Comparison Plot

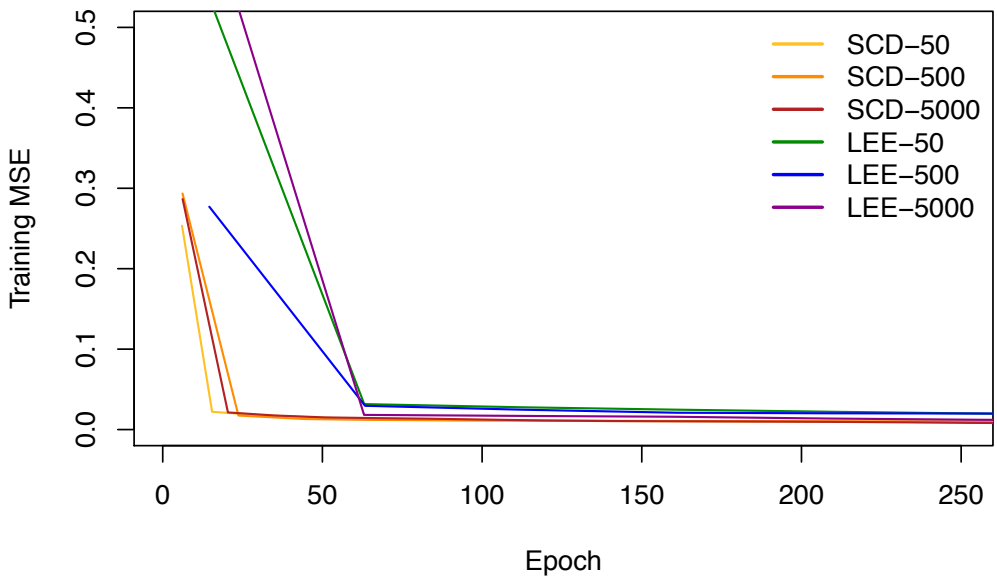

Figure 12: Non-Negative Matrix Factorization Convergence Speed Comparison Plot

Table 4: Iteration Maximum & Convergence Speed Comparison

	SCD-50	SCD-500	SCD-5000	LEE-50	LEE-500	LEE-5000
MSE	0.008712	0.005774	0.002488	0.005942	0.006558	0.002822
MKL	0.004168	0.008655	0.00205	0.003665	0.009246	0.002201
Target	0.08831	0.01209	0.002479	0.083	0.01242	0.002663
Rel. Tol.	0.01792	0.001604	9.997e-05	0.01735	0.001518	9.998e-05
Total Epoch	1002	1.074e+04	1.375e+05	2410	2.413e+04	2.037e+05
Interation Max	50	500	4519	50	500	4231
user.self	0.849	8.414	94.74	1.577	18.24	176.8
sys.self	0.005	0.042	0.331	0.005	0.054	0.571
elapsed	0.221	2.199	24.44	0.41	4.717	46.01

4.1.3 Leave One Out Cross Validation

The optimal low k rank was selected by Leave One Out Cross Validation (LOOCV). The NNMF model was cross validated with a range of k ranks from *1 to 8*. An algorithm was created to sample out a single observed value from the training data, then the NNMF model was used to predict that value using the k rank range (*1 to 8*). This was repeated for every value in the matrix. Meaning, exactly n models were fit for every rank from *1 to 8*. LOOCV was an exhaustive test that was able to get the most use of the existing information (Gareth et al. 2000). The results indicated that a single rank ($k = 1$) could sufficiently explain the underlying matrix. More information regarding LOOCV can be found in *"An Introduction to Statistical Learning with Applications in R"* by Gareth et al. (2000) and in *"The Elements of Statistical Learning"* by Hastie, Tibshirani, and Friedman (2009).

There were limitations to this method. Perhaps the most obvious one was the computational cost involved. The studied relied on a Google Compute Engine virtual machine in order to perform this particular technique. Because of the computational cost in implementing LOOCV, there are obvious practical reasons that this may not want to be considered in future studies. However, in an effort to demonstrate extending the maximum utility from the data, as eluded to in *Chapter 3 - Study Area & Data*, this study implemented LOOCV.

4.1.4 Prediction Performance Comparison

NNMF was compared with four other alternative methods for imputing values in sparse matrices. The four other methods were: Random, Row Means, Random Forest, and Simon Funk's Singular Value Decomposition with Stochastic Gradient Decent (Funk SVD). The Random method imputed missing values with those randomly observed within sparse matrix A_{ij}. The Row Means method imputed missing values by averaging the observed values in rows A_i. The Random Forest method utilized the non-linear ensemble model, Random Forest with the default tuning parameters. Detailed information regarding Random Forest can be found in *"An Introduction to Statistical Learning"* by Gareth et al. (2000) and in *"The Elements of Statistical Learning"* by Hastie, Tibshirani, and Friedman (2009). The Funk SVD method utilized the famed *"Netflix Prize"* algorithm with the default tuning parameters and $k = 1$ (equal to NNMF). Detailed information regarding Funk SVD, including explanation and notation can be found in *Matrix Factorization Techniques for Recommender Systems* by Koren, Bell, and Volinsky (2009). R implementation for Funk SVD can be found in *recommenderlab: A Framework for Developing and Testing Recommendation Algorithms* by Hahsler (2011). Comparing other alternative methods for imputing values in sparse matrices was not meant to fully explore alternatives, but rather to provide context NNMF. However, after conducting the experiment, there were unexpected results worth mentioning.

NNMF was not the highest performing prediction model. Funk SVD had marginally higher predictive accuracy for all matrices (variables) when compared to NNMF. Although Funk SVD had overall better performance when compared to NNMF, it was not more seriously considered because the algorithm did not contain a non-negative constraint. Random Forest also suffered from this limitation. As previously mentioned in *Chapter 3 - Study Area & Data*, there was a theoretical reason for maintaining non-negativity in the normalized MFR-CRE values in this study. Therefore, it was determined that NNMF was a more appropriate method because the results of the predictions were guaranteed to always be positive. In addition, NNMF performed nearly as well as the overall highest performing model in this experiment, Funk SVD.

Another unexpected result of this experiment was that Row Means actually had reasonably high predictive accuracy. In many cases, it out-performed the powerful non-linear ensemble model, Random Forest. In fact, Row Means actually out-performed NNMF in predictive accuracy. Furthermore, in the cases where NNMF had higher predictive accuracy, it was only marginally higher than Row Means. This seem to suggests two things. One, that MFR-CRE values are likely highly linear. Two, that they could be generally more stable than previously hypothesized. However, this required further investigation and was beyond the scope of this study. Although Row Means did tend to have higher predictive accuracy when measured by the Test MSE, no surprisingly, the Test MKL demonstrated that the NNMF prediction results had generally more consistent distributions with the observed values. Therefore, it was determined that NNMF was an appropriate method to proceed with.

The optimized NNMF model was repeated for all five matrices (variables) for final refitting, prediction, and imputation. Recall that the optimized NNMF model's k rank was specified at $1 = k$, and the *alpha-beta* regularization was specified at *0.0004 alpha*, *0.02 beta*. There were limitations to this method. It is fully recognized that not all five matrices (variables) would necessarily result in similar performance to the model tuned to the Sale Price matrix. However, recall that raw predictive performance was not the main focus of this study. The additional steps necessary to optimally tune each of the parameters for each feature engineered variable, was beyond the scope of this study and serves as the basis for future investigation.

Table 5: Prediction Performance Comparison of Price

	Random	Means	Forest	Funk SVD	NNMF
MSE	258.02	1.04	1.94	0.74	0.90
MKL	5.47	0.09	0.13	0.08	0.08

Table 6: Prediction Performance Comparison of Price Ratio / Residential Units

	Random	Means	Forest	Funk SVD	NNMF
MSE	22.43	0.16	0.18	0.15	0.17
MKL	1.27	0.06	0.04	0.04	0.05

Table 7: Prediction Performance Comparison of Price / Total Units

	Random	Means	Forest	Funk SVD	NNMF
MSE	15.79	0.21	0.40	0.19	0.23
MKL	0.99	0.06	0.05	0.04	0.05

Table 8: Prediction Performance Comparison of Price / Gross Square Footage

	Random	Means	Forest	Funk SVD	NNMF
MSE	187.12	0.36	0.37	0.37	0.47
MKL	3.00	0.11	0.06	0.06	0.10

Table 9: Prediction Performance Comparison of Price / Land Square Footage

	Random	Means	Forest	Funk SVD	NNMF
MSE	164.06	1.31	4.25	1.14	1.26
MKL	3.24	0.14	0.15	0.11	0.12

4.1.5 Remarks on Prediction

As a concluding remark regarding NNMF and the prediction results contained in this section, it was not assumed that any of them were necessarily correct. However, it was assumed that they were close enough in order to provide sufficiently accurate data to answer the research questions asked in this study. Although care was taken to minimize information loss and maximize prediction quality, the NNMF prediction methods exhibited in this study were an intermediate, but necessary step for the model specification contained in the second procedure, Spatially Constrained Weight-Multivariate Hierarchical Clustering. The R implementation of the NNMF prediction methods described in this section were made available for reproduction in *Appendix E - Methodology*.

4.2 Spatially Constrained Weight-Multivariate Hierarchical Clustering

The second procedure focused on Spatially Constrained Weight-Multivariate Hierarchical Clustering (SCWMHC). The clustering method was conducted using the Hierarchical Climate Regionalization algorithm (HiClimR). HiClimR was originally developed by Badr, Zaitchik, and Dezfuli (2015) for regionalizing rainfall across equatorial Africa (Badr, Zaitchik, and Dezfuli 2015). However, it was adapted for utilization in this study.

Conducting SCWMHC was fundamentally a classic agglomerative hierarchical clustering (AHC) approach (Badr, Zaitchik, and Dezfuli 2015). AHC is commonly found in the machine learning literature and in practice. For the sake of brevity, AHC was not more thoroughly covered in this study. Detailed information regarding AHC can be found in *"An Introduction to Statistical Learning with Applications in R"* by Gareth et al. (2000), and in *"The Elements of Statistical Learning: Data Mining, Inference, and Prediction"* by Hastie, Tibshirani, and Friedman (2009). For even greater detail regarding AHC, the subject was more thoroughly covered by Podani (2000) in *"Chapter 5 of Introduction to the Exploration of Multivariate Biological Data"*.

Hierarchial Climate Regionalization Algorithm Detailed Flow Chart

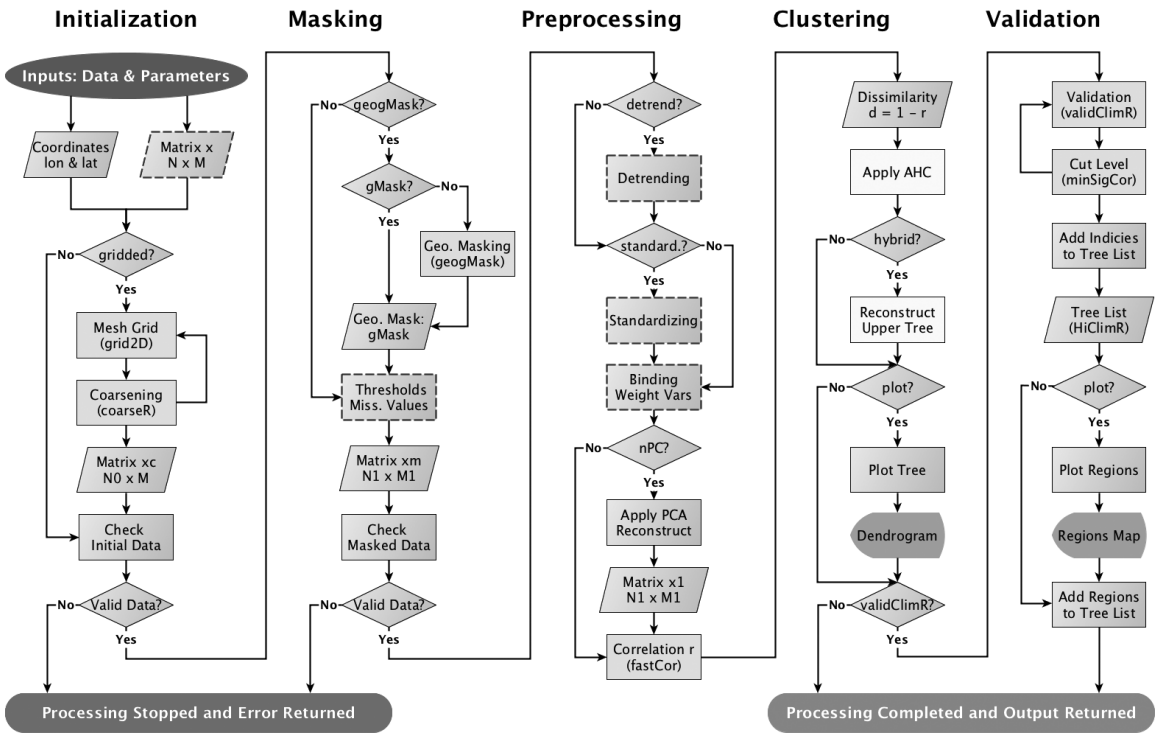

Figure 13: HiClimR Algorithm Detailed Flow Chart (Badr et al. 2015)

4.2.1 Spatial Contiguity Constraint

Although the SCWMHC model in this study was fundamentally similar to a traditional AHC model, it incorporated an additional spatial contiguity constraint. The HiClimR algorithm contains a unique parameter, which was utilized to adjust the dissimilarity matrix (multivariate space) account for spatial contiguity (geographic space). That is to say, the spatial contiguity constraint was applied in order to weight the dissimilarity matrix to account for the geographic distances between the geocoordinates of each observation (normalized observed and predicted MFR-CRE values).

The SCWMHC model was specified using a high spatial contiguity constraint. The authors outlined that the parameter should be a non-negative real value, which the model was specified to. However, there was no further guidance on how to tune the parameter, other than the recommendation between *0* and *1*, where *0* was unconstrained. Recall that the original motivation for developing this algorithm was to regionalize rainfall across equatorial Africa, a continental scale application. The data drawn from weather stations across vast geography is different than the urban scale data contained in this study.

The urban data in this study was specified and analyzed at a much lower geographic scale or higher spatial resolution than what the algorithm was originally designed for. Weather station data is likely to observe less dramatic differences between geographically near observations when compared to MFR-CRE observed and predicted MFR-CRE values. Accounting for the difference between the original application for the HiClimR algorithm and the interests of this study, the settings for the spatial contiguity constraint relied on Tobler's First Law of Geography in combination with Bourassa's submarket definition for guidance. In other words, tuning the spatial contiguity constraint relied on theory of guidance. Recall that Tobler's First Law of Geography States that:

> *"everything is related to everything else, but near things are more related than distant things"* (Miller 2004)

Also recall Bourassa's definition of submarkets:

> *"a set of dwellings that are reasonably close substitutes for one another, but relatively poor substitutes for dwellings in other submarkets."* (Bourassa et al. 1999).

It was determined that in order to capture the greatest degree of MFR-CRE substitution, while also capturing the greatest degree of geographic relatedness, it was reasonable to parametrically (not deterministically) ensure that the submarket models exhibit a tightly controlled geographic compactness. The way this was achieved was by specifying a high, but arbitrary number to ensure that the submarket models were theoretically consistent with the combined definitions of both Tobler and Bourassa. Therefore, the study proceeded in specifying the spatial contiguity constraint at *12345*. There were limitations to this method. Relying on theory for guidance in this regard is open to improvement. However, this approach was one way to maintain consistency with what was understood in the literature as theoretically prudent.

4.2.2 Multivariate Weighted Clustering

The algorithm also included a parameter, which was utilized to specify the weight assigned to each matrix (variable). Weighting each variable differently is known axiomatically as Multivariate Weighted Clustering (MVC). The correlation distance of MVC represents the weighted average of the distances between all variables, as outlined by Badr, Zaitchik, and Dezfuli (2015).

The five weighted variables were previously specified in *Chapter 3 - Study Area & Data*. This study weighted the five variables based on the resolution of price comparability across MFR-CRE assets using common standards found in the MFR-CRE industry (Schmitz 2000). For example, the results from the X_4 variable contained the highest resolution of price based comparison across MFR-CRE assets. Therefore, X_4 was weighted the highest. Where as the X_1 variable contained the lowest resolution of price based comparison. Therefore, X_1 was weighted the lowest. X_5 was inconsistent in this regard. However, was manually reweighed to reflect common industry standards of price comparability across MFR-CRE assets (Schmitz 2000). There were limitations to this method. Determining the weight assigned to each variable specified in the model is subject to improvement. However, this approach was one way to maintain some degree of consistency with the price comparability standards observed from within industry. Below are the weights specified to the matrices (variables).

List of Multivariate Weight Specifications

- *5 * X_4 = Normalized Sale Price / Gross Square Footage*
- *4 * X_3 = Normalized Sale Price / Total Number of Units (Including Commercial Units)*
- *3 * X_2 = Normalized Sale Price Ratio of Residential Units / Number of Residential Units*
- *2 * X_5 = Normalized Sale Price / Land Square Footage*
- *1 * X_1 = Normalized Sale Price*

4.2.3 Linkage Methods

There were five linkage methods explored using SCWMHC. All five linkage methods utilized Euclidean distance for measuring the distance between clusters. Euclidean distance was specified as a distance metric, as it is perhaps the most common and generally understood method in traditional AHC (Gareth et al. 2000). Also, the number of clusters was truncated at *100*. There were limitations to this method. However, it was thought that *2 to 100* accounted for a sufficient enough range to capture the optimal number of clusters or submarkets in Manhattan, while effectively reducing spatial complexity in a way that the results could be easily interpreted. The five linkage methods explored in this study were: Average Linkage, Median Linkage, Complete Linkage, Centroid Linkage, and Ward's Method.

Average Linkage (Group Average) was utilized in this study to link two different clusters by measuring the distance between the arithmetic average of between-cluster distance values. This was understood to be perhaps the most common of the five linkage methods explored here (Podani 2000).

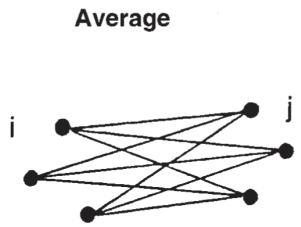

Figure 14: Average Linkage Schematic (Podani 2000)

Median Linkage (Weighted Pair Group Method Using Centroids) was utilized in this study to link two different clusters by the weighted distance between cluster centroids. The weight was controlled by the proportion of the number of observations per group (Podani 2000).

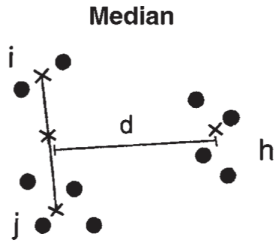

Figure 15: Median Linkage Schematic (Podani 2000)

Complete Linkage (Furthest Neighbor) was utilized in this study to link two different clusters by measuring the distance between their furthest observations. This method focused on cluster cohesion, rather than cluster separation (Podani 2000).

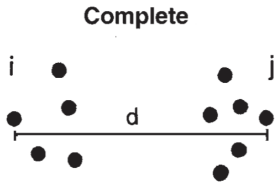

Figure 16: Complete Linkage Schematic (Podani 2000)

Centroid Linkage (Unweighted Pair Group Method Using Centroids) was utilized in this study to link two different clusters by the distance between clusters centroids (Podani 2000). It was recommended that this method was best used with Euclidean distance for measuring the the distance between clusters (NCSS 2019), which this study acknowledged.

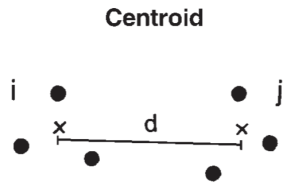

Figure 17: Centroid Linkage Schematic (Podani 2000)

Ward's Method (Incremental Sum of Squares) was utilized in this study to link different clusters by minimizing the within cluster sum of squares. It was included here as perhaps the most common homogeneity optimization method, which appears in the biological sciences (Podani 2000).

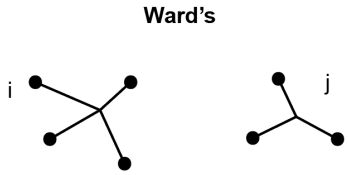

Figure 18: Ward's Method Schematic

4.2.4 Remarks on Clustering

Testing these five linkage methods was conducted in order to explore how various clustering results might differ. In this case, the number of clusters were directly equivalent to the number of submarkets. The linkage methods were analyzed with an objective criteria for determining the optimal number of clusters. The optimal number of clusters was determined by measuring the maximum absolute value of the difference between the mean intra-cluster correlation (average within cluster similarity) and the maximum inter-cluster correlation (maximum outside of cluster dissimilarity). The correlation range was specified from 0 to 1 (absolute value basis).

It is important to recall that the linkage methods calculated in the dissimilarity matrix naturally fall within multivariate space, not geographic space. However, because the spatial contiguity constraint parameter was tuned to tightly control for geographic compactness, it influenced the results found in the dissimilarity matrices to account for Tobler's First Law of Geography. Moreover, the MVC also influenced the dissimilarity matrices by the integral rank weight specified for each of the variables. The R implementation of the SCWMHC method described in this section was made available for reproduction in *Appendix E - Methodology*.

4.3 Geographic Information Systems

The third procedure focused on geographically modeling each MFR-CRE property's observed and predicted value to its corresponding land lot polygon, and labeling each with its corresponding submarket identification. This procedure was conducted using basic geoprocessing functions with a Geographic Information System (GIS). The submarket geography was modeled by spatial joining the geocoordinate point data containing the submarket identification information, to the intersecting land lot polygons in the PLUTO data. Then, the individual land parcel boundaries were color classified under the criteria that they contained the same submarket identification information (same submarket identification, same color code). After the spatial join was conducted, the results were visualized and discussed in *Chapter 5 - Results & Discussion*.

4.3.1 Remarks on Geoprocessing

GIS was introduced to more appropriately represent the geography of the submarket models exhibited in this study. GIS was not introduced to include geoprocessing for conducting spatial analysis. However, it was considered, but ultimately fell beyond the scope of this study. This limitation serves as a natural next step for future investigation, but was beyond the scope of this study. The R implementation of the GIS methods described in this section were made available for reproduction in *Appendix E - Methodology*.

5 Results & Discussion

Submarket Identification & Analysis

5.1 Results

The results from the SCWMHC procedure were analyzed with an objective criteria for determining the optimal number of clusters. Recall that the optimal number of clusters was determined by measuring the maximum absolute value of the difference between the mean intra-cluster correlation (average within cluster similarity) and the maximum inter-cluster correlation (maximum outside of cluster dissimilarity). The correlation range was specified from 0 to 1 (absolute value basis). Also, recall the high spatial contiguity constraint placed on the dissimilarity matrix, as well as the MVC weight specification.

5.1.1 Average Linkage

Average Linkage found that 22 was the optimal number of clusters, when the minimum number of clusters was 2 and the maximum number to cluster was 100. For **22 clusters**, the absolute value of the difference between the mean intra-cluster correlation and the maximum inter-cluster correlation was **0.1068**.

The lowest performing solution using Average Linkage and the previous objective criteria found that 98 clusters was the most suboptimal. For 98 clusters, the absolute value of the difference between the mean intra-cluster correlation and the maximum inter-cluster correlation was 0.0002.

Table 10: Average Linkage Correlation Summary

	interCor	intraCor	diffCor
Min.	0.7416503	0.8930840	-0.1067616
1st Qu.	0.9781198	0.9421859	-0.0576598
Median	0.9889730	0.9542043	-0.0456413
Mean	0.9801291	0.9482760	-0.0473425
3rd Qu.	0.9944972	0.9658599	-0.0339857
Max.	0.9998456	1.0000000	0.0001544

Average Linkage Cluster Correlation Differences Histogram

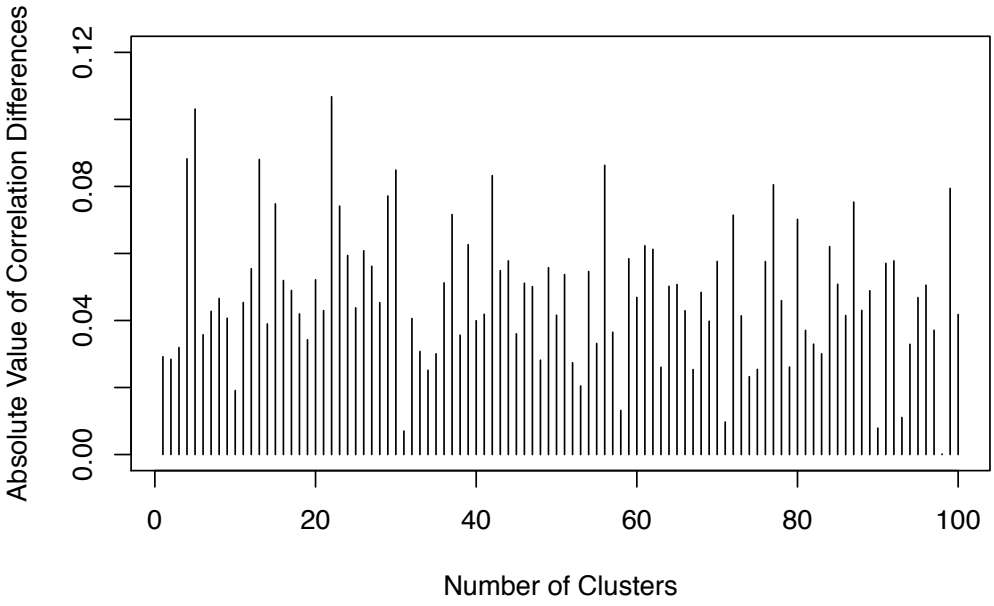

Figure 19: Average Linkage Cluster Correlation Differences Histogram

Average Linkage Dendrogram: Cut & Cluster Results

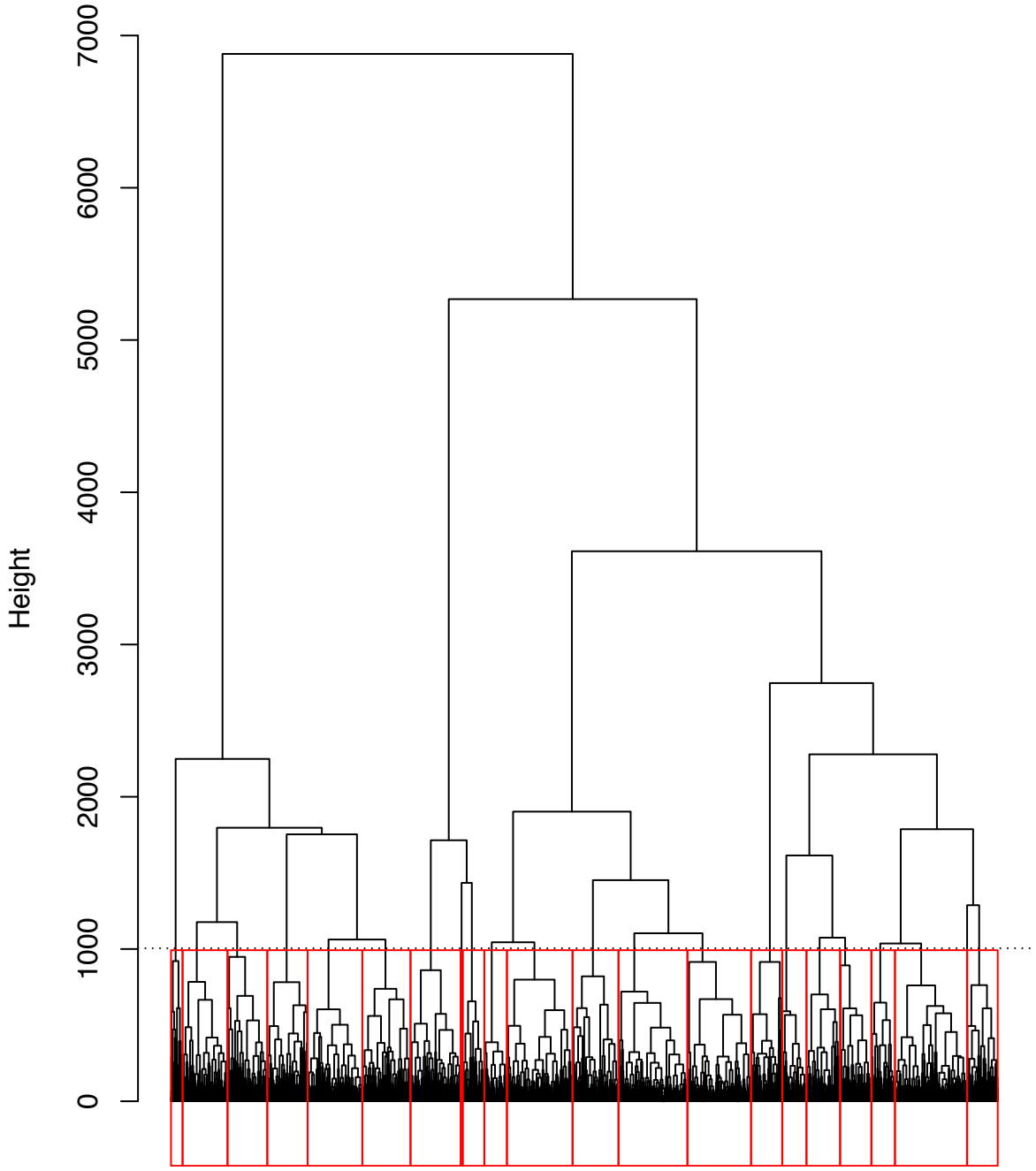

Figure 20: Average Linkage Dendrogram

5.1.2 Median Linkage

Median Linkage found that 39 was the optimal number of clusters, when the minimum number of clusters was 2 and the maximum number to cluster was 100. For **39 clusters**, the absolute value of the difference between the mean intra-cluster correlation and the maximum inter-cluster correlation was **0.1152**.

The lowest performing solution using Median Linkage and the previous objective criteria found that 24 clusters was the most suboptimal. For 24 clusters, the absolute value of the difference between the mean intra-cluster correlation and the maximum inter-cluster correlation was 0.0002.

Table 11: Median Linkage Correlation Summary

	interCor	intraCor	diffCor
Min.	0.7067924	0.8846275	-0.1151743
1st Qu.	0.9756532	0.9386296	-0.0611721
Median	0.9881621	0.9579792	-0.0418226
Mean	0.9753037	0.9484692	-0.0429057
3rd Qu.	0.9942467	0.9706144	-0.0291873
Max.	0.9998017	1.0000000	0.0001983

Median Linkage Cluster Correlation Differences Histogram

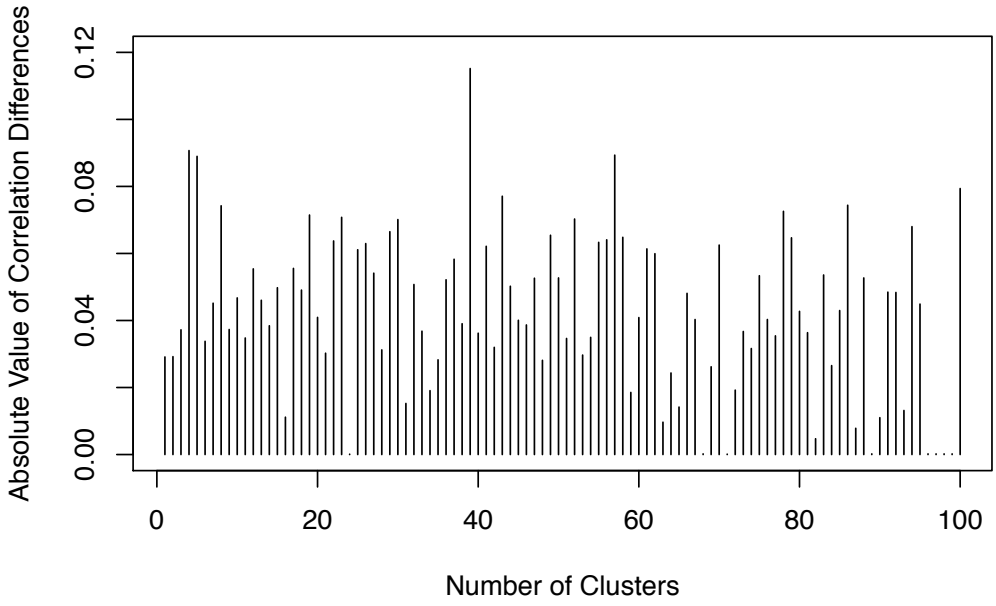

Figure 21: Median Linkage Cluster Correlation Differences Histogram

Median Linkage Dendrogram: Cut & Cluster Results

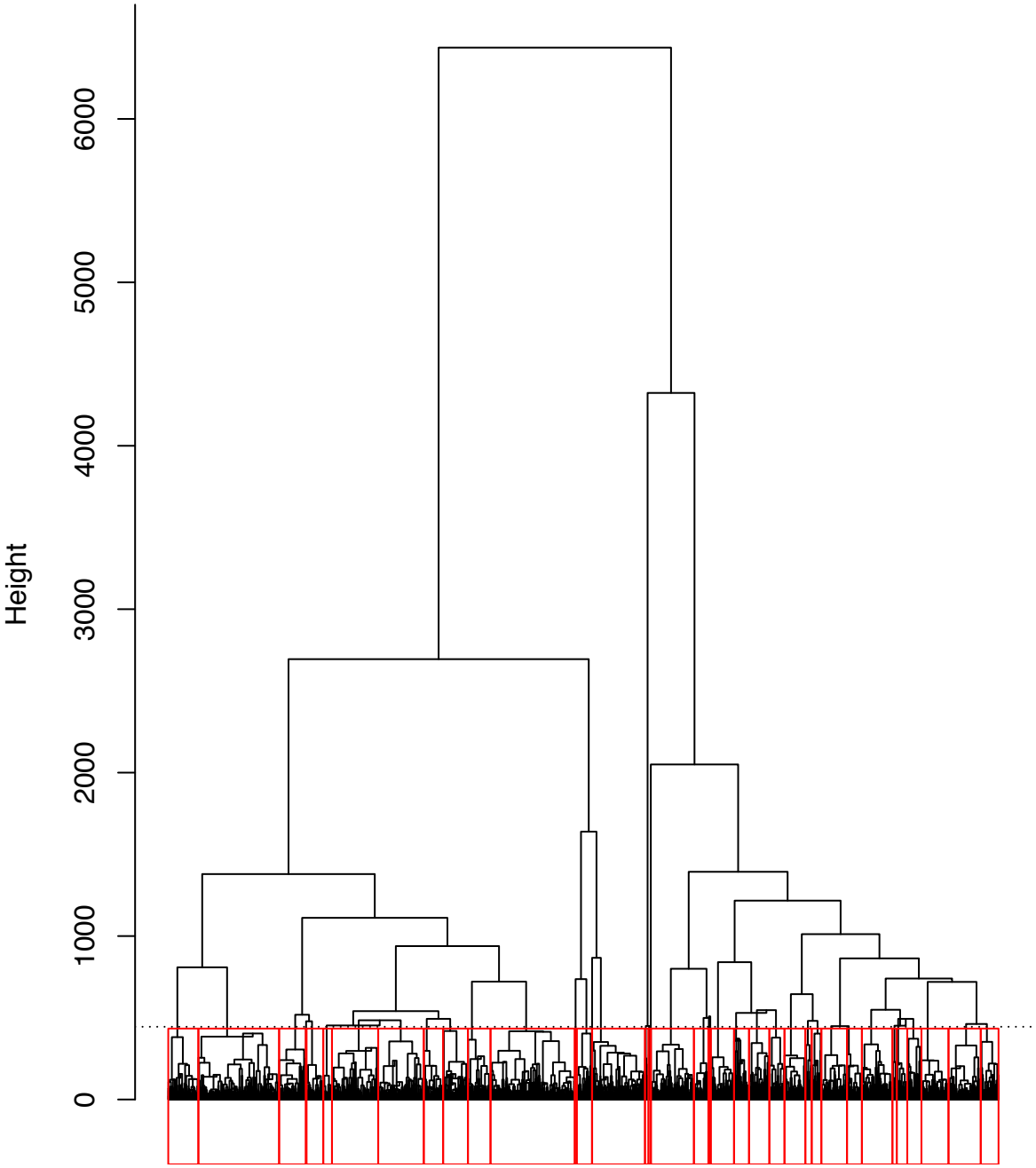

Figure 22: Median Linkage Dendrogram

5.1.3 Complete Linkage

Complete Linkage found that 14 was the optimal number of clusters, when the minimum number of clusters was 2 and the maximum number to cluster was 100. For **14 clusters**, the absolute value of the difference between the mean intra-cluster correlation and the maximum inter-cluster correlation was **0.1007**.

The lowest performing solution using Complete Linkage and the previous objective criteria found that 99 clusters was the most suboptimal. For 99 clusters, the absolute value of the difference between the mean intra-cluster correlation and the maximum inter-cluster correlation was 0.0002.

Table 12: Complete Linkage Correlation Summary

	interCor	intraCor	diffCor
Min.	0.8519831	0.8990999	-0.1007238
1st Qu.	0.9825920	0.9394462	-0.0603774
Median	0.9907321	0.9515716	-0.0482520
Mean	0.9853599	0.9477320	-0.0488574
3rd Qu.	0.9953178	0.9646799	-0.0351437
Max.	0.9998236	1.0000000	0.0001764

Complete Linkage Cluster Correlation Differences Histogram

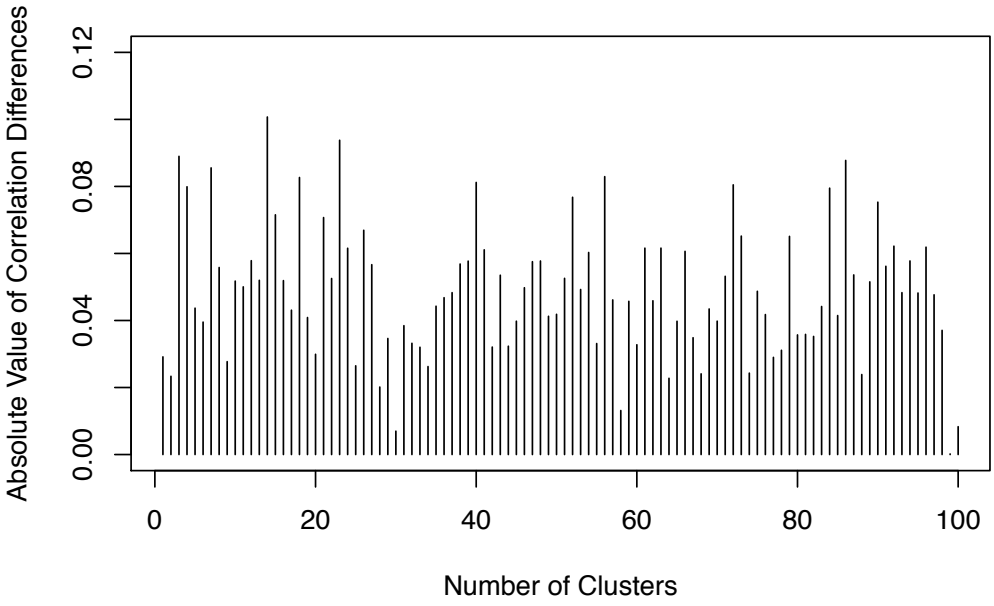

Figure 23: Complete Linkage Cluster Correlation Differences Histogram

Complete Linkage Dendrogram: Cut & Cluster Results

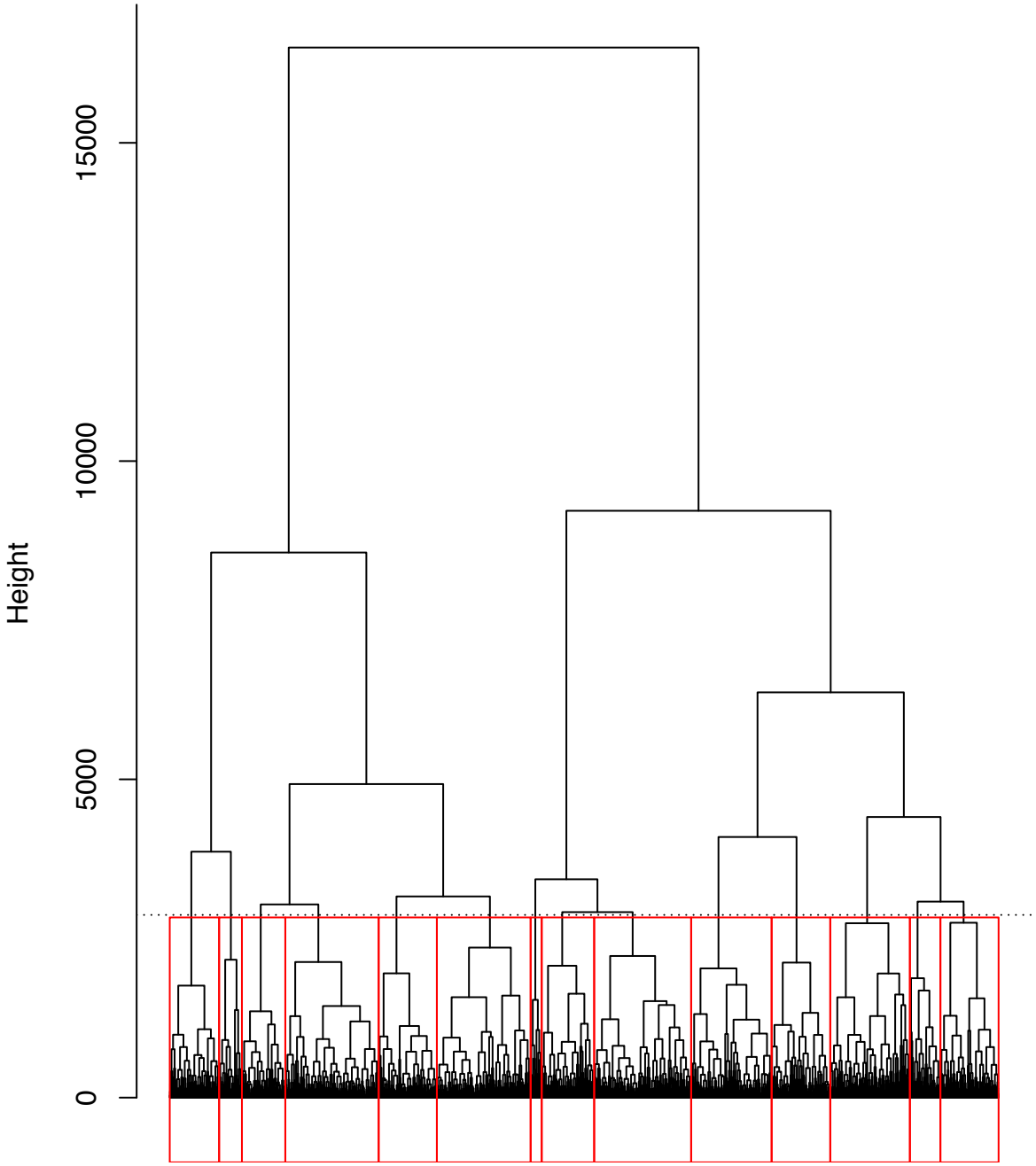

Figure 24: Complete Linkage Dendrogram

5.1.4 Centroid Linkage

Centroid Linkage found that 20 was the optimal number of clusters, when the minimum number of clusters was 2 and the maximum number to cluster was 100. For **20 clusters**, the absolute value of the difference between the mean intra-cluster correlation and the maximum inter-cluster correlation was **0.1068**.

The lowest performing solution using Centroid Linkage and the previous objective criteria found that 69 clusters was the most suboptimal. For 69 clusters, the absolute value of the difference between the mean intra-cluster correlation and the maximum inter-cluster correlation was 0.0001.

Table 13: Centroid Linkage Correlation Summary

	interCor	intraCor	diffCor
Min.	0.5926963	0.8930840	-0.1068080
1st Qu.	0.9626231	0.9454105	-0.0544816
Median	0.9851853	0.9582846	-0.0416075
Mean	0.9687695	0.9486799	-0.0414846
3rd Qu.	0.9936216	0.9703105	-0.0295815
Max.	0.9998921	1.0000000	0.0001079

Centroid Linkage Cluster Correlation Differences Histogram

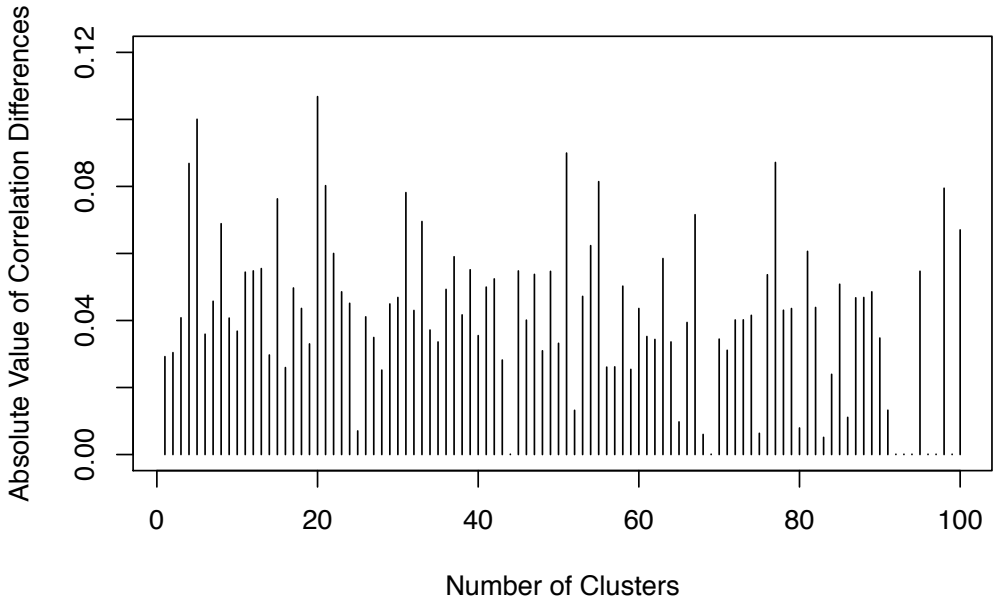

Figure 25: Centroid Linkage Cluster Correlation Differences Histogram

Centroid Linkage Dendrogram: Cut & Cluster Results

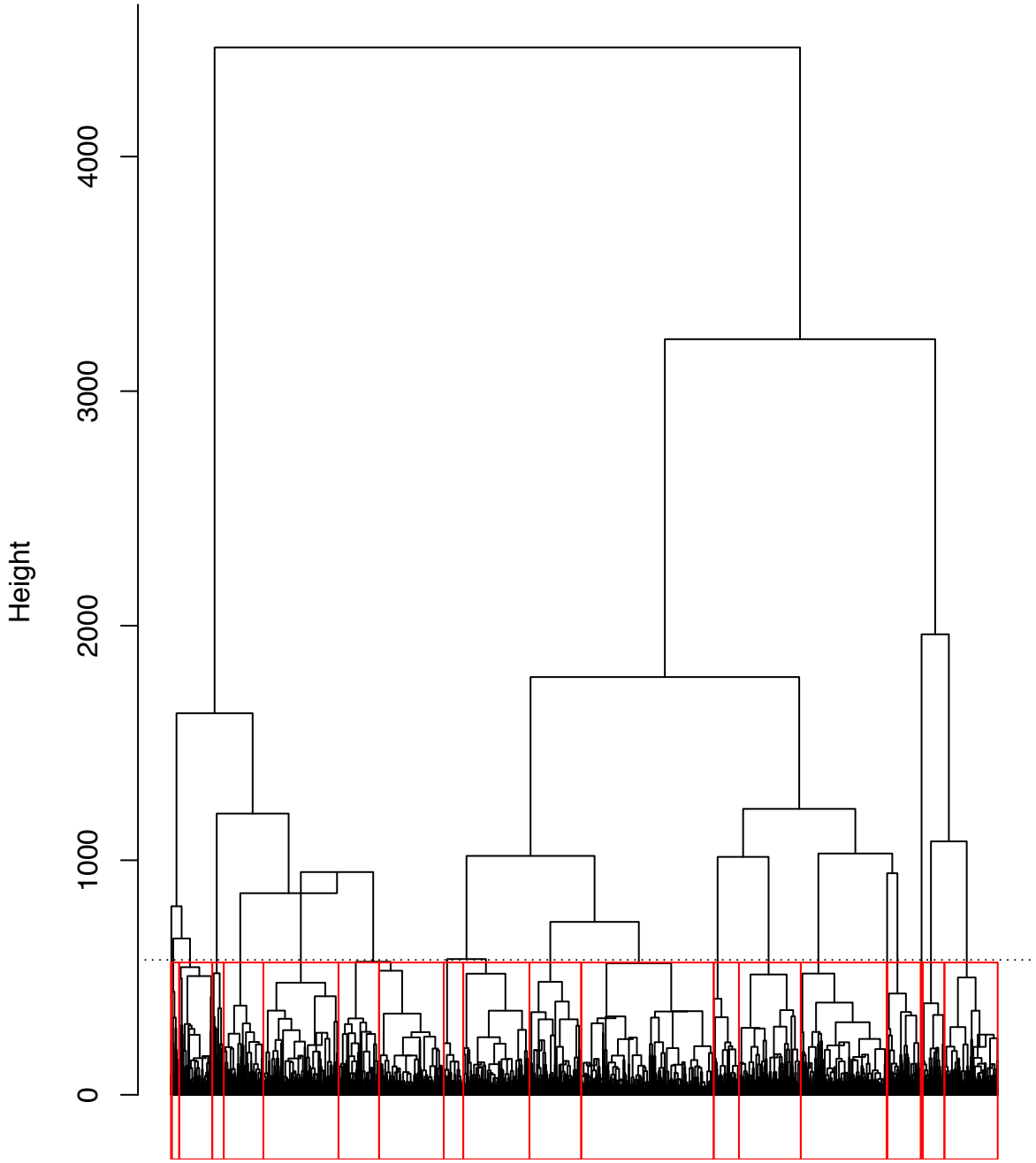

Figure 26: Centroid Linkage Dendrogram

5.1.5 Ward's Method

Ward's Method found that 43 was the optimal number of clusters, when the minimum number of clusters was 2 and the maximum number to cluster was 100. For **43 clusters**, the absolute value of the difference between the mean intra-cluster correlation and the maximum inter-cluster correlation was **0.1167**.

The lowest performing solution using Ward's Method and the the previous objective criteria found that 62 clusters was the most suboptimal. For 62 clusters, the absolute value of the difference between the mean intra-cluster correlation and the maximum inter-cluster correlation was 0.0132.

Table 14: Ward's Linkage Correlation Summary

	interCor	intraCor	diffCor
Min.	0.8655394	0.8831795	-0.1166651
1st Qu.	0.9826398	0.9343307	-0.0655140
Median	0.9907541	0.9499244	-0.0499202
Mean	0.9863059	0.9479982	-0.0515178
3rd Qu.	0.9954491	0.9632391	-0.0366055
Max.	0.9998446	0.9866903	-0.0131543

Ward's Method Cluster Correlation Differences Histogram

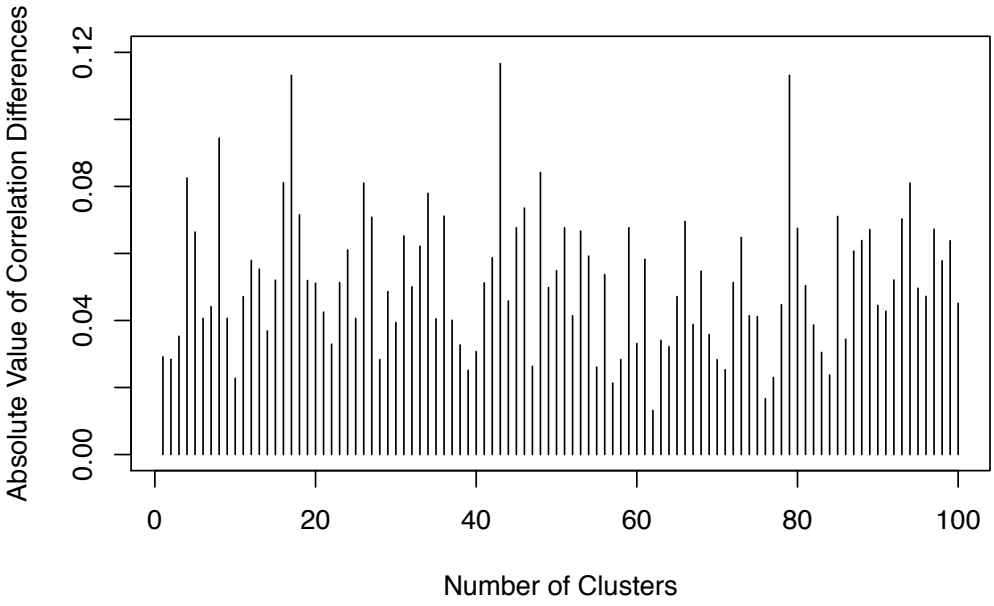

Figure 27: Ward's Method Cluster Correlation Differences Histogram

Ward's Method Dendrogram: Cut & Cluster Results

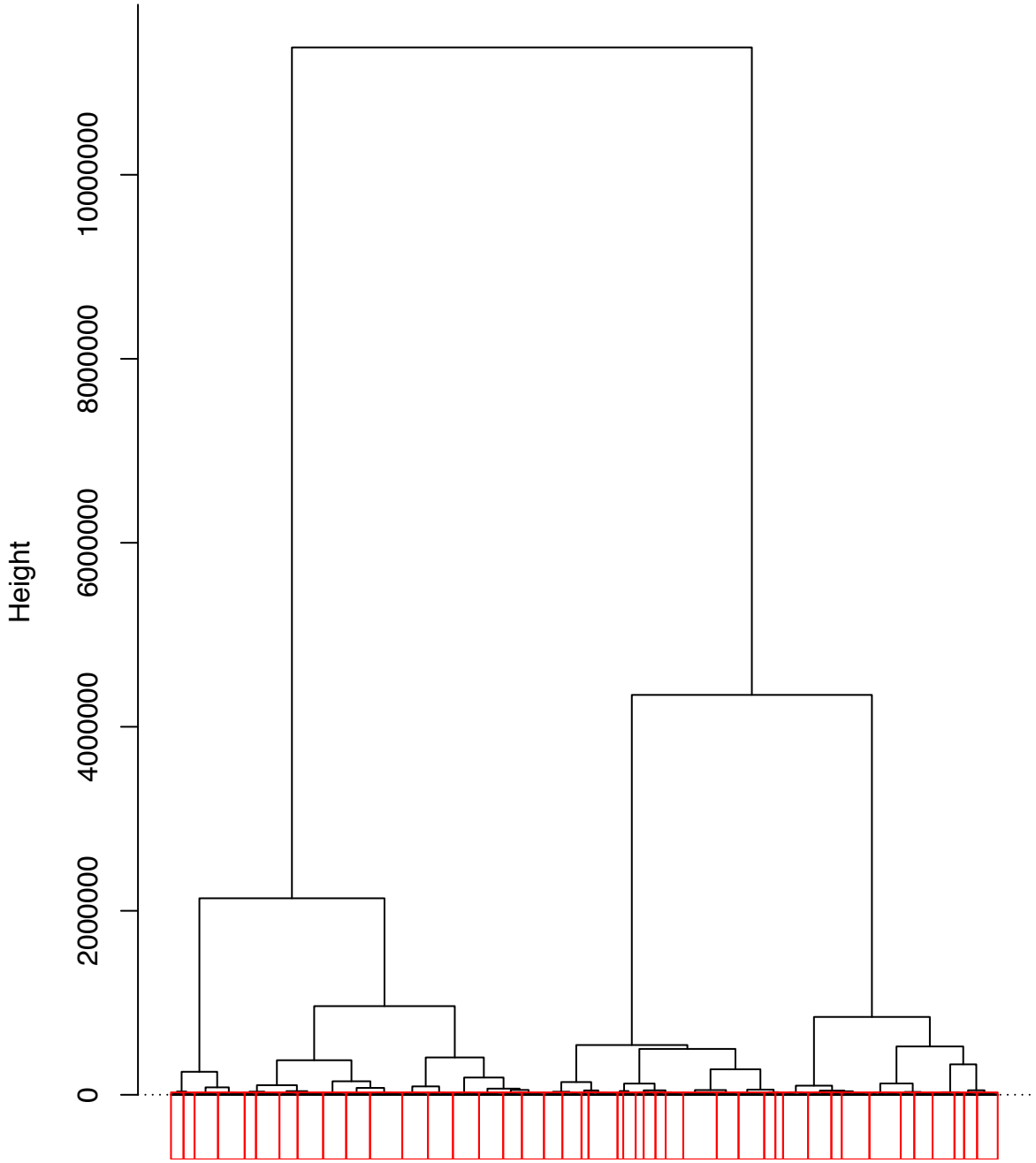

Figure 28: Ward's Method Dendrogram

5.2 Discussion

5.2.1 Comparing the Highest Performing Results

The results indicated that the highest performing linkage method when measured by the maximum absolute value of the difference between the mean intra-cluster correlation (average within cluster similarity) and the maximum inter-cluster correlation (maximum outside of cluster dissimilarity) was Ward's Method. The absolute value of the correlation difference was 0.1167, when the correlation range was specified from 0 to 1 (absolute value basis). Ward's Method found that 43 was the optimal number of clusters.

The lowest performing linkage method when measured by the maximum absolute value of the difference between the mean intra-cluster correlation (average within cluster similarity) and the maximum inter-cluster correlation (maximum outside of cluster dissimilarity) was Complete Linkage. The absolute value of the correlation difference was 0.1007, when the correlation range was specified from 0 to 1 (absolutely value basis). Complete Linkage found that 14 was the optimal number of clusters.

Table 15: Linkage Method Comparison of High Performing Results

	Average	Median	Complete	Centroid	Ward's
Number of Clusters	22.0000000	39.0000000	14.0000000	20.000000	43.0000000
Correlation Difference	-0.1067616	-0.1151743	-0.1007238	-0.106808	-0.1166651

5.2.2 Comparing the Lowest Performing Results

Conversely, the linkage methods were also compared utilizing the lowest performing results under the same objective evaluation criteria. That is to say, the results were analyzed to find the optimal number of clusters using the lowest performing result of each of the linkage methods. This was conducted to further validate the linkage method specification. The results from evaluating the optimal linkage method using the most suboptimal of each of their results, indicated that Ward's Method was still the optimal linkage method, while Median Linkage was the most suboptimal.

Recall that the lowest performing solution using Ward's Method was 0.0132 with 62 clusters, when measured by the same evaluation criteria as before. Also recall that the lowest performing solution using Median Linkage was 0.0001 with 24 clusters, when measured by the same evaluation criteria as before. This seemed to confirm the validity of Ward's Method as an appropriate linkage specification. However, raises other questions regarding the range of clustering performance using Median Linkage, as Median Linkage was the second highest performing linkage method when measured by the *highest* performing of each of the linkage method results, but also the lowest performing when measured by the *lowest* performing of each of the linkage method results.

Table 16: Linkage Method Comparison of Low Performing Results

	Average	Median	Complete	Centroid	Ward's
Number of Clusters	98.0000000	24.0000000	99.0000000	96.0000000	62.0000000
Correlation Difference	0.0001544	0.0001072	0.0001764	0.0001079	-0.0131543

5.2.3 Ward's Method

Although Ward's Method had superior results of under the absolute value of the correlation differences criteria, it is important to mention that its performance was only marginally better than the second best performing linkage method, Median Linkage. Recall that the Ward's Method result was 0.1168 with 43 clusters and the Median Linkage result was 0.1152 with 39 clusters. Moreover, Ward's Method was just marginally better than even the lowest performing linkage method, Complete Linkage. Recall that the Complete Linkage result was 0.1007 with 14 clusters.

What this seemed to have suggested is that although Ward's Method was the optimal linkage method and 43 clusters were found to be the optimal number of clusters, there were still concerns regarding the reliability of the solution. Had the margins of the results exhibited higher differences, they might be interpreted with less criticism. However, in this case, it is important to view them with some additional scrutiny.

5.2.4 Similar Results Between Linkage Methods

There were also other important findings worth considering. The results suggested that there was a small range in the optimal number of clusters that different linkage methods shared. For example, Recall that Ward's Method found that the optimal number of clusters was 43. Median Linkage found that 39 was the optimal number of clusters. The approximate range between 43 and 39 clusters may suggest there is some underlying reliability in that small range of clusters underlying the data.

Another way this was viewed, was by searching for a small range of the most suboptimal number of clusters to *avoid* considering. Approximating the most suboptimal range of clusters was determined by measuring the two lowest performing results under the same absolute value correlation difference criteria. It was found that the most suboptimal linkage methods were Average Linkage and Complete Linkage, where the lowest performing number of clusters was 98 and 99, respectively. What this seemed to have suggested, was that the number of clusters roughly near 100 was the least worth considering.

This analysis was also conducted *ad hoc*. When visually inspecting the cluster histograms of all the linkage methods, the approximate range between 60 and 70 clusters generally appeared to exhibit low absolute value correlation differences. Although this was not empirically validated, visually analyzing the data helped to develop intuitions about it, which could be empirically tested later. However, testing them was beyond the scope of this study and was largely conducted to demonstrate alternative ways to interpret the findings.

5.2.5 Submarket Identification

The final clustering results using Ward's Method, were directly applied to classify each MFR-CRE property into a submarket. The clustering results indicated that a perfect match was achieved in classifying all MFR-CRE properties into a submarket. However, the spatial join results not illustrated by a color (illustrated by dark gray) on the submarket maps were due to unknown spatial join errors. Although all submarket identifications were not able to be joined to their respective land lot polygons, it appeared that a largely sufficient enough number of spatial joins were achieved for interpretable results when illustrated on the Submarket Map of Manhattan and the various neighborhood maps. The R implementation of the results described in this section were made available for reproduction in *Appendix F - Results*.

5.2.6 Remarks on Future Results

Although the the SCWMHC model was trained to identify submarkets in Manhattan based on historic data from 2004 to 2018, one of the issues that was raised while conducting this study was the reliability of the results in future analysis. Although the results are likely to be generally reliable in a contemporary study due to the infrequent occurrence of MFR-CRE transactions, as demonstrated in the *"Data Exploration"* section of *Chapter 3 - Study Area & Data*. Another, perhaps more empirically prudent recommendation in this regard would be to simply update the models when new data becomes available. Because of the R implementation conducted in this study, new models could be produced for every new observation or time series of observations, which becomes available.

As previously outlined throughout this study, all implementation steps in R were made available in the *Appendicies*. Reproduction is actively encouraged and is the reason for providing full transparency. It might even be worth exploring a completely different study area and scope. In the spirit of the open source ethos, please observe appropriate citation etiquette when using the resources made available in this study.

Submarket Map of Manhattan from 2004 to 2018

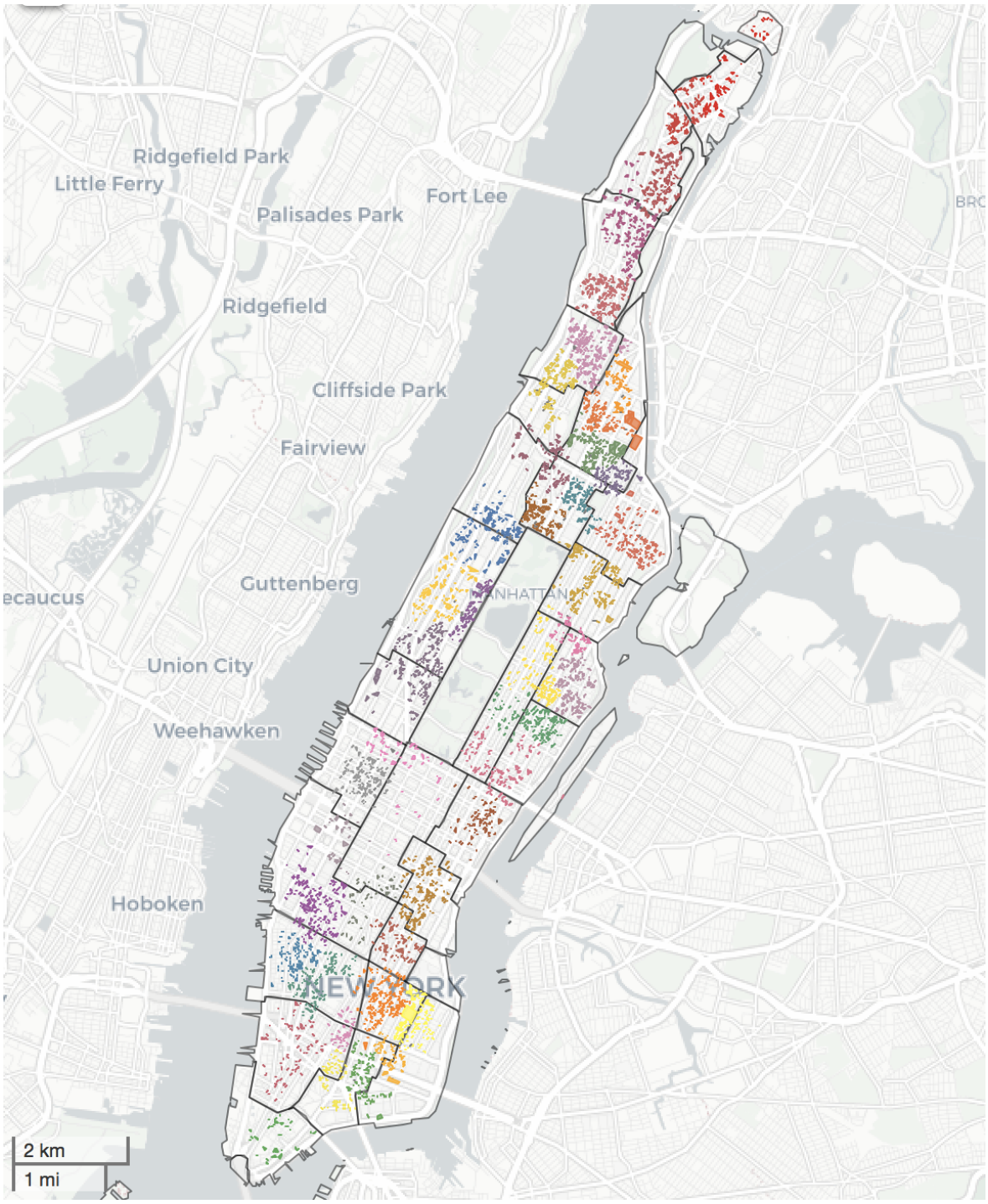

Figure 29: Submarket Map of Manhattan

Submarket Map of Central Harlem from 2004 to 2018

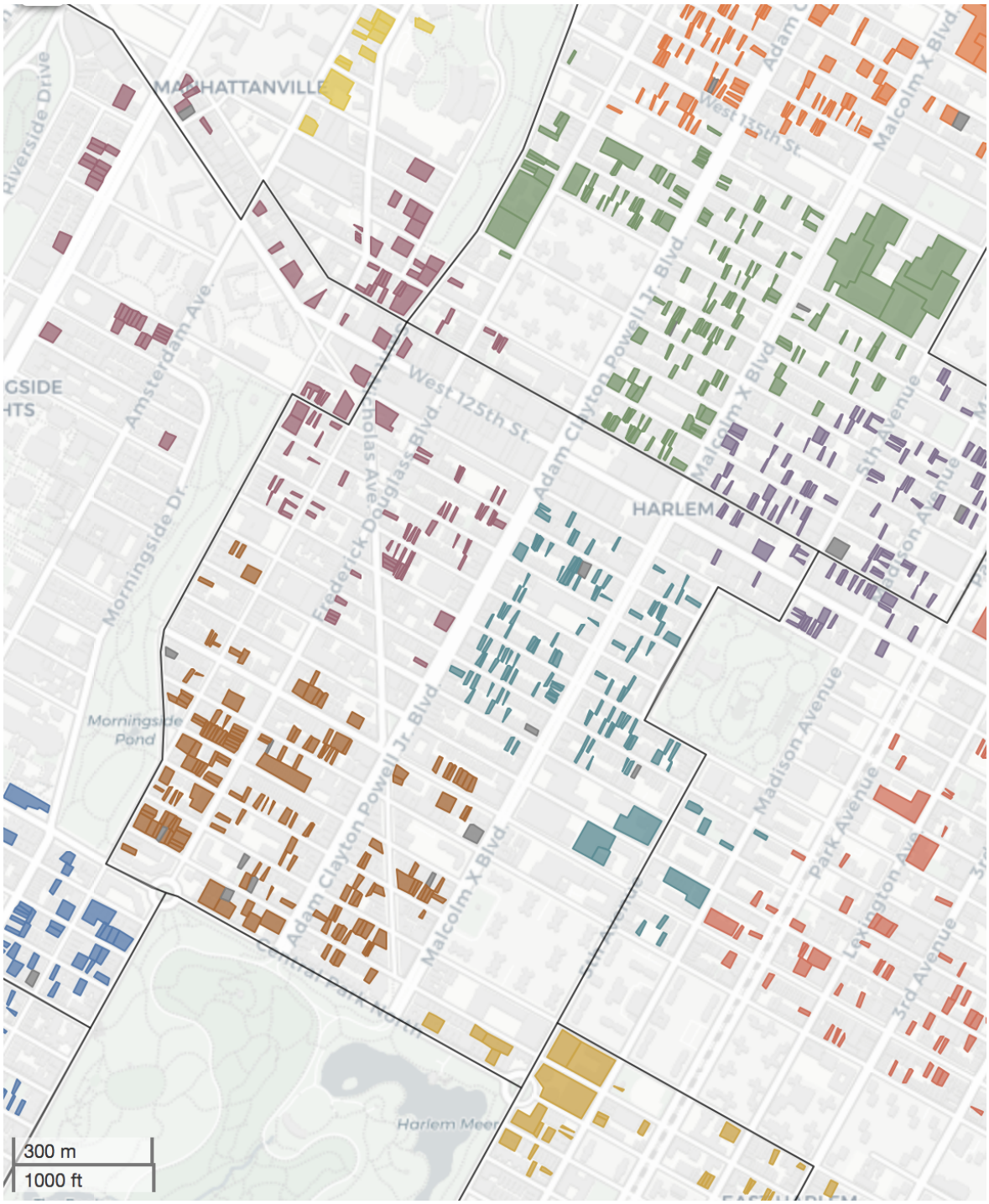

Figure 30: Submarket Map of Central Harlem

Submarket Map of Chelsea from 2004 to 2018

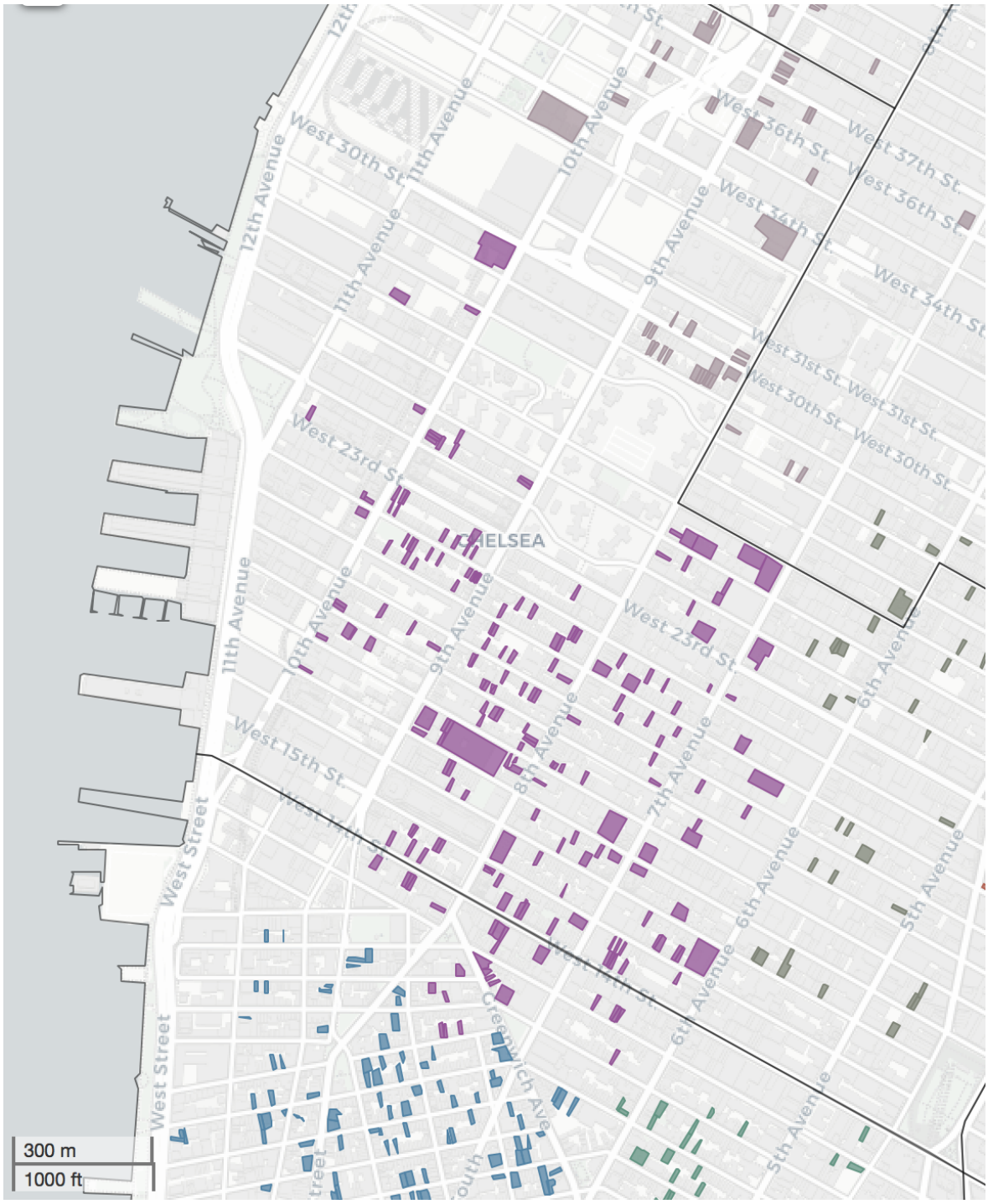

Figure 31: Submarket Map of Chelsea

Submarket Map of East Harlem from 2004 to 2018

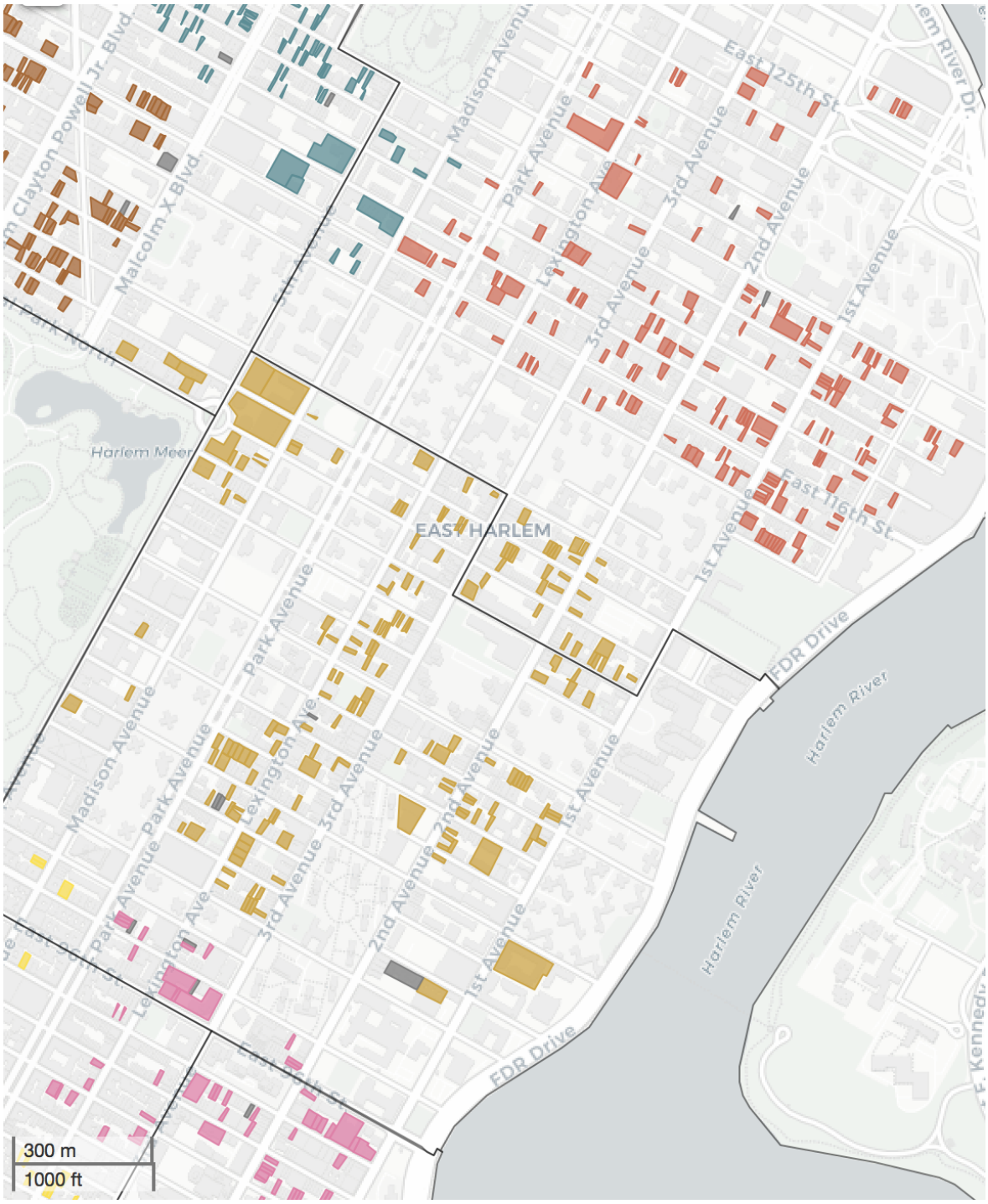

Figure 32: Submarket Map of East Harlem

Submarket Map of Hamilton Heights from 2004 to 2018

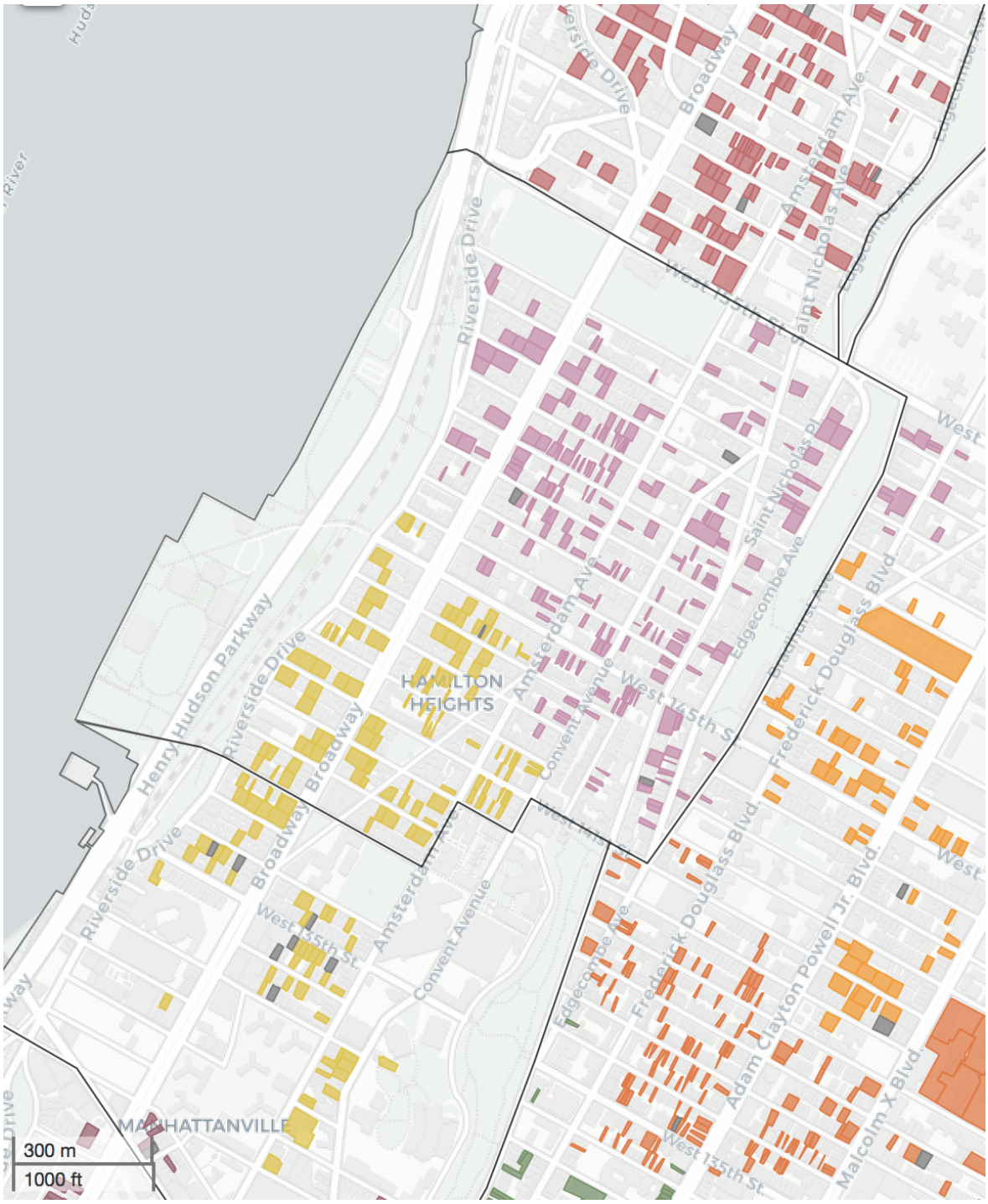

Figure 33: Submarket Map of Hamilton Heights

Submarket Map of The Upper West Side from 2004 to 2018

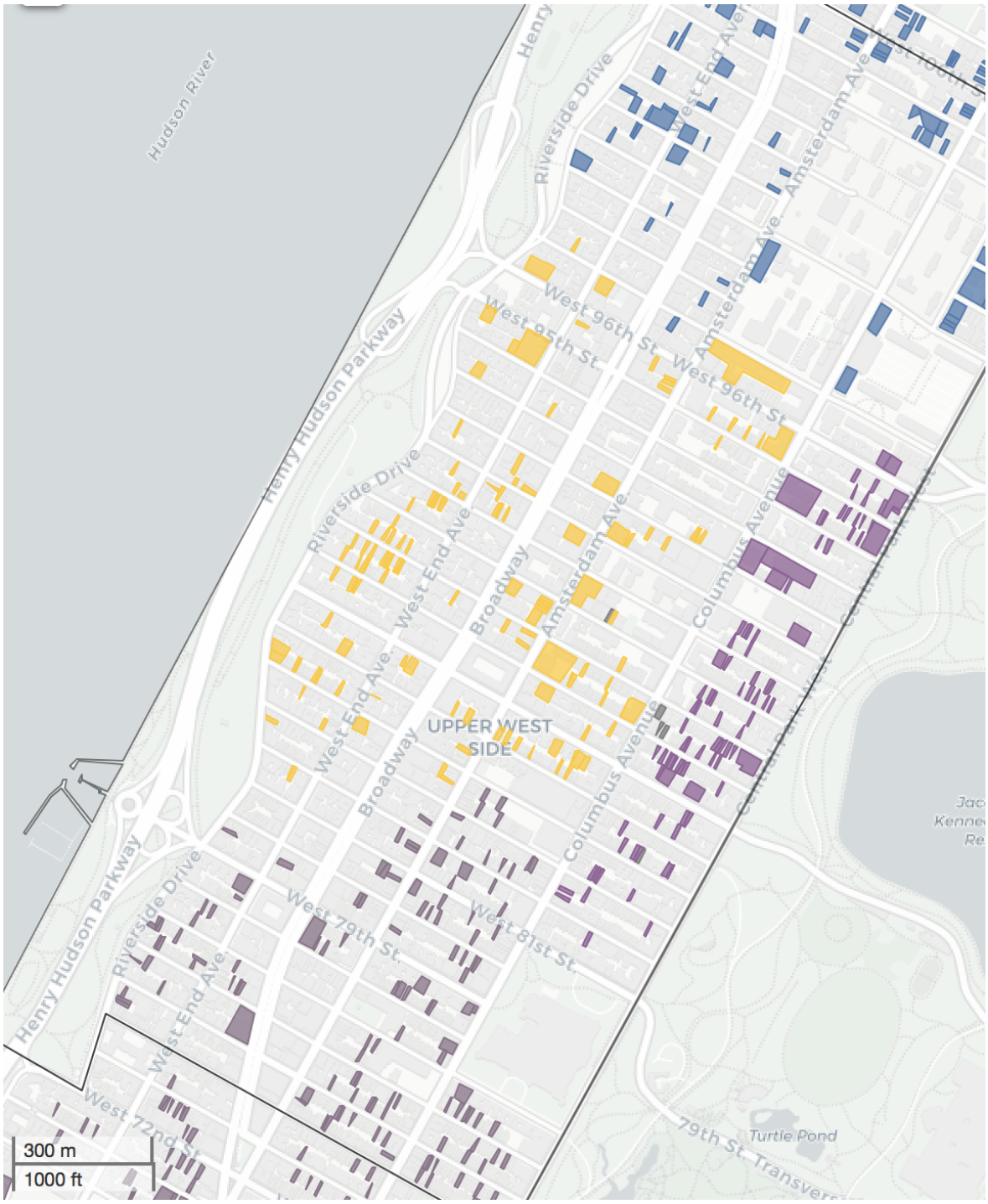

Figure 34: Submarket Map of The Upper West Side

6 Implications & Limitations

A Proxy for Neighborhood Change

6.1 Implications

6.1.1 A Proxy for Neighborhood Change

New York City recognizes 29 different neighborhoods in Manhattan per the NTA data provided by NYC Department of City Planning (2018a). The results of this study found 43 different MFR-CRE submarkets in Manhattan per SCWMHC. This begs the question whether or not the difference between the number of neighborhoods and neighborhood locations relative to the number of submarkets and submarket locations, matter with regard to the neighborhoods themselves and the people within them.

It was speculated that the answer to that question was, yes. The number of MFR-CRE submarkets and submarket locations relative to the number of neighborhoods and neighborhood locations, should matter. It was suspected to matter because different MFR-CRE submarkets are broad indicators of different MFR-CRE prices, as this study has demonstrated. If we accept that MFR-CRE prices are accurate reflections of their fair market values, which have direct affects on the rents that are later sought from tenants by the new owners of those MFR-CRE assets - then, it is possible to imagine that if an owner paid a premium to acquire and / or improve a MFR-CRE asset, the owner would likely seek rents, which correspond to or are correlated with its purchase and sale price and / or improvement cost. This is not always the case, but a generally reliable assumption.

In fact, the relationship between tenant rents / revenues and sales prices is commonly used by subtracting out operating costs from effective gross tenant rent / revenues and dividing the result by the sales price. This is commonly known as the capitalization rate or simply "cap rate" in practice. It is relied upon as a standard metric in the commercial real estate industry (Schmitz 2000). This further suggests that the previous assumption, where fair market values correspond to or are correlated with tenant rents, is a generally reliable one.

It was thought that if the previous scenario happened once (an owner paid a premium to acquire and / or improve a MFR-CRE asset and later sought rents, which correspond to or were correlated with the purchase and sale price and / or improvement cost of that asset), it might likely just appear as noise in the residuals. However, if that scenario were to occur a number of times, in a particular location, over the course of a number of years, a pattern of that phenomena might likely begin to appear in the data. If that pattern occurred enough times, it could further develop into what clustering algorithms like the ones utilized in this study, might likely identify as a cluster. In this particular case, a submarket. This was suspected to have happened in a number of cases.

This study found a strong presence of multiple submarkets fragmented throughout a given neighborhood (see Figure 29: Submarket Map of Central Harlem). This seems to be proximal evidence of an important phenomena that cannot be understated. It appears that neighborhoods, which contained an obvious presence of multiple submarkets, experienced MFR-CRE value *divergence* while at the same time spatial *convergence*. In other words, the stability of MFR-CRE values in a given neighborhood might be roughly equivalent to the number of submarkets contained within its boundary. It was suspected that the more submarkets contained within a discrete neighborhood boundary, the less stable MFR-CRE values might be. This phenomena is important to consider because it might likely indicate future changes to tenant rents (if we accept the previous scenario where MFR-CRE values correspond to or

are correlated with tenant rents). Tenant rents are important, as they can be generally thought of a proxy related to neighborhood change. More directly, they are a part of a broader public discourse regarding gentrification and displacement.

It is possible that the submarket models in this study have the potential to the improve the explanatory precision of the spatiality of this discourse. Visualizing submarkets on an easily interpretable map, could help advanced this much broader and complex discussion with some additional degree of empiricism. It was hoped that the spatial specificity of submarket modeling, as demonstrated in this study, could be useful for advancing this discussion in a more nuanced way than before. The details of a neighborhood scale map could help to enrich the ways we view neighborhoods that is both theoretically compelling and practically useful in ways that the areal unit aggregation and boundary drawing, like the methods found in the previous submarket studies summarized in *Chapter 2 - Background & Theory* are not.

6.1.2 Why Not Areal Unit Aggregation?

A more direct example of why existing methods of areal unit aggregation might not be appropriate for describing submarket phenomena at the neighborhood scale was demonstrated through a similar approach taken by the United States Department of Housing & Urban Development (HUD) for determining Fair Market Rents (FMR). As briefly mentioned in *Chapter 3 - Study Area & Data* the geography of FMR's in Manhattan and throughout New York were determined by and direct equivalents of their respective Zip Code Tabulation Areas (ZCTA) (HUD Office of Policy Development and Research 2019a). This is in contrast to how FMR are traditionally determined on a city-wide basis or Core-Based Statistical Areas (CBSA) (HUD Office of Policy Development and Research 2019a). Determining FMR's by ZCTA's is referred to by HUD as Small Area Fair Market Rents (SAFMR) (HUD Office of Policy Development and Research 2019b). Although SAFMR's are calculated at a higher spatial resolution than FMR's and are likely to be more accurate reflections of the market geography of tenant rents when compared to CBSA's overall, it is believed that the geographic boundaries of SAFMR's are still too generalized and do not provide enough spatial specificity in order to appropriately represent the underlying phenomena to be useful within the boundary of a neighborhood.

The following demonstrates this issue by exhibiting the same series of maps previously exhibited in *Chapter 5 - Results & Discussion*. However, the following SAFMR submarket maps are illustrated by the geographic boundaries of SAFMR's, compared with the same NTA boundaries found in the submarket modeling maps. The SAFMR submarket maps *should* be interpreted as a basis of visual comparison when compared to the previous submarket modeling maps. They *should not* be interpreted as a viable alternative meant to be fully explored by this study. By examining the following SAFMR maps, it is possible to both visually infer, as well as conceptually understand that *utilizing the geographic boundaries of SAFMR for areal unit aggregation does not attempt to model the geographic phenomena of submarkets explicitly, but rather it necessarily imposes it.* If this study were to rely on the geographic boundaries of SAFMR's for areal unit aggregation in an effort to identify submarkets, it would likely not benefit from additional insight into the phenomena. In fact, relying solely on areal unit aggregation, would have answered the research question *"How many submarkets are there in Manhattan?"* prior to even analyzing it. Again, this is distinctly different than the submarket models previously demonstrated in this study and has been included here as a basis of comparison. In the following example, there were 69 different SAFMR submarkets in Manhattan; not surprisingly, directly equivalent to the number of individual ZCTA's.

Small Area Fair Market Rent Map of Manhattan in 2018

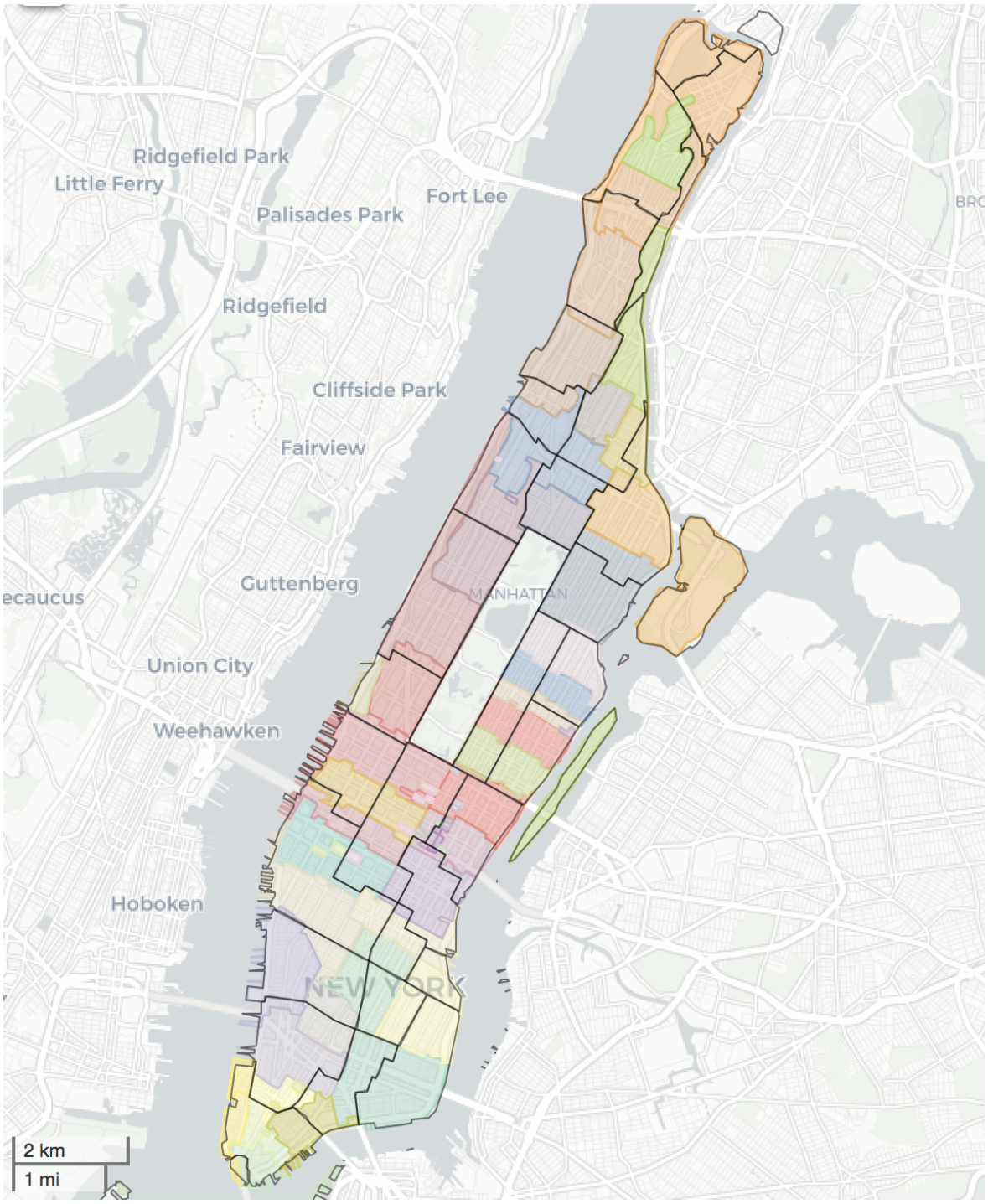

Figure 35: Small Area Fair Market Rent Map of Manhattan

Small Area Fair Market Rent Map of Central Harlem in 2018

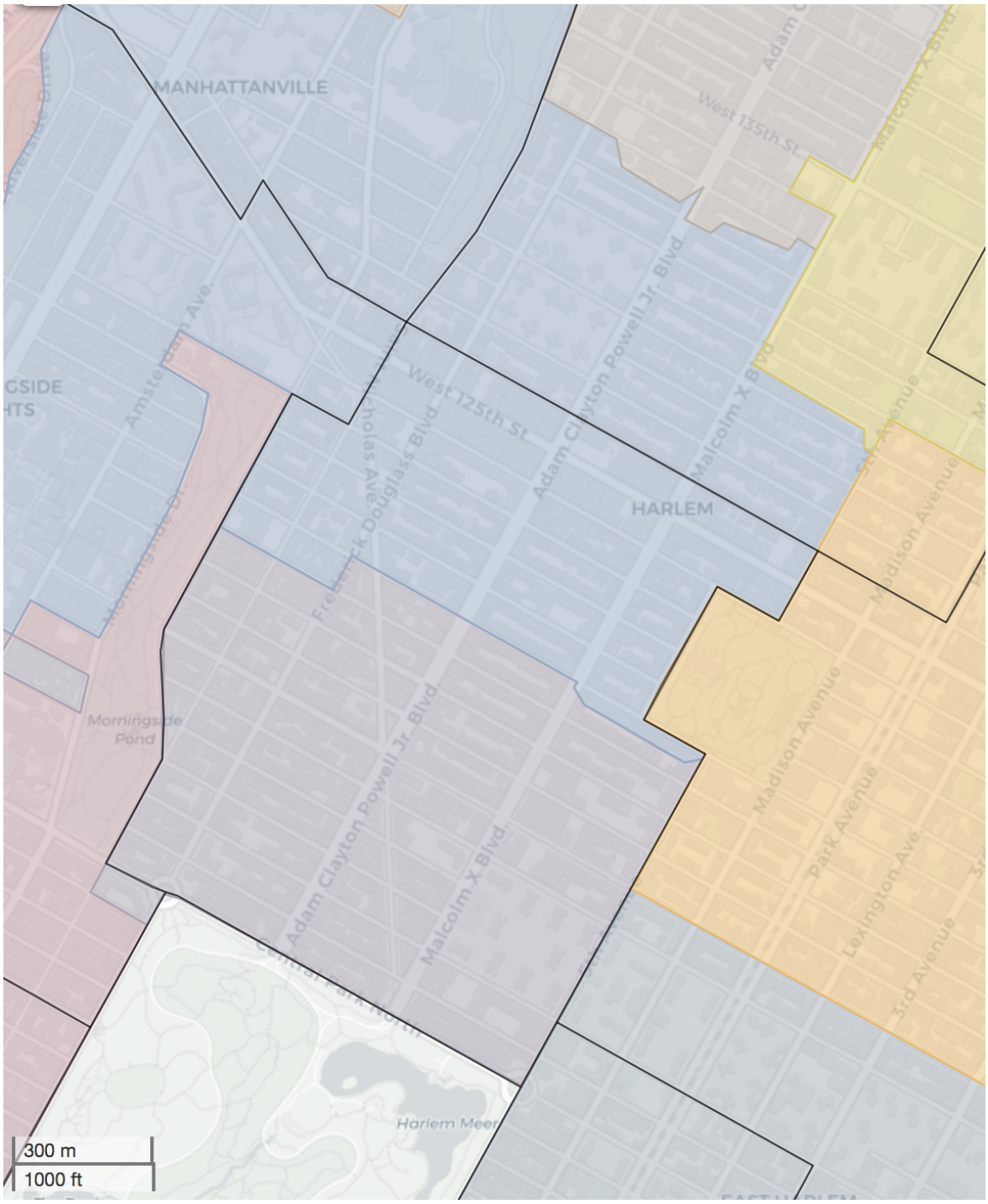

Figure 36: Small Area Fair Market Rent Map of Central Harlem

Small Area Fair Market Rent Map of Chelsea in 2018

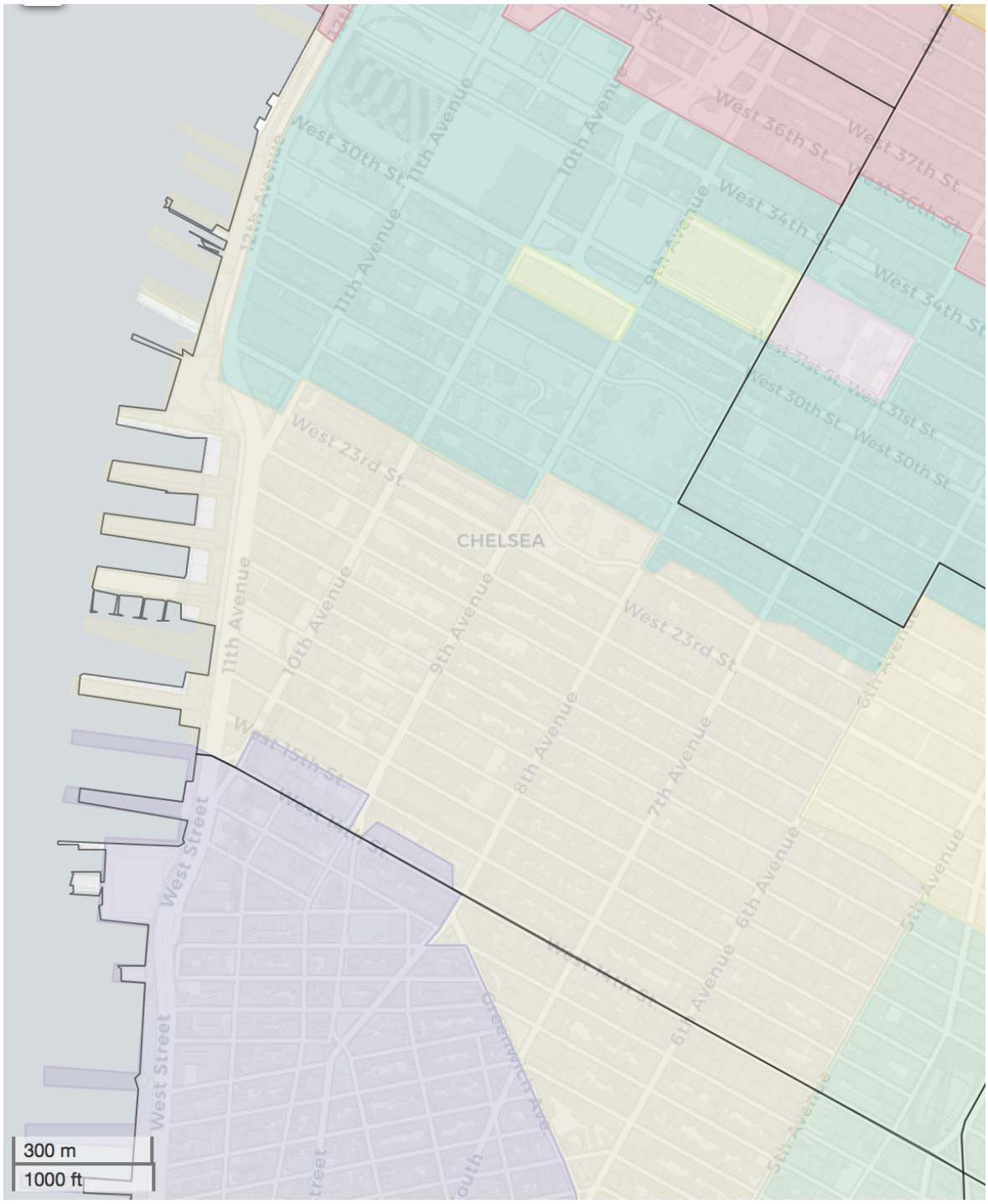

Figure 37: Small Area Fair Market Rent Map of Chelsea

Small Area Fair Market Rent Map of East Harlem in 2018

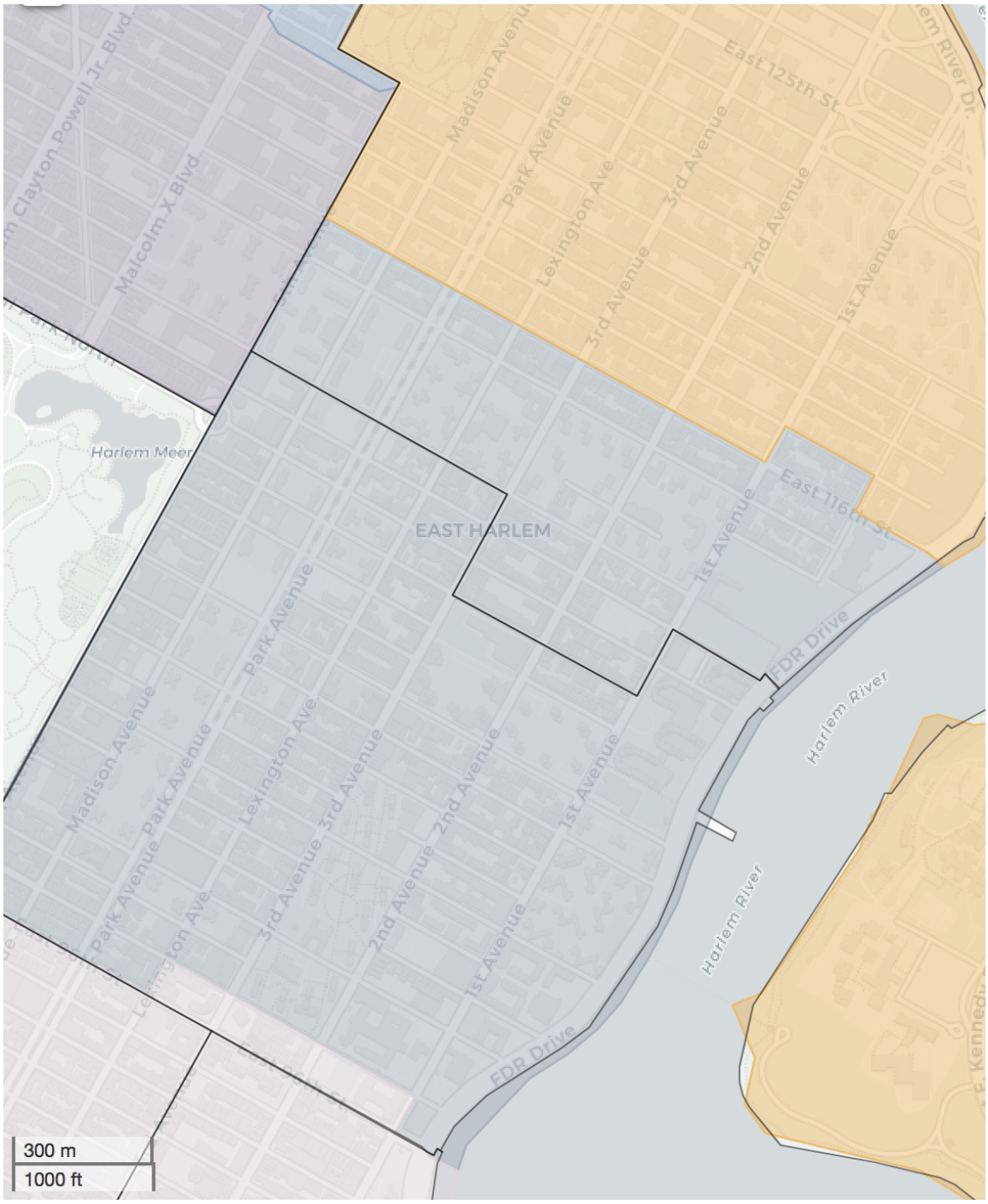

Figure 38: Small Area Fair Market Rent Map of East Harlem

Small Area Fair Market Rent Map of Hamilton Heights in 2018

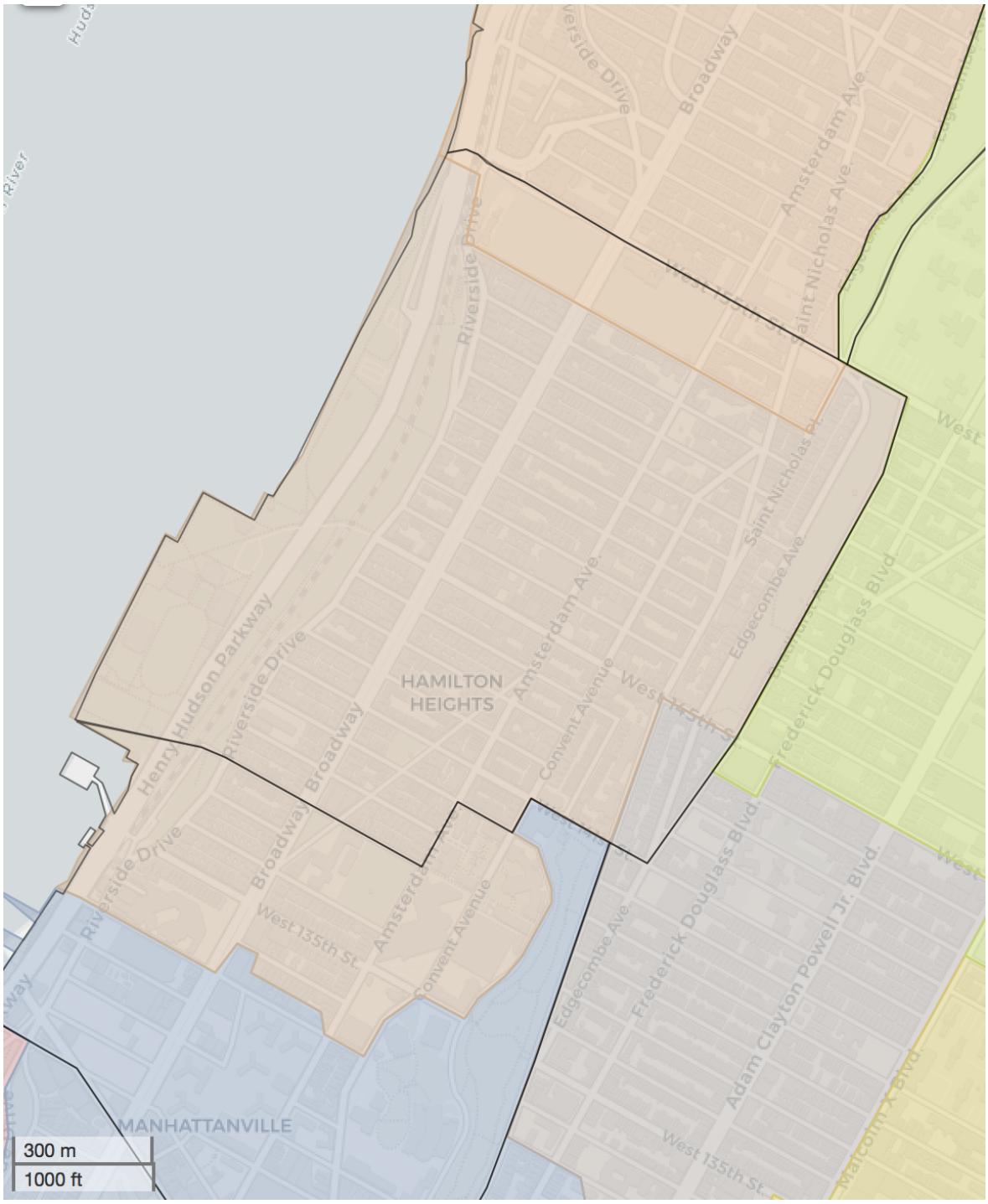

Figure 39: Small Area Fair Market Rent Map of Hamilton Heights

Small Area Fair Market Rent Map of The Upper West Side in 2018

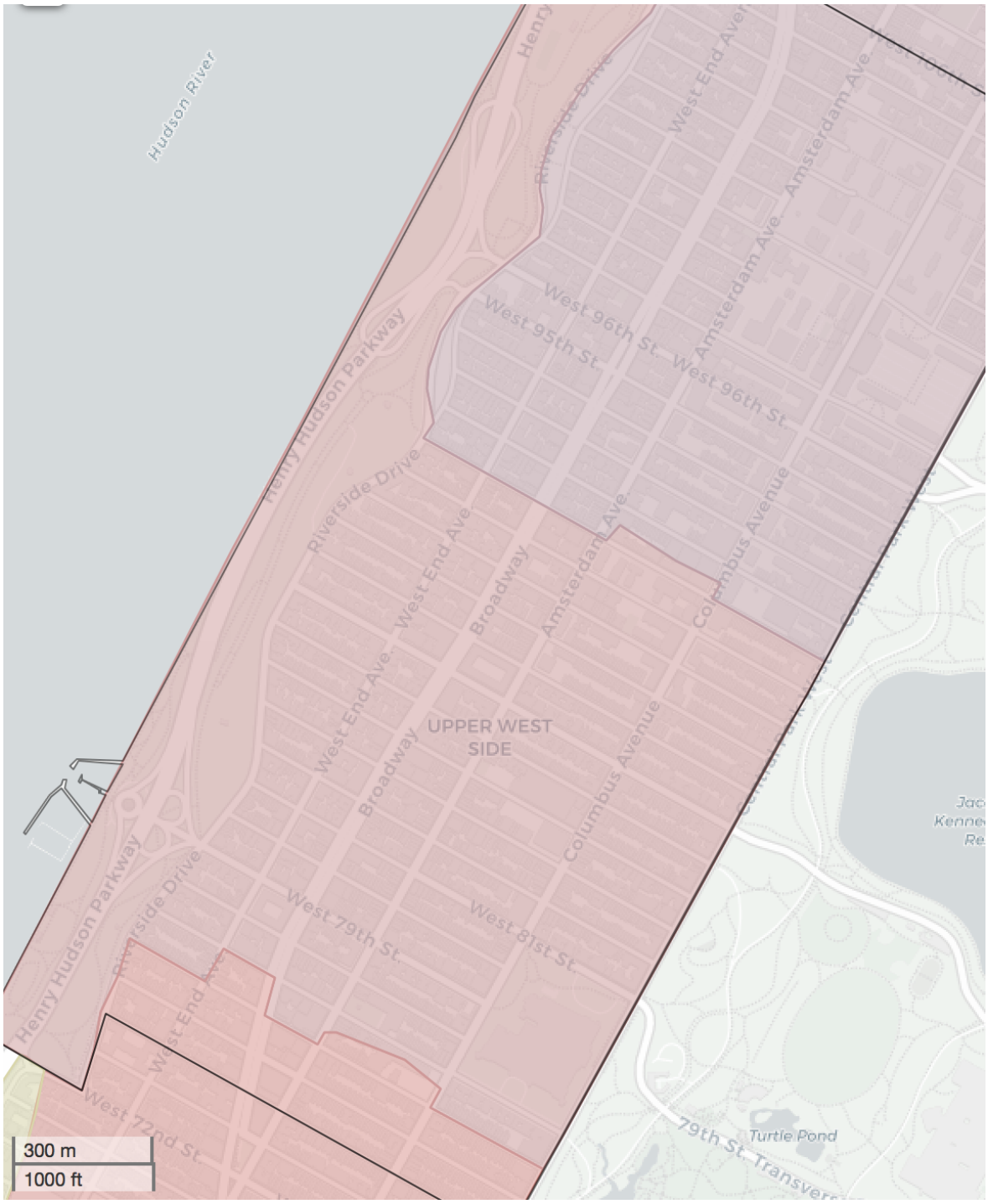

Figure 40: Submarket Map of The Upper West Side

6.1.3 Augmented Geography of Small Area Fair Market Rents

One of the ways planning & policy could directly benefit from the submarket models contained in this study, is their possible application in augmenting the geography of SAFMR's. It is believed that the location and spatial extents SAFMR's could be parametrically adjusted to better fit submarket models, rather than simply relying on ZCTA's for determining the market geography of local tenant rents. Submarket models might likely be better representations of observed tenant rents, especially considering their rough correlation to MFR-CRE sale prices, as demonstrated by the common use of cap rates.

Reestablishing both the location and spatial extents of SAFMR's by accounting for submarket models, could lead to higher correlations to actual tenant rents. If the augmented geography of SAFMR's was found to be more consistent with observed tenant rents, then a more efficient allocation of resources for existing HUD programs might be possible. The programs, which would likely realize the greatest benefit, would be the ones that rely on the SAFMR calculation. For example, the payment standard for the Section 8 Vouchers and initial rents for Housing Assistance Payment contracts, as well as rent ceilings for residential units in the HOME Investment program, all rely on SAFMR's (HUD Office of Policy Development and Research 2019b). Although the positive affect of augmenting the geography of SAFMR's by submarket modeling is speculative and the outcome unpredictable, it is worth considering the ways in which submarket models can better inform existing planning & policy at the neighborhood scale.

One way that the submarket models contained in this study could be used to augment the geography of SAFMR's, is with the use of vornoi diagrams. The application of voronoi diagrams with a novel distance function calculated by street network, might be useful to consider. The polygons produced from a voronoi diagram using street network distance between different, but adjacent submarkets, might likely appear as natural breaks in the urban fabric and perhaps result in relative consistency with land lot parcel boundaries, rather than by strictly creating voronoi diagrams through euclidean space, which tend to result in poor representations of the underlying urban form. The voronoi diagrams generated by a custom distance function could then be spatial joined to the land lot polygons to generate sensible polygonal boundaries around submarket clusters. This is just one speculative idea that a future study, design, plan, or policy could begin to explore.

Another way that the submarket models contained in this study could be used to augment the geography of SAFMR's, is with the amalgamation of both map outputs (submarket models and SAFMR areal unit aggregation) into a single composite map. The composite maps generated here illustrated both the submarket models and the SAFMR areal unit aggregation together with the NTA boundaries. This was done in an effort to begin to reimagine how the existing SAFMR boundaries could be determined, while directly considering the findings of this study. Although an augmented SAFMR boundary was not explicitly proposed due to the limits of this study, the composite maps are designed to both compare the final map output results, as well as help imagine new ways to augment the existing geography of SAFMR's, which has yet to be considered.

It is also important to reiterate that the implications outlined in this chapter were not intended to be conclusive or perhaps even literal. Rather, they were intended to be somewhat imaginative and speculative. They ought to be regarded as important limitations that serve as the basis for future investigation. There were certainly other limitations of this study; many of which were previously outlined. However, the ones that were not, were generally broader, and reserved for the following section, *Limitations*.

Composite Map of Manhattan

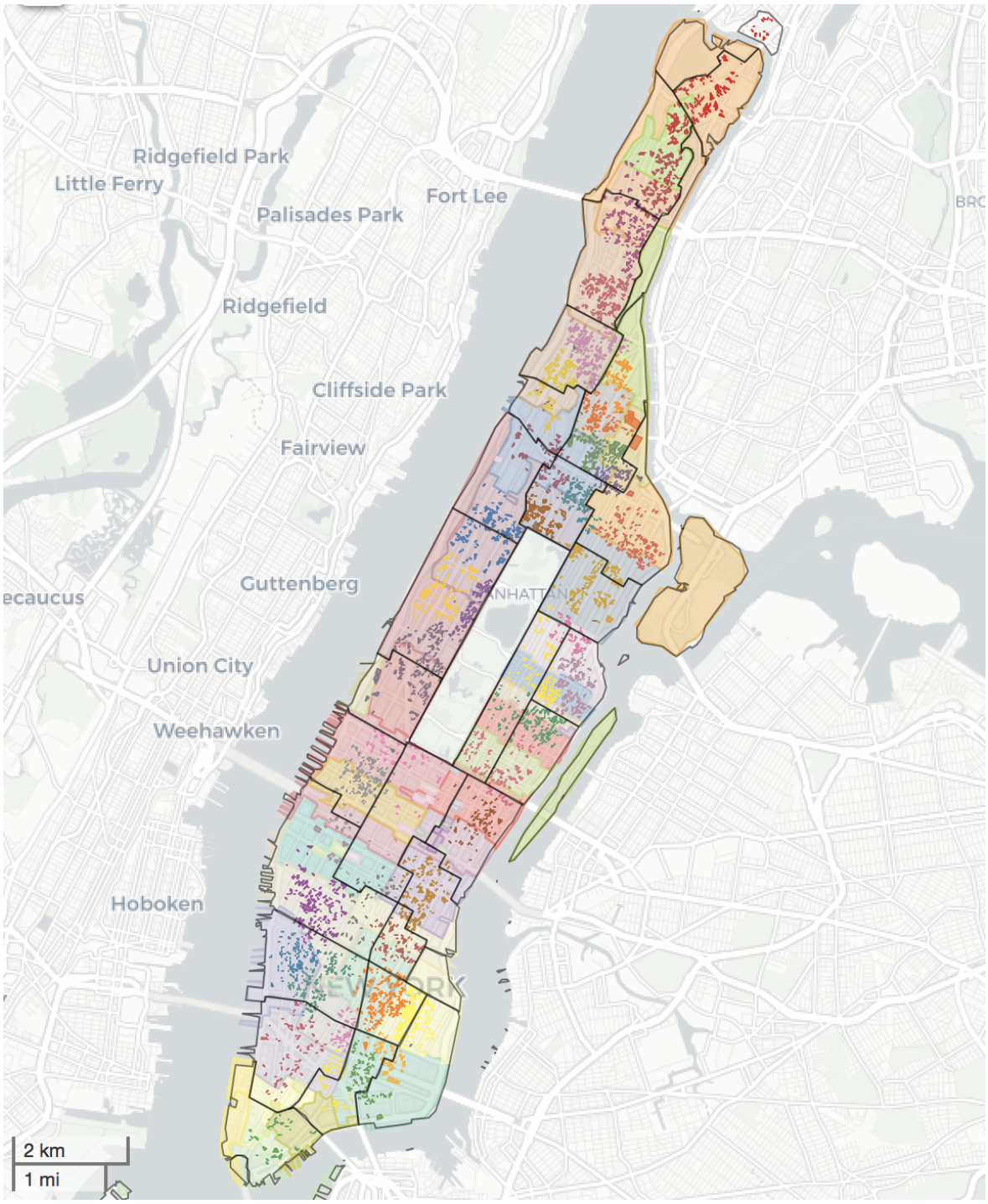

Figure 41: Composite Map of Manhattan

Composite Map of Central Harlem

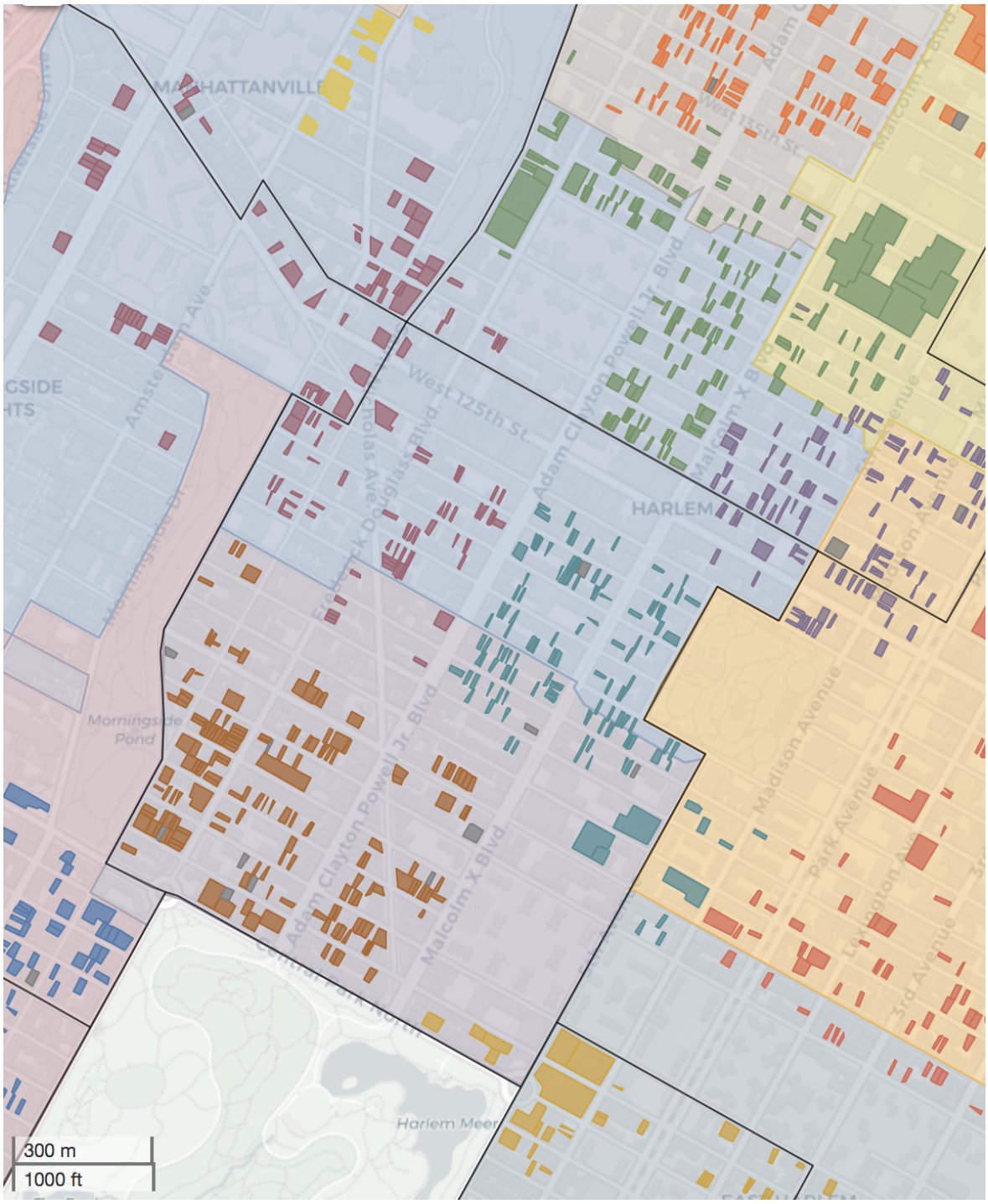

Figure 42: Composite Map of Central Harlem

Composite Map of Chelsea

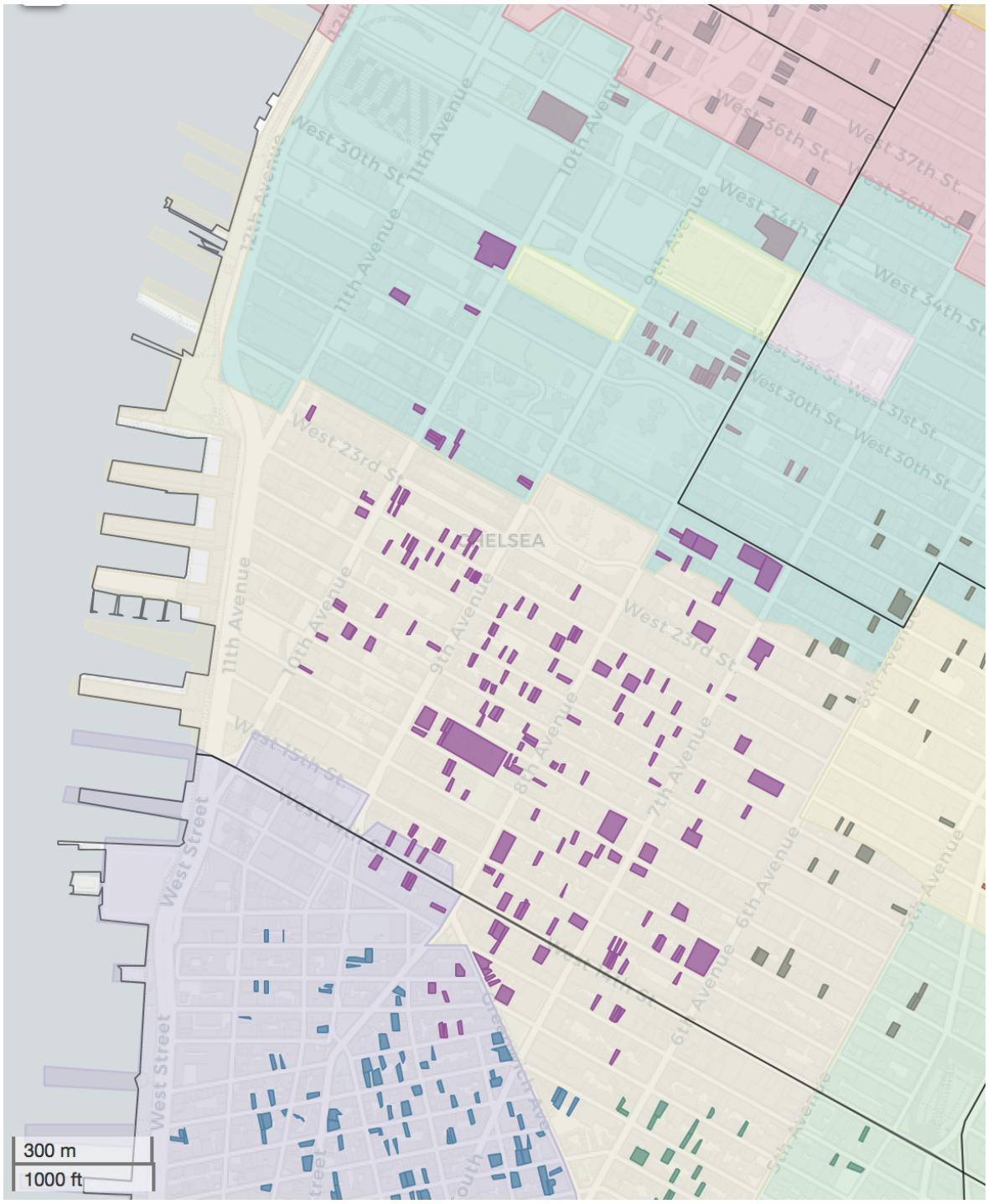

Figure 43: Composite Map of Chelsea

Composite Map of East Harlem

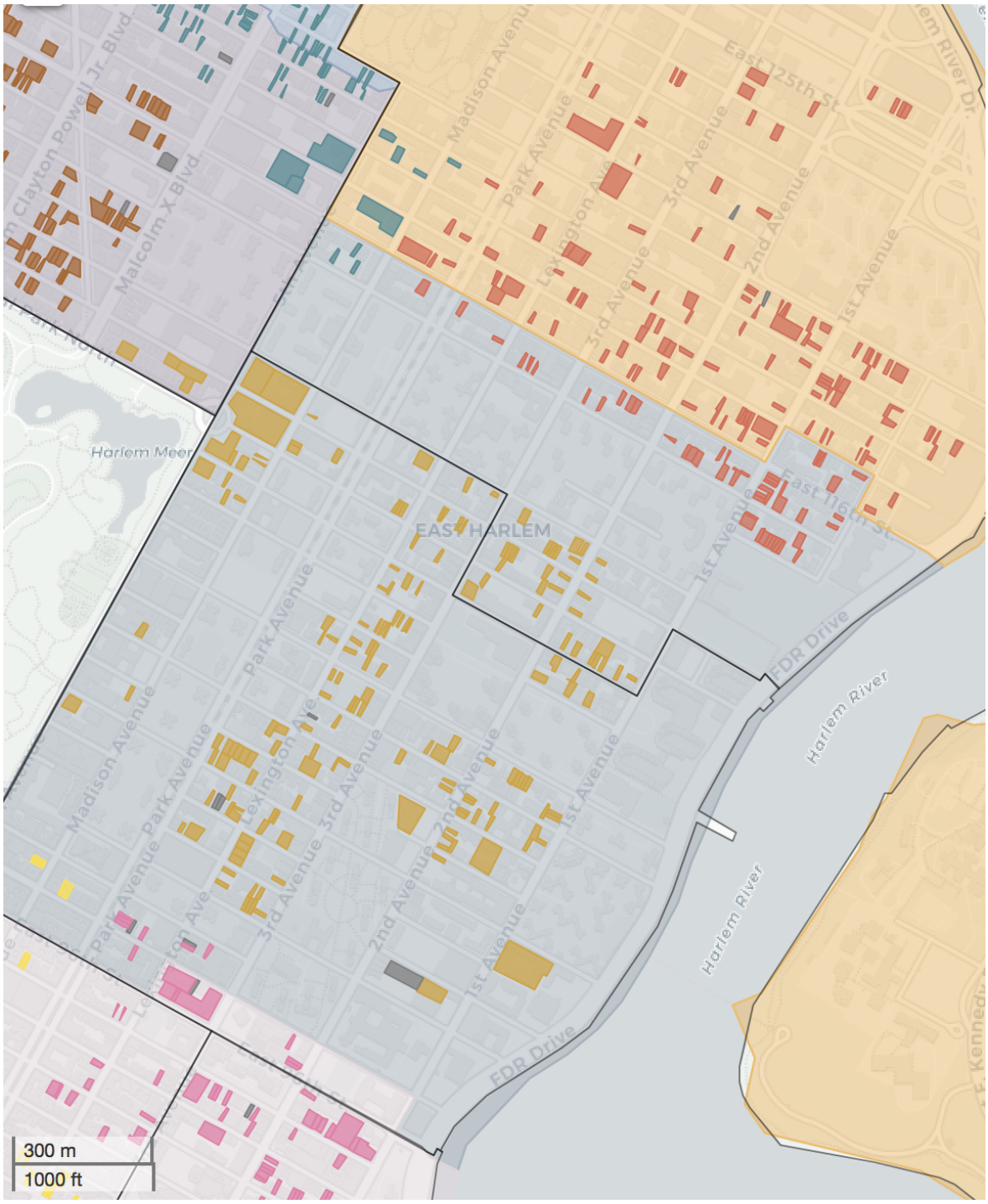

Figure 44: Composite Map of East Harlem

Composite Map of Hamilton Heights

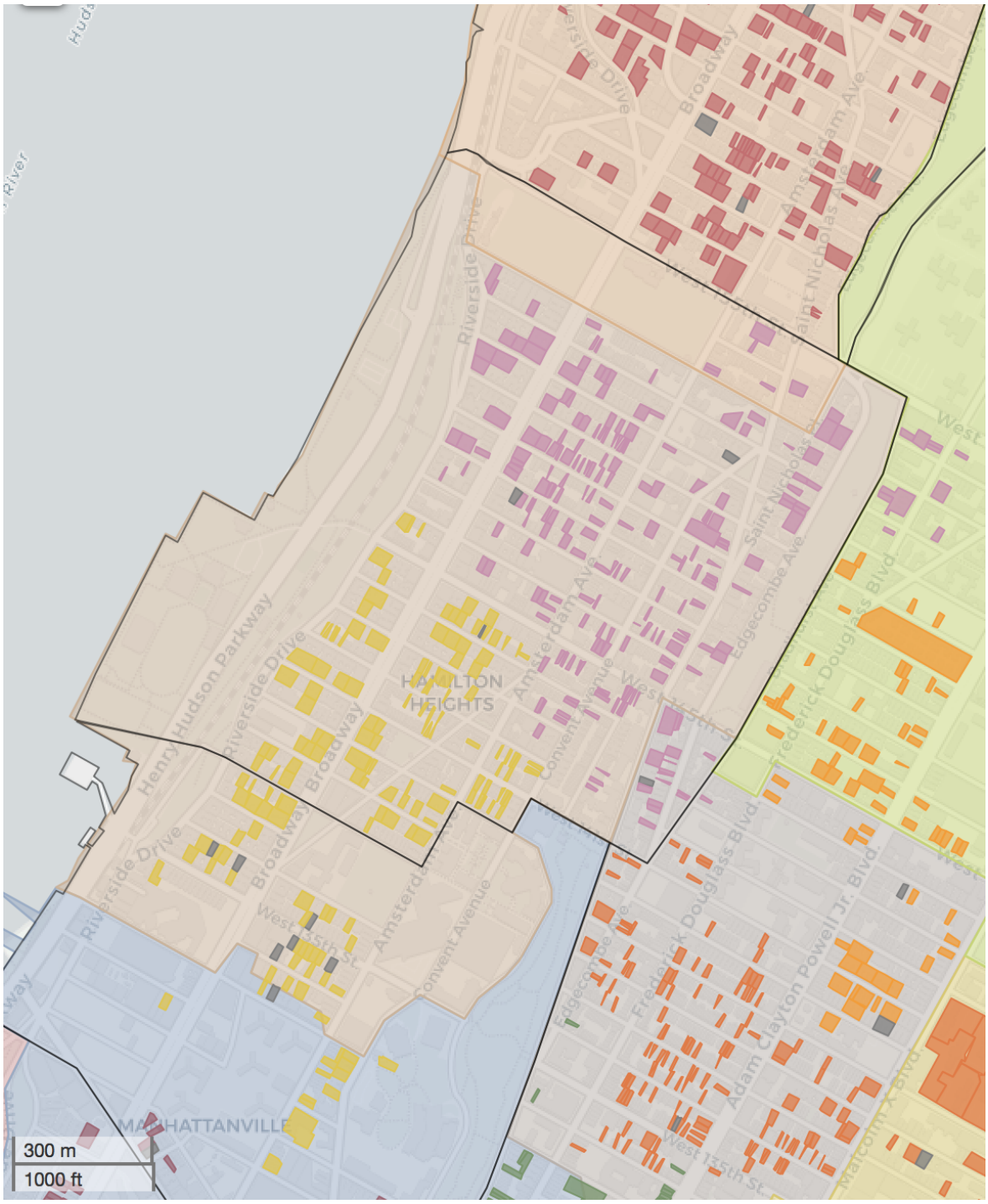

Figure 45: Composite Map of Hamilton Heights

Composite Map of The Upper West Side

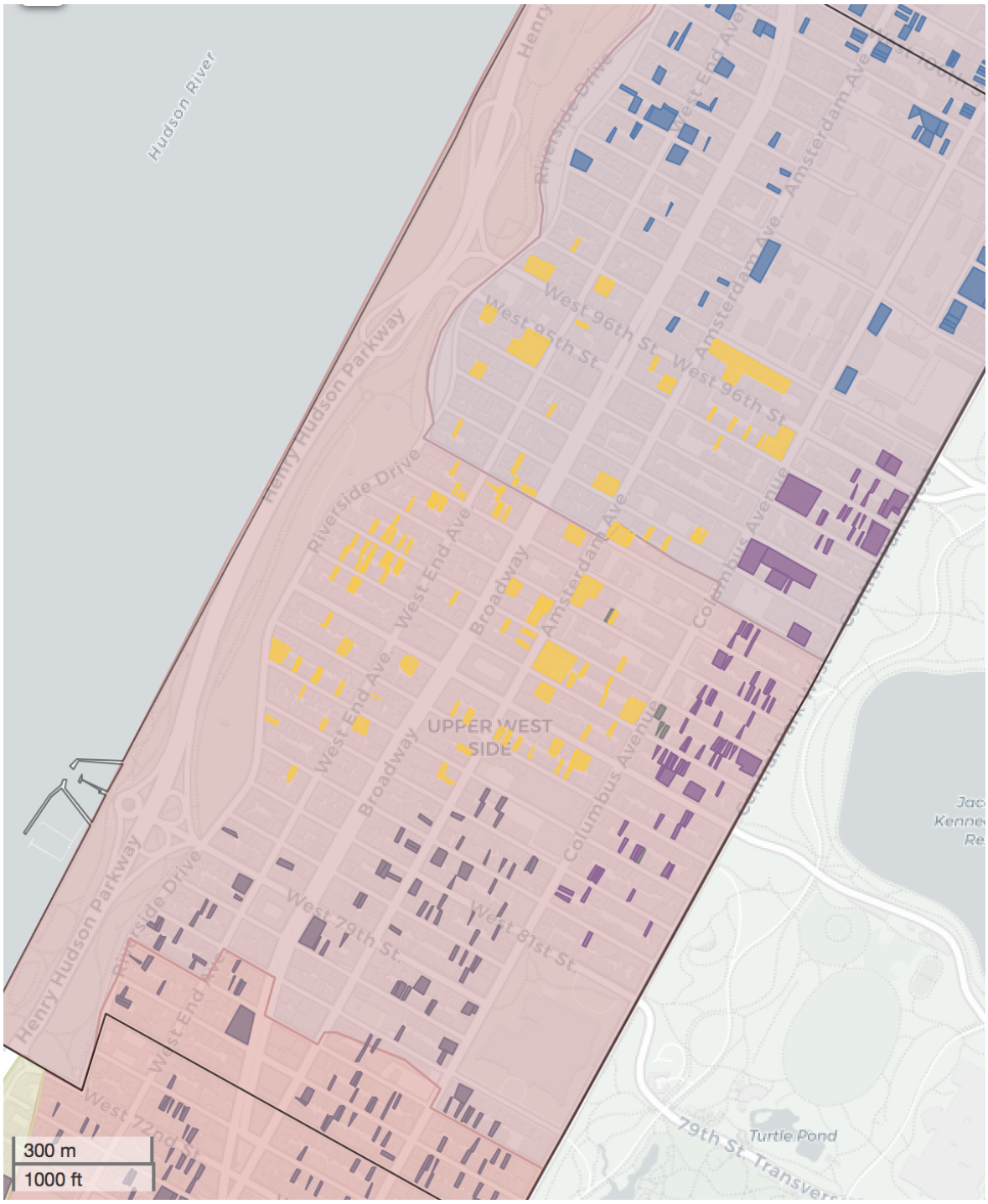

Figure 46: Composite Map of The Upper West Side

6.2 Limitations

Although this study attempted to engage most of what was thought to be important with regard to submarkets and their relationship to neighborhoods, there were still other important and much broader limitations worth mentioning. There were three main types of limitations outlined here: theoretical limitations, empirical limitations, and limitations regarding validation.

6.2.1 Theoretical Limitations

An important theoretical limitation of this study was its reliance on submarket theory. As explained in *Chapter 2 - Background & Theory*, the unitary market theory has largely been supplanted by submarket theory. However, there is still the possibility that the unitary market theory may later be favored or found to be more valid. In that case, this study would largely suffer from defending its empirical findings. This study requires the assumption that submarkets exists or at least that they are a useful ontological prior, which can then be empirically operationalized. The theoretical argument against submarkets is a limitation worth considering. However, it does not necessarily mean that the findings in this study should be dismissed. There is a general consensus amongst researchers, practitioners, and even the general public that the submarket theory is a valid one.

6.2.2 Empirical Limitations

An important empirical limitation of this study was its reliance on prediction. Recall in *Chapter 4 - Methodology* that the clustering results from SCWMHC largely relied on predicted values of five matrices originally containing roughly 91% missingness. The quality of the final clustering results, largely depended on the predictions made in the prior step. In other words, if the prediction results from NNMF were found to be invalid or unreliable, then it would be unreasonable to attempt to interpret the clustering results from SCWMHC.

Although the reliance on prediction was a limitation of this study, care was taken to reasonably maximize the quality of the NNMF results without overly focusing on raw prediction. It was thought that focusing too heavily on NNMF or other imputation methods for sparse matrices, would draw away from the main focus of this study. A balance was attempted to be reached between NNMF and SCWMHC, which necessarily drew away from purely focusing on optimizing the prediction models for the five sparse matrices (variables).

Although more attention could have been dedicated to optimizing predictive performance in this regard, the prediction quality of NNMF was still taken seriously. Recall in *Chapter 4 - Methodology* the introduction of the novel conditional random sampling technique, an empirical test for algorithm selection and convergence speed, the optimal low k rank matrix exhaustively tested using LOOCV, and the results contextualized with other sophisticated prediction models for sparse matrices. Perhaps most importantly, the predictive accuracy of all NNMF models were validated. Although there still could be technical arguments made against this approach. It was believed that the prediction results were sufficient enough to answer the research questions asked in this study.

6.2.3 Validation Limitations

Another important limitation of this study was that the submarket models were not validated in the same way traditionally found in the literature. As mentioned in *Chapter 2 - Background & Theory*, submarket modeling orthodoxy relied on validating the results of their studies by the marginal increase in the predictive performance of hedonic modeling before and after controlling for submarket boundaries. This study did not validate its submarket models in the same way traditionally found in the literature.

Recall in *Chapter 5 - Results & Discussion* that the results of this study were validated by an objective criteria, which measured the maximum absolute value of the difference between the mean intra-cluster correlation (average within cluster similarity) and the maximum inter-cluster correlation (maximum outside of cluster dissimilarity), where the correlation range was specified from 0 to 1 (absolute value basis). The results were validated in this way because the purpose of the study was designed to serve different ends than most, if not all of the existing literature regarding submarkets.

Recall that the purpose statement in *Chapter 1 - Introduction*, was to attempt to reduce the spatial complexity of real estate market phenomena in a way that could be useful for planning & policy at the neighborhood scale. Therefore, it was thought that validating the quality of the submarket models by the marginal increase in the predictive performance of prices was inappropriate for that objective. However, what this study did find appropriate was to reduce the spatial complexity of real estate market phenomena by maximizing the homogeneity of values, as well as the relationship between those values and the underlying structural characteristics of the assets (Price / X variables), while at the same time maximizing spatial contiguity and compactness.

In addition, the results of this study were further visually validated by exhibiting the spatial contrast between the number of submarkets and submarket locations against the number of neighborhoods and neighborhood locations. This additional validation step was largely to exhibit the results in a way that made sense to planning, policy, and the public. It was also validated this way to serve the purpose of raising additional questions, advance the discussion, and encourage further research. The limitations of this study regarding validation, could largely benefit from rigorous spatial analysis. It is also important to remember that a traditional unsupervised learning setting, such as the clustering one found in this study, does not always require validation and can be used for developing more nuanced hypothesis for further testing.

7 Conclusion

Summary & Future Recommendations

7.1 Summary

This study focused on submarket modeling with unsupervised learning and geographic information system fundamentals to better understand urbanism at the neighborhood scale. The study trained a Spatially Constrained Weighted-Multivariate Hierarchical Clustering algorithm to identify the optimal number of Multifamily Residential Commercial Real Estate submarkets in Manhattan, New York from and including 2004 to 2018. It supported the notion that submarket models were best approached as a spatially dependent classification problem, rather than an areal unit aggregation one. The study suggested that by assigning submarket identifications to individual properties, which exhibited both multivariate price similarity and spatial contiguity, were more effective submarket models for understanding neighborhood change at the neighborhood scale. Furthermore, that in order to effectively and responsibly proceed in implementing planning & policy interventions at the neighborhood scale, it is useful to consider them.

The methodology applied in this study utilized Non-Negative Matrix Factorization for predicting the annual normalized values of every multifamily residential commercial property that had been transacted in Manhattan from and included 2004 to 2018. A conditional random sampling technique for train and test set splitting sparse matrices was introduced. The methodology included Leave One Out Cross Validation for estimating the optimal low rank matrix and compared Non-Negative Matrix Factorization with other imputation methods for sparse matrices. The observed and predicted Multifamily Residential Commercial Real Estate values were then clustered on a weighted basis using the Spatially Constrained Weighted-Multivariate Hierarchical Clustering algorithm with a high spatial contiguity constraint. Five different linkage methods were explored and Ward's Method was selected. The resulting 43 clusters were then used for identifying submarkets and their respective geolocations were spatial joined to the intersecting land lot polygons for geographic representation.

7.2 Future Recommendations

It is believed that unsupervised learning and machine learning more broadly, have other useful and important applications in the field of urbanism that are worth exploring. Unsupervised learning can tell us things about our cities that would not otherwise be possible or obvious. It can reveal patterns in urban data that other methods cannot. Even when reasonable hypothesis can be tested with rigorous analysis and statistical inference, those methods are still less equipped to reveal the types of information that unsupervised learning is designed for.

Unsupervised learning can help to answer important questions regarding our cities, but perhaps even more importantly, it can help to *ask* them. It engenders new and interesting questions for planning & policy. Although it is obvious that other technical fields such as computer science, engineering, and data science are more capable of directly engaging unsupervised learning, it is believed that planning & policy have a more important role to play in applying those methods, as they related to cities and the people who inhabit them. Without planning & policy to intervene in asking the right questions, it is a concern that such powerful methods of inquire could be applied in ways that other technical fields are less equipped to understand the social, economic, and political implications of doing so.

Although this study focused on urban real estate economics and optimizing submarket models for applications in planning & policy at the neighborhood scale, the same or similar methods could be applied to a wide range of urban phenomena. Other fields such as signal processing, bioinformatics, and finance, already rely on unsupervised learning to reveal useful patterns underlying their data. It is worth exploring what other ways unsupervised learning and machine learning more broadly, can be applied to urban data to help guide our understanding of cities and make them better places for everyone.

Bibliography

Adair, A. S., J. N. Berry, and W. S. McGreal. 1996. "Hedonic Modelling, Housing Submarkets and Residential Valuation." *Journal of Property Research*. https://doi.org/10.1080/095999196368899.

Badr, Hamada S., Benjamin F. Zaitchik, and Amin K. Dezfuli. 2015. "A Tool for Hierarchical Climate Regionalization." *Earth Science Informatics*. https://doi.org/10.1007/s12145-015-0221-7.

Baffes, J. 2006. "Some Further Evidence on the Law of One Price: The Law of One Price Still Holds." *American Journal of Agricultural Economics*. https://doi.org/10.2307/1242454.

Baruch College City University of New York. 2016. "2010 New York City Zip Code Tabulation Areas (ZCTA)." Newman Library, Baruch College, City University of New York. http://hdl.handle.net/2451/34509.

Bourassa, Steven C., Foort Hamelink, Martin Hoesli, and Bryan D. Macgregor. 1999. "Defining Housing Submarkets." *Journal of Housing Economics*. https://doi.org/10.1006/jhec.1999.0246.

Buuren, Stef van, and Karin Groothuis-Oudshoorn. 2011. "MICE: Multivariate Imputation by Chained Equations in R." *Journal of Statistical Software*. https://doi.org/10.18637/jss.v045.i03.

Fisher, Ernest M., and Robert M. Fisher. 1954. *Urban Real Estate*.

Franc, Vojtěch, Václav Hlaváč, and Mirko Navara. 2005. "Sequential Coordinate-Wise Algorithm for the Non-Negative Least Squares Problem." In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/11556121_50.

Galster, George. 1996. "William Grigsby and The Analysis of Housing Submarkets and Filtering." *Urban Studies*. https://doi.org/10.1080/0042098966376.

Gareth, James, Witten Daniela, Hastie Trevor, and Tibshirani Rober. 2000. *An Introduction to Statistical Learning with Applications in R*. https://doi.org/10.1007/978-1-4614-7138-7.

Goodman, Allen C. 1981. "Housing Submarkets Within Urban Areas: Definitions and Evidence." *Journal of Regional Science*. https://doi.org/10.1111/j.1467-9787.1981.tb00693.x.

Goodman, Allen C., and Thomas G. Thibodeau. 1998. "Housing Market Segmentation." *Journal of Housing Economics*. https://doi.org/10.1006/jhec.1998.0229.

———. 2003. "Housing Market Segmentation and Hedonic Prediction Accuracy." *Journal of Housing Economics* 12 (3): 181–201. https://doi.org/10.1016/S1051-1377(03)00031-7.

Goodman, Allen C, and Thomas G Thibodeau. 2007. "The Spatial Proximity of Metropolitan Area Housing Submarkets." *Real Estate Economics* 35 (2): 209–32. https://doi.org/10.1111/j.1540-6229.2007.00188.x.

Hahsler, Michael. 2011. "recommenderlab: A Framework for Developing and Testing Recommendation Algorithms." https://cran.r-project.org/web/packages/recommenderlab/vignettes/recommenderlab.pdf.

Handbury, J., and D. E. Weinstein. 2014. "Goods Prices and Availability in Cities." *The Review of Economic Studies*. https://doi.org/10.1093/restud/rdu033.

Hastie, Trevor, Rober Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. https://doi.org/10.1007/978-0-387-84858-7.

HUD Office of Policy Development and Research. 2019a. "Fair Market Rents (FMR)." United States Department of Housing & Urban Development. https://www.huduser.gov/portal/datasets/fmr.html.

———. 2019b. "Small Area Fair Market Rents (SAFMR)." United States Department of Housing & Urban Development. https://www.huduser.gov/portal/datasets/fmr/smallarea/index.html%7B/#%7Dfinal-rulemaking.

Keskin, Berna, and Craig Watkins. 2017. "Defining Spatial Housing Submarkets: Exploring the Case for Expert Delineated Boundaries." *Urban Studies*. https://doi.org/10.1177/0042098015620351.

Koren, Yehuda, Robert Bell, and Chris Volinsky. 2009. "Matrix Factorization Techniques for Recommender Systems." *IEEE Computer Society*. https://doi.org/10.1109/MC.2009.263.

Lee, Daniel D., and H. Sebastian Seung. 1999. "Learning the Parts of Objects by Non-Negative Matrix Factorization." *Nature* 401. https://doi.org/10.1038/44565.

Lin, Xihui, and Paul C Boutros. 2018. "Fast Nonnegative Matrix Factorization and Applications to Pattern Extraction, Deconvolution and Imputation," 1–10. https://doi.org/10.1101/321802.

Miller, Harvey J. 2004. "Tobler's First Law and Spatial Analysis." *Annals of the Association of American Geographers*. https://doi.org/10.1111/j.1467-8306.2004.09402005.x.

Montalbán Pozas, Beatriz, and Francisco Javier Neila González. 2018. "Housing Building Typology Definition in a Historical Area Based on a Case Study: The Valley, Spain." *Cities*. https://doi.org/10.1016/j.cities.2017.07.020.

NCSS. 2019. "Hierarchical Clustering / Dendrograms." https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Hierarchical%7B/_%7DClustering-Dendrograms.pdf.

NYC Department of City Planning. 2018a. "Neighborhood Tabulation Areas (NTA)." New York City OpenData. https://data.cityofnewyork.us/City-Government/Neighborhood-Tabulation-Areas/cpf4-rkhq.

———. 2018b. "Primary Land Use Tax Lot Output (PLUTO)." New York City Dept. of City Planning. https://www1.nyc.gov/site/planning/data-maps/open-data/dwn-pluto-mappluto.page.

NYC Department of Finance. 2018. "Rolling Real Estate Sales Data." New York, New York: New York City Dept. of Finance. https://www1.nyc.gov/site/finance/taxes/property-rolling-sales-data.page.

Podani, János. 2000. "Chapter 5 - Hierarchical Clustering." In *Introduction to the Exploration of Multivariate Biological Data*, 135–74. Leiden, The Netherlands: Backhuys Publishers.

Schmitz, Adrienne. 2000. *Multifamily Housing Development Handbook*. Washington, D.C.: Urban Land Institute.

Schnare, Ann B., and Raymond J. Struyk. 1976. "Segmentation in Urban Housing Markets." *Journal of Urban Economics* 3 (2): 146–66. https://doi.org/10.1016/0094-1190(76)90050-4.

Watkins, Craig A. 2001. "The Definition and Identification of Housing Submarkets." *Environment and Planning A: Economy and Space* 33 (12): 2235–53. https://doi.org/10.1068/a34162.

Whitehead, Christine M. E. 1999. "Urban Housing Markets: Theory and Policy." *Handbook of Regional and Urban Economics* 3: 1559–94. https://doi.org/10.1016/S1574-0080(99)80009-1.

Wu, Changshan, and Rashi Sharma. 2012. "Housing Submarket Classification: The Role of Spatial Contiguity." *Applied Geography*. https://doi.org/10.1016/j.apgeog.2011.08.011.

Yeung, Albert Au. 2010. "Matrix Factorization: A Simple Tutorial and Implementation in Python." http://www.quuxlabs.com.

8 Appendices

8.1 Appendix A: Software Requirements

```
## requirements

## settings
options(warn = -1)      # disable warning messages (0 = on, -1 = off)
options(scipen = 999)   # disable scientific notation

## load libraries

## data handling
library(tidyr)          # data cleaning
library(dplyr)          # data cleaning, selection
library(stringr)        # data cleaning, string whitespace removal
library(data.table)     # data cleaning, rename columns, matrix
library(lubridate)      # data cleaning, date / time
library(ggplot2)        # data viz
library(knitr)          # tables
library(ape)            # data viz
library(png)            # images
library(dendextend)     # data viz
library(pastecs)        # desc stat table

## cloud comuting
library(googleComputeEngineR) # cloud compute

## matrix factorization & imputation
library(mice)           # imputation
library(NNLM)           # matrix factor
library(VIM)            # matrix viz
library(imputeTS)       # random imputation
library(missForest)     # random forest
library(recommenderlab) # funk svd

## hierarchical clustering
library(HiClimR)        # clustering

## geographic info systems
library(rgdal)          # shapefile handling
library(rgeos)          # centroid
library(sp)             # spatial join
library(tmaptools)      # geolocation search
library(leaflet)        # interactive mapping
```

8.2 Appendix B: Data

8.2.1 Real Estate Sales Data Loading & Inspection

```
## load data

## load csv data - real estate sales
file_names_sales <- list.files(path = "Data/sales_data/",  # read file names
                               pattern = "*.csv");  # file type extension

file_path_sales <- file.path(path = "Data/sales_data/",  # read file paths
                             file_names_sales);  # file names

sales <- do.call("rbind",  # load files and merge all rows together
                 lapply(file_path_sales,  # file paths of data
                        read.csv,  # load csv files into data frame
                        skip = 0,  # skip first n observations
                        header = TRUE,  # keep header names
                        check.names = FALSE,  # do not change header names
                        strip.white = TRUE,  # remove whitespace
                        stringsAsFactors = FALSE,  # spec string data type
                        na.strings = c("", "NA")));  # assign na to missing data

## remove components
rm(file, file_names_sales, file_path_sales)
```

Dimensions

```
# view data frame dimensions
dim(sales)
```

List of Variables

```
# view feat names
names(sales)
```

8.2.2 Primary Land Use Tax Lot Output Data Loading & Inspection

```
## load shapefile - land lots - wgs84 projection
pluto_shape <- readOGR("Data/mn_pluto_shape")
pluto_shape <- spTransform(pluto_shape, CRS("+proj=longlat +ellps=WGS84"))
```

Dimensions

```
# view data frame dimensions
dim(pluto_shape@data)
```

List of Variables

```
# view feat names
names(pluto_shape@data)
```

8.2.3 Zip Code Tabulation Areas Data Loading & Inspection

```
## load shapefile - zip code tabulation areas (zcta) - wgs84 projection
zcta_shape <- readOGR("Data/zcta_shape")  # load data
zcta_shape <- subset(zcta_shape, bcode == "36061")  # subset only manhattan
zcta_shape <- spTransform(zcta_shape, CRS(proj4string(pluto_shape)))  # reproject
```

Dimensions

```
# view data frame dimensions
dim(zcta_shape@data)
```

List of Variables

```
# view feat names
names(zcta_shape@data)
```

8.2.4 Neighborhood Tabulation Areas Data Loading & Inspection

```
## load shapefile - neighborhood tabulation areas - wgs84 projection
nta_shape <- readOGR("Data/nta_shape")
nta_shape <- spTransform(nta_shape, CRS(proj4string(pluto_shape)))
nta_shape <- subset(nta_shape, boro_name == "Manhattan") # subset only manhattan
```

Dimensions

```
# view data frame dimensions
dim(nta_shape@data)
```

Variables

```
# view feat names
names(nta_shape@data)
```

8.3 Appendix C: Standard Data Handling Procedures

8.3.1 Data Cleaning Implementation

```
## clean data - real estate sales

## remove duplicate observations
sales <- sales[!duplicated(sales), ]

## remove spaces from header names
colnames(sales) <- gsub(" ", "", colnames(sales))

## remove $ sign for observations in SALEPRICE feature
sales$SALEPRICE <- gsub("[\\$,]", "", sales$SALEPRICE)

## remove "," for observations in LANDSQUAREFEET feature
sales$LANDSQUAREFEET <- gsub(",", "", sales$LANDSQUAREFEET)

## remove "," for observations in GROSSSQUAREFEET feature
sales$GROSSSQUAREFEET <- gsub(",", "", sales$GROSSSQUAREFEET)

## remove leading & trailing whitespaces
sales$BUILDINGCLASSCATEGORY <- str_trim(sales$BUILDINGCLASSCATEGORY)

## remove within string whitespaces
sales$BUILDINGCLASSCATEGORY <- str_squish(sales$BUILDINGCLASSCATEGORY)

## remove leading & trailing whitespaces
sales$ADDRESS <- str_trim(sales$ADDRESS)

## remove within string whitespaces
sales$ADDRESS <- str_squish(sales$ADDRESS)

## remove number ordinals
sales$ADDRESS <- gsub("(\\d)(ST|ND|RD|TH)\\b", "\\1", sales$ADDRESS)
```

8.3.2 Feature Selection Implementation

```
## feature selection - real estate sales

## remove features not applicable, select the following
sales <- sales[c("BUILDINGCLASSCATEGORY",
                 "ADDRESS", "ZIPCODE",
                 "RESIDENTIALUNITS", "COMMERCIALUNITS", "TOTALUNITS",
                 "GROSSSQUAREFEET", "LANDSQUAREFEET",
                 "SALEPRICE", "SALEDATE")]

## feature selection - land lots

## remove features not applicable and redundate select the following
pluto_shape@data <- pluto_shape@data[c("Address", "ZipCode")]
```

8.3.3 Data Type Correction Implementation

```
## data types - real estate sales

## convert data types
sales$ADDRESS <- as.character(sales$ADDRESS)
sales$ZIPCODE <- as.character(sales$ZIPCODE)
sales$RESIDENTIALUNITS <- as.numeric(as.character(sales$RESIDENTIALUNITS))
sales$COMMERCIALUNITS <- as.numeric(as.character(sales$COMMERCIALUNITS))
sales$TOTALUNITS <- as.numeric(as.character(sales$TOTALUNITS))
sales$GROSSSQUAREFEET <- as.numeric(as.character(sales$GROSSSQUAREFEET))
sales$LANDSQUAREFEET <- as.numeric(as.character(sales$LANDSQUAREFEET))
sales$SALEPRICE <- as.numeric(sales$SALEPRICE)

## transform dates, create SALEYEAR and SALEYRMO feature
sales$SALEDATE <- mdy(sales$SALEDATE)
sales$SALEYEAR <- as.Date(floor_date(sales$SALEDATE, "year"))
```

8.3.4 Recalculation Implementation

```
## recalcuate - real estate sales

## convert zero na values to zero in commercial unit count feature
sales$COMMERCIALUNITS[is.na(sales$COMMERCIALUNITS)] <- 0

## recalculate total unit feature
sales$TOTALUNITS <- (sales$RESIDENTIALUNITS + sales$COMMERCIALUNITS)

## convert zero values to na in bldg gross sqft feature
sales$GROSSSQUAREFEET[sales$GROSSSQUAREFEET == 0] <- NA
```

8.3.5 Subsetting Implementation

```
## subsetting - real estate sales

## remove observations categorically sales outside of scope
sales_subset <- subset(sales,
                       BUILDINGCLASSCATEGORY == "07 RENTALS - WALKUP APARTMENTS" |
                       BUILDINGCLASSCATEGORY == "08 RENTALS - ELEVATOR APARTMENTS" |
                       BUILDINGCLASSCATEGORY == "11A CONDO-RENTALS" |
                       BUILDINGCLASSCATEGORY == "14 RENTALS - 4-10 UNIT")

## remove observations numerically sales outside of scope
sales_subset <- subset(sales_subset,             # call data frame
                       LANDSQUAREFEET > 1 &      # properties with land rights
                       RESIDENTIALUNITS >= 4 &   # properties not multifamily
                       SALEPRICE >= 1000)        # sales threshold
```

8.3.6 Real Estate Sales Address Standarization Implementation

```
## address cleaning - real estate sales

## remove features for address format standard
sales_address <- sales_subset[c("ADDRESS",
                                "ZIPCODE")];
## rename headers
sales_address$street <- sales_address$ADDRESS
sales_address$city <- "New York"
sales_address$state <- "NY"
sales_address$zipcode <- sales_address$ZIPCODE

## remove original features
sales_address$ADDRESS <- NULL
sales_address$ZIPCODE <- NULL

## export .csv file
write.csv(sales_address,
          file <- "Data/other_data/sales_manhattan_subset.csv",
          row.names <- FALSE);

## remove original data frame
rm(sales_address)

## address clean with smartystreets
## reloading - post process

## reload data
sales_subset_add <- read.csv("Data/other_data/sales_manhattan_geocoded_subset.csv",
                             skip = 0,  # skip first n observations
                             header = TRUE,  # spec header names
                             check.names = FALSE,  # do not change header names
                             strip.white = TRUE,  # remove whitespace
                             stringsAsFactors = FALSE,  # spec string data type
                             na.strings = c("", "NA"));  # assign blank cells na

## rename column headers
sales_subset_add <- setnames(sales_subset_add,
                             old = c("street",
                                     "[delivery_line_1]",
                                     "[latitude]",
                                     "[longitude]"),
                             new = c("ADDRESS",
                                     "ADDRESS.USPS",
                                     "lat",
                                     "lon"));
## select features
sales_subset_add <- sales_subset_add[c("ADDRESS",
                                       "ADDRESS.USPS",
                                       "lat",
```

```
                                        "lon")];

## clean feature usps address - remove unit designators, #
sales_subset_add$ADDRESS.USPS <-
    gsub("#..*", "", sales_subset_add$ADDRESS.USPS)

## clean feature usps address - remove unit designators, apt
sales_subset_add$ADDRESS.USPS <-
    gsub("Apt..*", "", sales_subset_add$ADDRESS.USPS)

## clean feature usps address - remove leading & trailing whitespaces
sales_subset_add$ADDRESS.USPS <-
    str_trim(sales_subset_add$ADDRESS.USPS)

## clean feature usps address - remove within string whitespaces
sales_subset_add$ADDRESS.USPS <-
    str_squish(sales_subset_add$ADDRESS.USPS)

## merging - clean usps addresses

## pre-process for merging by renaming row index
row.names(sales_subset) <- 1:nrow(sales_subset)
row.names(sales_subset_add) <- 1:nrow(sales_subset_add)

## merge sales and sales clean by matching matching index
sales_subset_add <- merge(sales_subset,     # original data frame
                          sales_subset_add, # address clean data frame
                          by = 0,           # match by row index
                          all = TRUE);      # only include matching

## remove redundate and not applicable features
sales_subset_add <- sales_subset_add[c("ADDRESS.USPS",
                                       "ZIPCODE",
                                       "RESIDENTIALUNITS",
                                       "TOTALUNITS",
                                       "GROSSSQUAREFEET",
                                       "LANDSQUAREFEET",
                                       "SALEPRICE",
                                       "SALEDATE",
                                       "SALEYEAR",
                                       "lat",
                                       "lon")];
```

8.3.7 Land Lot Polygon Address Standarization Implementation

```
## clean address - land lots

## remove features for address clean format standard
pluto_add <- select(pluto_shape@data, "Address")
names(pluto_add) <- "street"

## rename headers
pluto_add$city <- "New York"
pluto_add$state <- "NY"

## include zipcode
pluto_add[4] <- select(pluto_shape@data, "ZipCode")
names(pluto_add)[4] <- "zipcode"

## export .csv file
write.csv(pluto_add,
          file <- "Data/other_data/pluto_address.csv",
          row.names <- FALSE)

## remove original data frame
rm(pluto_add)

## address clean with smartystreets
## reloading - post process

## reload data
pluto_add_clean <- read.csv("Data/other_data/pluto_address_clean.csv",
                        skip = 0,  # skip first n observations
                        header = TRUE,  # spec header names
                        check.names = FALSE,  # do not change header names
                        strip.white = TRUE,  # remove whitespace
                        stringsAsFactors = FALSE,  # spec string data type
                        na.strings = c("", "NA"));  # assign blank cells na

## rename column headers
pluto_add_clean <- setnames(pluto_add_clean,
                            old = c("street",
                                    "[delivery_line_1]"),
                            new = c("ADDRESS",
                                    "ADDRESS.USPS"));
## select features
pluto_add_clean <- pluto_add_clean[c("ADDRESS",
                                     "ADDRESS.USPS")];

## clean feature usps address - remove leading & trailing whitespaces
pluto_add_clean$ADDRESS.USPS <-
    str_trim(pluto_add_clean$ADDRESS.USPS)

## clean feature usps address - remove within string whitespaces
```

```
pluto_add_clean$ADDRESS.USPS <-
    str_squish(pluto_add_clean$ADDRESS.USPS)

## merging - clean usps addrreses

## pre-process for merging by renaming row index
row.names(pluto_shape@data) <- 1:nrow(pluto_shape@data)
row.names(pluto_add_clean) <- 1:nrow(pluto_add_clean)

## merge clean addresses to pluto shapefile by matching index
pluto_shape@data <- merge(pluto_shape@data, # original data frame
                          pluto_add_clean,  # address clean data frame
                          by = 0,           # match by row index
                          all = TRUE);      # only include matching

## remove features redundate and not applicable
pluto_shape@data <- pluto_shape@data["ADDRESS.USPS"]
```

List of Address Standardization Results

```
## preview clean usps addresses
head(pluto_shape@data$ADDRESS.USPS, 40)
```

8.3.8 Geocoding Implementation

```
## generate centroids - land lots

## calculate land lot centroids - wgs84 proj
pluto_shape_cent <- gCentroid(pluto_shape, byid = TRUE)
proj4string(pluto_shape_cent) <- proj4string(pluto_shape)
pluto_shape_cent <- as.data.frame(pluto_shape_cent@coords)
row.names(pluto_shape_cent) <- 1:nrow(pluto_shape_cent)
colnames(pluto_shape_cent) <- c("LON", "LAT")

## merge land lot centroids with clean usps addrreses
pluto_shape_cent_add <- merge(pluto_shape_cent,
                              pluto_add_clean,
                              by = 0,
                              all = FALSE)
## geocoordinates - merging

## pre-process for merging, remove duplicate land lot usps addreses
pluto_shape_cent_add <- pluto_shape_cent_add[!duplicated(
                        pluto_shape_cent_add$ADDRESS.USPS), ]

## ------------------------------------------------------ ##
## merge sales to land lot centroid coords by usps address
sales_subset_geo <- merge(sales_subset_add,
                          pluto_shape_cent_add,
                          by = "ADDRESS.USPS",
```

```
                          all.x = TRUE)  # keep all sales
## ------------------------------------------------------- ##

## remove redundate and not applicable features
sales_subset_geo <- sales_subset_geo[c("ADDRESS.USPS",
                                       "RESIDENTIALUNITS",
                                       "TOTALUNITS",
                                       "GROSSSQUAREFEET",
                                       "LANDSQUAREFEET",
                                       "SALEPRICE",
                                       "SALEYEAR",
                                       "lat", "lon",
                                       "LAT", "LON")];
## geocoordinates - substitution

## replace only if land lot centroid geo coords did not match sales address
sales_subset_geo$LON <- ifelse(is.na(sales_subset_geo$LON),
                               sales_subset_geo$lon, sales_subset_geo$LON);
sales_subset_geo$LAT <- ifelse(is.na(sales_subset_geo$LAT),
                               sales_subset_geo$lat, sales_subset_geo$LAT);

## replace clean address coords feature columns
sales_subset_geo$lon <- NULL
sales_subset_geo$lat <- NULL
```

8.3.9 Data Exploration Implementation

100 Most Transacted Properties Plot

```
## data exploration

## calculate number of sales per property
freq_table <- data.frame(table(sales_subset_geo$ADDRESS.USPS))

## truncate 100 most active properties by number of sales
active <- head(freq_table[order(freq_table$Freq,
                                decreasing = TRUE), ], 100);

## plot 100 most active properties by sales
plot(active$Freq,
     ylab = "Number of Sales",
     xlab = "Property Index",
     type = "h")
```

Average Number of Transactions Table

```
## data exploration

## view average number of sales per property
summary(freq_table$Freq)
```

8.4 Appendix D: Advanced Data Handling Procedures

8.4.1 Multiple Imputation by Chained Equations Implementaiton

```
## multiple imputation by chained equations

## conduct multiple imputation by chained equations (mice)
gsf_mice_model <- mice(sales_subset_geo,   # call data frame
                       method = "cart",    # class and regression tree method
                       m = 10,             # number of imputations
                       printFlag = FALSE); # hide status

## impute values to original data frame
sales_subset_mice <- mice::complete(gsf_mice_model)
```

Multiple Imputation by Chained Equations Density Plot

```
## plot imputation density distribution
densityplot(gsf_mice_model)
```

8.4.2 Feature Engineering Implementation

```
## feature engineering

## calculate price per residential to total unit ratio
sales_subset_mice$SALEPRICE.PER.RES.UNITS.RATIO <-
    round(sales_subset_mice$SALEPRICE *
        (sales_subset_mice$RESIDENTIALUNITS /
            sales_subset_mice$TOTALUNITS) /
                sales_subset_mice$TOTALUNITS, digits = 0);

## calculate price per unit total
sales_subset_mice$SALEPRICE.PER.TOTALUNITS <-
    round(sales_subset_mice$SALEPRICE /
        sales_subset_mice$TOTALUNITS, digits = 0);

## calculate price per gross sqft
sales_subset_mice$SALEPRICE.PER.GROSSSQUAREFEET <-
    round(sales_subset_mice$SALEPRICE /
        sales_subset_mice$GROSSSQUAREFEET, digits = 0);

## calculate price per land sqft
sales_subset_mice$SALEPRICE.PER.LANDSQUAREFEET <-
    round(sales_subset_mice$SALEPRICE /
            sales_subset_mice$LANDSQUAREFEET, digits = 0);
```

List of Feature Engineered Variables

```
# view feat names
names(sales_subset_mice)
```

8.4.3 Normalization Implementation

```
## normalization

## normalize - price
sales_subset_mice$NORM.SALEPRICE <-

        ## define vector
        sapply(X = sales_subset_mice["SALEPRICE"],

        ## create function
        FUN = function(x) {

        ## conduct min-max normalization
        (x - min(x)) / (max(x) - min(x)) * 100

    } # close function
) # close lapply

## normalize - price per residential to total unit ratio
sales_subset_mice$NORM.SALEPRICE.PER.RES.UNITS.RATIO <-

        ## define vector
        sapply(X = sales_subset_mice["SALEPRICE.PER.RES.UNITS.RATIO"],

        ## create function
        FUN = function(x) {

        ## conduct min-max normalization
        (x - min(x)) / (max(x) - min(x)) * 100

    } # close function
) # close lapply

## normalize - price per unit total
sales_subset_mice$NORM.SALEPRICE.PER.TOTALUNITS <-

        ## define vector
        sapply(X = sales_subset_mice["SALEPRICE.PER.TOTALUNITS"],

        ## create function
        FUN = function(x) {

        ## conduct min-max normalization
        (x - min(x)) / (max(x) - min(x)) * 100

    } # close function
) # close lapply

## normalize - price per gross sqft
sales_subset_mice$NORM.SALEPRICE.PER.GROSSSQUAREFEET <-
```

```
        ## define vector
        sapply(X = sales_subset_mice["SALEPRICE.PER.GROSSSQUAREFEET"],

        ## create function
        FUN = function(x) {

        ## conduct min-max normalization
        (x - min(x)) / (max(x) - min(x)) * 100

    } # close function
) # close lapply

## normalize - price per land sqft
sales_subset_mice$NORM.SALEPRICE.PER.LANDSQUAREFEET <-

        ## define vector
        sapply(X = sales_subset_mice["SALEPRICE.PER.LANDSQUAREFEET"],

        ## create function
        FUN = function(x) {

        ## conduct min-max normalization
        (x - min(x)) / (max(x) - min(x)) * 100

    } # close function
) # close lapply
```

8.4.4 Matrix Transformation Implementation

```
## data transformation

## price

# melt data into matrix from long to wide - price
melted_price <- dcast(sales_subset_mice,  # data frame
                ADDRESS.USPS + LAT + LON ~ SALEYEAR,  # rows ~ columns
                value.var = "NORM.SALEPRICE",  # measurement
                fun.aggregate = mean);  # aggregation procedure (annual)

## replace 'unknown' char string in the data with na
melted_price[melted_price == "NaN" ] = NA

## group by address in ADDRESS.USPS feat
melted_price <- melted_price %>% group_by(ADDRESS.USPS)

## price per residential unit ratio

## melt data into matrix from long to wide - price per residential unit count ratio
melted_price_res_un <- dcast(sales_subset_mice,  # data frame
```

```
                        ADDRESS.USPS + LAT + LON ~ SALEYEAR,  # rows ~ columns
                        value.var = "NORM.SALEPRICE.PER.RES.UNITS.RATIO", # measure
                        fun.aggregate = mean);  # aggregation procedure

## replace 'unknown' char string in the data with na
melted_price_res_un[melted_price_res_un == "NaN" ] = NA

## group by address in ADDRESS.USPS feat
melted_price_res_un <- melted_price_res_un %>% group_by(ADDRESS.USPS)

## price per total unit count

## melt data into matrix from long to wide - price per total unit count
melted_price_tot_un <- dcast(sales_subset_mice,  # data frame
                        ADDRESS.USPS + LAT + LON ~ SALEYEAR,  # rows ~ columns
                        value.var = "NORM.SALEPRICE.PER.TOTALUNITS", # measurement
                        fun.aggregate = mean);  # aggregation procedure

## replace 'unknown' char string in the data with na
melted_price_tot_un[melted_price_tot_un == "NaN" ] = NA

## group by address in ADDRESS.USPS feat
melted_price_tot_un <- melted_price_tot_un %>% group_by(ADDRESS.USPS)

## price per bldg gross sqft

## melt data into matrix from long to wide - price per bldg gross sqft
melted_price_gsf <- dcast(sales_subset_mice,  # data frame
                    ADDRESS.USPS + LAT + LON ~ SALEYEAR,  # rows ~ columns
                    value.var = "NORM.SALEPRICE.PER.GROSSSQUAREFEET", # measurement
                    fun.aggregate = mean);  # aggregation procedure

## replace 'unknown' char string in the data with na
melted_price_gsf[melted_price_gsf == "NaN" ] = NA

## group by address in ADDRESS.USPS feat
melted_price_gsf <- melted_price_gsf %>% group_by(ADDRESS.USPS)

## price per land sqft

## melt data into matrix from long to wide - price per land sqft
melted_price_lsf <- dcast(sales_subset_mice,  # data frame
                    ADDRESS.USPS + LAT + LON ~ SALEYEAR,  # rows ~ columns
                    value.var = "NORM.SALEPRICE.PER.LANDSQUAREFEET", # measurement
                    fun.aggregate = mean);  # aggregation procedure

## replace 'unknown' char string in the data with na
melted_price_lsf[melted_price_lsf == "NaN" ] = NA

## group by address in ADDRESS.USPS feat
melted_price_lsf <- melted_price_lsf %>% group_by(ADDRESS.USPS)
```

```r
## matrix transformation

## create matrix and lists - price
sales_matrix <- list()
sales_matrix[[1]] <- as.list(melted_price$ADDRESS.USPS)  # store address list
sales_matrix[[2]] <- as.list(melted_price$LON)  # store lon coord list
sales_matrix[[3]] <- as.list(melted_price$LAT)  # store lat coord list
sales_matrix[[4]] <- as.matrix(melted_price[4:18])  # store price matrix
sales_matrix[[5]] <- as.matrix(melted_price_res_un[4:18]) # store price per res un
sales_matrix[[6]] <- as.matrix(melted_price_tot_un[4:18])  # store price tot un
sales_matrix[[7]] <- as.matrix(melted_price_gsf[4:18])  # store price per bldg sqft
sales_matrix[[8]] <- as.matrix(melted_price_lsf[4:18])  # store price per land sqft

# rename matrix and lists to order
names(sales_matrix) <- c("address",
                         "lon",
                         "lat",
                         "price",
                         "price_res_un",
                         "price_tot_un",
                         "price_gsf",
                         "price_lsf")
```

Matrix Missingness Plot

```r
## randomly reorder for plot
random_index <- sample(nrow(sales_matrix$price))
matrix_plot <- sales_matrix$price[random_index, ]

## plot percentage missing values
matrixplot(matrix_plot,
           ylab = "Property Index",
           col = "firebrick",
           interactive = FALSE);
```

8.4.5 Descriptive Statistic Tables Implementation

Descriptive Statistics Table of Original Variables

```
## create descriptive stat subset summary
sales_subset_geo_plot <- sales_subset_geo[c("RESIDENTIALUNITS",
                                            "TOTALUNITS",
                                            "GROSSSQUAREFEET",
                                            "LANDSQUAREFEET",
                                            "SALEPRICE")]

## rename headers for table summary
sales_subset_geo_plot <- setnames(sales_subset_geo_plot,
                                  old = c("RESIDENTIALUNITS",
                                          "TOTALUNITS",
                                          "GROSSSQUAREFEET",
                                          "LANDSQUAREFEET",
                                          "SALEPRICE"),
                                  new = c("Residential Units",
                                          "Total Units",
                                          "GSF",
                                          "LSF",
                                          "Sale Price"))

## create descriptive stat table
kable(stat.desc(sales_subset_geo_plot),

digits = 2,  # truncate test mse decimal
caption = "Descriptive Statistics Summary of Original Variables"

)# close table creation
```

Descriptive Statistics Table of Feature Engineered Variables

```
## create descriptive stat subset summary
sales_subset_mice_plot <- sales_subset_mice[c("SALEPRICE.PER.RES.UNITS.RATIO",
                                              "SALEPRICE.PER.TOTALUNITS",
                                              "SALEPRICE.PER.GROSSSQUAREFEET",
                                              "SALEPRICE.PER.LANDSQUAREFEET")]

## rename headers for table summary
sales_subset_mice_plot <- setnames(sales_subset_mice_plot,
                                  old = c("SALEPRICE.PER.RES.UNITS.RATIO",
                                          "SALEPRICE.PER.TOTALUNITS",
                                          "SALEPRICE.PER.GROSSSQUAREFEET",
                                          "SALEPRICE.PER.LANDSQUAREFEET"),
                                  new = c("Sale / Residental Units",
                                          "Sale / Total Units",
                                          "Sale / GSF",
                                          "Sale / LSF"))

## create descriptive stat table
kable(stat.desc(sales_subset_mice_plot, basic = FALSE),
```

```
digits = 2,  # truncate test mse decimal
caption = "Descriptive Statistics Summary of Feature Engineered Variables"

)# close table creation
```

Descriptive Statistics Table of Normalized Feature Engineered Variables

```
## create descriptive stat subset summary
sales_subset_mice_plot_norm <-
    sales_subset_mice[c("NORM.SALEPRICE.PER.RES.UNITS.RATIO",
                        "NORM.SALEPRICE.PER.TOTALUNITS",
                        "NORM.SALEPRICE.PER.GROSSSQUAREFEET",
                        "NORM.SALEPRICE.PER.LANDSQUAREFEET")]

## rename headers for table summary
sales_subset_mice_plot_norm <- setnames(sales_subset_mice_plot_norm,
                                  old = c("NORM.SALEPRICE.PER.RES.UNITS.RATIO",
                                          "NORM.SALEPRICE.PER.TOTALUNITS",
                                          "NORM.SALEPRICE.PER.GROSSSQUAREFEET",
                                          "NORM.SALEPRICE.PER.LANDSQUAREFEET"),
                                  new = c("Sale / Residential Units",
                                          "Sale / Total Units",
                                          "Sale / GSF",
                                          "Sale / LSF"))

## create descriptive stat table
kable(stat.desc(sales_subset_mice_plot_norm, basic = FALSE),

digits = 2,  # truncate test mse decimal
caption = "Descriptive Statistics Summary of Normalized Feature Engineered Variables"

)# close table creation
```

8.5 Appendix E: Methodology

8.5.1 Conditional Random Sampling Implementation

```
## training and test set split by conditional random sampling

## introduce data into prediction model
M <- sales_matrix$price

## test set sampling threshold of observed values (not na)
## not percentage of total matrix
test_split <- 0.20

## fix random number for reproduction
set.seed(1234)

## create conditional repeat
repeat {

    ## duplicate original matrix (test set)
    A <- M

    ## conditional sampling for loop
    for (i in 1:nrow(A)) {  ## loop through rows not na
        if (length(which(!is.na(A[i, ]))) > 1) {  ## conditional statement

        ## conduct single sample removal for rows that meet conditional
        A[i, ][sample(which(!is.na(A[i, ])), size = 1)] <- NA

        } ## close matrix for loop conditional
    } ## close matrix for loop through rows

    ## stop repeat until test set sampling threshold is met
    if ( (mean(is.na(A)) - mean(is.na(M))) /
          mean(!is.na(M)) > test_split ) break

} ## close conditional repeat
```

8.5.2 Sequential Coordinante Wise Descent - 50 Iteration Maximum

```
## non-negative matrix factorization
k_rank <- 2 ## set k rank

## sequential coordinante wise descent - 50 iteration maximum
system.time(  # calculate cost

    ## conduct non-negative matrix factorization
    nnmf_scd_50 <- nnmf(A,                  # call matrix
                        k = k_rank,         # call k rank
                        method = "scd",     # call algorithm
                        max.iter = 50,      # max num of iterations epoch
                        alpha = 0.0004,     # alpha-beta param
                        beta = 0.02,        # alpha-beta param
                        loss = "mse",       # calc test mse loss fun type
                        trace = 1,          # calc test mse every iteration
                        check.k = FALSE,    # do not check k rank
                        verbose = FALSE,    # do not preview training status
                        n.threads = 0);     # utilize all cores in parallel

) # close cost calculation

## view result
nnmf_scd_50
```

8.5.3 Sequential Coordinante Wise Descent - 500 Iteration Maximum

```
## sequential coordinante wise descent - 500 iteration maximum
system.time(  # calculate cost

    ## conduct non-negative matrix factorization
    nnmf_scd_500 <- nnmf(A,                  # call matrix
                         k = k_rank,         # call k rank
                         method = "scd",     # call algorithm
                         alpha = 0.0004,     # alpha-beta param
                         beta = 0.02,        # alpha-beta param
                         max.iter = 500,     # max num of iterations epoch
                         loss = "mse",       # calc test mse loss fun type
                         trace = 1,          # calc test mse every iteration
                         check.k = FALSE,    # do not check k rank
                         verbose = FALSE,    # do not preview training status
                         n.threads = 0);     # utilize all cores in parallel

) # close cost calculation

## view result
nnmf_scd_500
```

8.5.4 Sequential Coordinante Wise Descent - 5000 Iteration Maximum

```
## sequential coordinante wise descent - 5000 iteration maximum
system.time(  # calculate cost

    ## conduct non-negative matrix factorization
    nnmf_scd_5000 <- nnmf(A,                  # call matrix
                          k = k_rank,         # call k rank
                          method = "scd",     # call algorithm
                          alpha = 0.0004,     # alpha-beta param
                          beta = 0.02,        # alpha-beta param
                          max.iter = 5000,    # max num of iterations epoch
                          loss = "mse",       # calc test mse loss fun type
                          trace = 1,          # calc test mse every iteration
                          check.k = FALSE,    # do not check k rank
                          verbose = FALSE,    # do not preview training status
                          n.threads = 0);     # utilize all cores in parallel

) # close cost calculation

## view result
nnmf_scd_5000
```

8.5.5 Lee & Seung's Multiplicative - 50 Iteration Maximum

```
## lee & seung's multiplicative - 50 iteration maximum
system.time(  # calculate cost

    ## conduct non-negative matrix factorization
    nnmf_lee_50 <- nnmf(A,                  # call matrix
                        k = k_rank,         # call k rank
                        method = "lee",     # call algorithm
                        alpha = 0.0004,     # alpha-beta param
                        beta = 0.02,        # alpha-beta param
                        max.iter = 50,      # max num of iterations epoch
                        loss = "mse",       # calc test mse loss fun type
                        trace = 1,          # calc test mse every iteration
                        check.k = FALSE,    # do not check k rank
                        verbose = FALSE,    # do not preview training status
                        n.threads = 0);     # utilize all cores in parallel

) # close cost calculation

## view result
nnmf_lee_50
```

8.5.6 Lee & Seung's Multiplicative - 500 Iteration Maximum

```
## lee & seung's multiplicative - 500 iteration maximum
system.time(  # calculate cost

    ## conduct non-negative matrix factorization
    nnmf_lee_500 <- nnmf(A,                  # call matrix
                         k = k_rank,         # call k rank
                         method = "lee",     # call algorithm
                         alpha = 0.0004,     # alpha-beta param
                         beta = 0.02,        # alpha-beta param
                         max.iter = 500,     # max num of iterations epoch
                         loss = "mse",       # calc test mse loss fun type
                         trace = 1,          # calc test mse every iteration
                         check.k = FALSE,    # do not check k rank
                         verbose = FALSE,    # do not preview training status
                         n.threads = 0);     # utilize all cores in parallel

) # close cost calculation

## view result
nnmf_lee_500
```

8.5.7 Lee & Seung's Multiplicative - 5000 Iteration Maximum

```
## lee & seung's multiplicative - 5000 iteration maximum
system.time(  # calculate cost

    ## conduct non-negative matrix factorization
    nnmf_lee_5000 <- nnmf(A,                  # call matrix
                          k = k_rank,         # call k rank
                          method = "lee",     # call algorithm
                          alpha = 0.0004,     # alpha-beta param
                          beta = 0.02,        # alpha-beta param
                          max.iter = 5000,    # max num of iterations epoch
                          loss = "mse",       # calc test mse loss fun type
                          trace = 1,          # calc test mse every iteration
                          check.k = FALSE,    # do not check k rank
                          verbose = FALSE,    # do not preview training status
                          n.threads = 0);     # utilize all cores in parallel

) # close cost calculation

## view result
nnmf_lee_5000
```

Iteration Maximum & Convergence Speed Plot

```
## plot convergence speed

## create plot
plot(x = NULL,                    # create empty plot
     xlim = c(1, 250),            # x axis range - num iter max
     ylim = c(0.00, 0.5),         # y axis range - test mse
     xlab = "Epoch",              # x label - epoch
     ylab = "Training MSE"); # title

        ## draw line - sequential coordinante wise descent - 50 iteration maximum
        lines(cumsum(nnmf_scd_50$average.epochs),
              nnmf_scd_50$mse,       # call test mse from nnmf object
              lwd = 1.25,            # line weight
              col = "goldenrod1");   # line color

        ## draw line - sequential coordinante wise descent - 500 iteration maximum
        lines(cumsum(nnmf_scd_500$average.epochs),
              nnmf_scd_500$mse,      # call test mse from nnmf object
              lwd = 1.25,            # line weight
              col = "darkorange");   # line color

        ## draw line - sequential coordinante wise descent - 5000 iteration maximum
        lines(cumsum(nnmf_scd_5000$average.epochs),
              nnmf_scd_5000$mse,     # call test mse from nnmf object
              lwd = 1.25,            # line weight
              col = "firebrick");    # line color

        ## draw line - lee & seung's multiplicative - 50 iteration maximum
        lines(cumsum(nnmf_lee_50$average.epochs),
              nnmf_lee_50$mse,      # call test mse from nnmf object
              lwd = 1.25,           # line weight
              col = "green4");      # line color

        ## draw line - ee & seung's multiplicative - 500 iteration maximum
        lines(cumsum(nnmf_lee_500$average.epochs),
              nnmf_lee_500$mse,     # call test mse from nnmf object
              lwd = 1.25,           # line weight
              col = "blue");        # line color

        ## draw line - lee & seung's multiplicative - 5000 iteration maximum
        lines(cumsum(nnmf_lee_5000$average.epochs),
              nnmf_lee_5000$mse,    # call test mse from nnmf object
              lwd = 1.25,           # line weight
              col = "darkmagenta"); # line color

## create legend
legend('topright', # legend location
       bty = "n",  # legend seperation type
       lwd = 2,    # legend line weight
```

```
       ## legend labels as ordered
       legend = c('SCD-50',
                  'SCD-500',
                  'SCD-5000',
                  'LEE-50',
                  'LEE-500',
                  'LEE-5000'),

       ## legend colors as ordered
       col = c("goldenrod1",
               "darkorange",
               "firebrick",
               "green4",
               "blue",
               "darkmagenta");

) # close legend
```

Iteration Maximum & Convergence Speed Table

```
## create table summary function
summary <- function(x) {

    if (x$n.iteration < 2) {
        rel.tol <- NA_real_;
    } else {
        err <- tail(x$target.loss, 2);
        rel.tol <- diff(err) / mean(err);
    }

    return(c(
        'MSE' = tail(x$mse, 1),
        'MKL' = tail(x$mkl, 1),
        'Target' = tail(x$target.loss, 1),
        'Rel. Tol.' = abs(rel.tol),
        'Total Epochs' = sum(x$average.epochs),
        'Interation Max' = x$n.iteration, x$run.time[1:3]));
}

## create table
kable(sapply(X = list(

            ## results as table contents
            "SCD-50" = nnmf_scd_50,
            "SCD-500" = nnmf_scd_500,
            "SCD-5000" = nnmf_scd_5000,
            "LEE-50" = nnmf_lee_50,
            "LEE-500" = nnmf_lee_500,
            "LEE-5000" = nnmf_lee_5000

            ), ## close table contents

        ##  call function
        FUN = function(x) {

            z <- summary(x)
            sapply(z, sprintf, fmt = '%.4g')

        } ## close function
    ), ## close sapply

    ## table label and values
    align = rep('r', 5),
    caption = "Iteration Maximum & Convergence Speed Comparison"

) ## close table creation
```

8.5.8 Leave One Out Cross Validation Implementation

```
## loocv on matrix for estimating optimal k rank

## calculate cost
system.time(

## conditional for loop through matrix
for (i in 1:nrow(M)  ) {   ## loop through rows
    for (j in 1:ncol(M) ) {   ## loop through columns
        if (!is.na(M[i, j]) ) {   ## loop through values (not na)

        ## leave one out resampling
        M2 <- M   # duplicate original matrix (test set)
        M2[i, j] <- NA  # remove single sample (training set)

        ## prediction for single sample across k rank range
        test_mse <- sapply(X = 1:8,  # set k rank limit for cross validation

           ## create k rank function
           FUN = function(k) {

               ## non-negative matrix factorization
               z <- nnmf(M2,                # call matrix
                         k = k,             # call k rank
                         method = "scd",    # call algorithm
                         alpha = 0.0004,    # alpha-beta param
                         beta = 0.02,       # alpha-beta param
                         max.iter = 50,     # max num of iterations epoch
                         loss = "mse",      # calc test mse loss fun type
                         trace = 1,         # calc test mse every iteration
                         check.k = FALSE,   # do not check k rank
                         verbose = FALSE,   # do not preview training status
                         n.threads = 0);    # utilize all cores in parallel

               ## calculate test mse
               mean((with(z, W %*% H)[i, j] - M[i, j]) ^ 2)

           } # close k rank function
        ) # close prediction for single sample test mse

        ## preview test mse results
        print(test_mse)

        ## return optimal k rank
        optimal_k_rank <- which.min(test_mse);

        } # close matrix for loop conditional
    } # close matrix nested for loop through rows
} # close matrix nested for loop through columns
) # close cost calculation
```

Google Cloud Engine

```
## google compute engine user setup
project_id <- "xxxxx"
compute_zone <- "xxxxx"
api_key <- "xxxxx.json"

## system environment settings
Sys.setenv(GCE_DEFAULT_PROJECT_ID=project_id,
           GCE_DEFAULT_ZONE=compute_zone,
           GCE_AUTH_FILE=api_key)

## google compute engine authentication
googleAuthR::gar_auth_service(api_key)

## create rstudio virtual machine on google compute engine
vm <- gce_vm(template = "rstudio",
             name = "ml-rstudio",
             username = "xxxxx", password = "xxxxx",
             predefined_type = "xxxx-8")

## confirm virtual machine status
vm
```

8.5.9 Prediction Performance Comparison Implementation

Non-Negative Matrix Factorization Impementation

```
## non-negative matrix factorization
nnmf_time <- proc.time()  # calculate cost - start time

## conduct non-negative matrix factorization
nnmf_model <- nnmf(A,                   # call matrix
                   k = 1,               # call k rank
                   method = "scd",      # call algorithm
                   alpha = 0.0004,      # alpha-beta param
                   beta = 0.02,         # alpha-beta param
                   max.iter = 50,       # max num of iterations epoch
                   loss = "mse",        # calc test mse loss fun type
                   trace = 1,           # calc test mse every iteration
                   check.k = FALSE,     # do not check k rank
                   verbose = FALSE,     # do not preview training status
                   n.threads = 0);      # utilize all cores in parallel

## non-negative matrix factorization prediction
nnmf <- with(nnmf_model, W %*% H)

## duplicate original matrix for imputation
nnmf_predict <- A

## non-negative matrix factorization imputation
nnmf_predict[is.na(nnmf_predict)] <- nnmf[is.na(nnmf_predict)]

## close cost calculation - end time
proc.time() - nnmf_time
```

Simon Funk's Singular Value Decomposition Implementation

```
## simon funk's singular value decomposition with stochastic gradient descent
svd_time <- proc.time()  # calculate cost - start time

## conduct simon funk's singular value decomposition
svd_model <- funkSVD(A, k = 1)

## simon funk's singular value decomposition prediction
svd_funk <- tcrossprod(svd_model$U, svd_model$V)

## duplicate original matrix for imputation
svd_predict <- A

## imputate simon funk's singular value decomposition prediction
svd_predict[is.na(svd_predict)] <- svd_funk[is.na(svd_predict)]

## close cost calculation - end time
proc.time() - svd_time
```

Random Forest Implementation

```
## random forest
rf_time <- proc.time() # calculate cost - start time

## conduct random forest regression
rf_model <- missForest(A, verbose = FALSE)

## random forest prediction
rf_predict <- rf_model$ximp

## close cost calculation - end time
proc.time() - rf_time
```

Row Means Implementation

```
## means
mean_time <- proc.time()  ## calculate cost

## preserve original values
data_obs <- which(is.na(A), arr.ind = TRUE)

## duplicate original matrix for imputation
mean_predict <- A

## mean imputation by row means
mean_predict[data_obs] <- rowMeans(mean_predict,
                                   na.rm = TRUE)[data_obs[ ,1]]
## clear na's from row means
mean_predict[mean_predict == "NaN"] <- NA

## preserve original values and row means
data_obs <- which(is.na(mean_predict), arr.ind = TRUE)

## remove components
rm(data_obs)

## close cost calculation
proc.time() - mean_time
```

Random Imputation Implementation

```
## random
random_time <- proc.time()  ## calculate cost

## conduct random imputation with existing values
random_predict <- na.random(A)

## close cost calculation
proc.time() - random_time
```

Prediction Comparison Table Implemenation

```
## comparison table

## create table
kable(sapply(X = list(

            # test mse results as table contents
            "Random" = random_predict,
            "Means" = mean_predict,
            "Forest" = rf_predict,
            "Funk SVD" = svd_predict,
            "NNMF" = nnmf_predict

            ), # close table contents

    # call mse function
    FUN = mse.mkl,

    # test set
    obs = M

), # close test mse

## table label and values
digits = 2,  # truncate test mse decimal
caption = "Prediction Performance Comparison"

) # close table creation
```

8.5.10 Non-Negative Matrix Factorization Prediction Implementation

```
## repeat non-negative matrix factorization - list of matrices

## calculate cost - start time
nnmf_list_time <- proc.time()

## conduct non-negative matrix factorization - list of matrices
sales_matrix_model <- lapply(X = sales_matrix[4:8],

         ## create function
         FUN = function(x) {

         ## conduct non-negative matrix factorization
         nnmf(x,              # call matrix
         k = 1,               # call k rank
         method = "scd",      # call algorithm
         alpha = 0.0004,      # alpha-beta param
         beta = 0.02,         # alpha-beta param
         max.iter = 50,       # max num of iterations epoch
         loss = "mse",        # calc test mse loss fun type
         trace = 1,           # calc test mse every iteration
         check.k = FALSE,     # do not check k rank
         verbose = FALSE,     # do not preview training status
         n.threads = 0)       # utilize all cores in parallel

    } # close function
) # close lapply on matrices

## non-negative matrix factorization prediction - list of matrices
nnmf <- lapply(X = sales_matrix_model,

        ## create function
        FUN = function(x) {

        ## non-negative factorization
        with(x, W %*% H)

    } # close function
) # close lapply on matrices

## duplicate original list of matrices
sales_matrix_predict <- sales_matrix

## non-negative matrix factorization imputation - list of matrices

## price
sales_matrix_predict$price[is.na(sales_matrix_predict$price)] <-
    nnmf$price[is.na(sales_matrix_predict$price)]

## price per res unit ratio
```

```
sales_matrix_predict$price_res_un[is.na(sales_matrix_predict$price_res_un)] <-
    nnmf$price_res_un[is.na(sales_matrix_predict$price_res_un)]

## price per total unit
sales_matrix_predict$price_tot_un[is.na(sales_matrix_predict$price_tot_un)] <-
    nnmf$price_tot_un[is.na(sales_matrix_predict$price_tot_un)]

## price per bldg gross sqft
sales_matrix_predict$price_gsf[is.na(sales_matrix_predict$price_gsf)] <-
    nnmf$price_gsf[is.na(sales_matrix_predict$price_gsf)]

## price per land sqft
sales_matrix_predict$price_lsf[is.na(sales_matrix_predict$price_lsf)] <-
    nnmf$price_lsf[is.na(sales_matrix_predict$price_lsf)]

## close cost calculation - end time
proc.time() - nnmf_list_time
```

8.5.11 Spatial Contiguity, Multivariate Weight, Cluster Number Implementation

```
## spatial contiguity constraint
spatial_const <- 12345

## multivariate weight specification - integerally rank ordered
multi_weight_spec <- list(1,  # price
                          3,  # price per res unit ratio
                          4,  # price per total unit (includes commercialf)
                          5,  # per per bldg gross sqft
                          2); # price per land sqft
## k number of clusters
k_clusters <- 100
```

HiClimR Schematic

```
## plot
knitr::include_graphics("Figures/HiClimR.png")
```

8.5.12 SCWMHC Average Linkage Implementation

```
## spatially constrained weighted multivariate hierarchical clustering
clust_mean <- HiClimR(

    ## input matrix actuals and imputations
    x = sales_matrix_predict[4:8],

    ## do not conduct coarsening spatial resolution
    lon = sales_matrix_predict$lon, lonStep = 1,
    lat = sales_matrix_predict$lat, latStep = 1,

    ## spatial contiguity constraint
    contigConst = spatial_const,

    ## pre-processing
    standardize = list(TRUE, TRUE, TRUE, TRUE, TRUE),
    weightMVC = multi_weight_spec,

    ## clustering specs
    method = "average",

    ## cluster validation
    validClimR = TRUE,
    rawStats = TRUE,
    minSize = 1,
    k = k_clusters,

    ## create plot
    plot = FALSE,
    dendrogram = TRUE,
    colPalette = NULL,
    labels = FALSE,
    hang = 0,
    pch = 15,
    cex = 1,

    ## preview status
    verbose = FALSE
)
```

8.5.13 SCWMHC Median Linkage Implementation

```
## spatially constrained weighted multivariate hierarchical clustering
clust_median <- HiClimR(

    ## input matrix actuals and imputations
    x = sales_matrix_predict[4:8],

    ## do not conduct coarsening spatial resolution
    lon = sales_matrix_predict$lon, lonStep = 1,
    lat = sales_matrix_predict$lat, latStep = 1,

    ## spatial contiguity constraint
    contigConst = spatial_const,

    ## pre-processing
    standardize = list(TRUE, TRUE, TRUE, TRUE, TRUE),
    weightMVC = multi_weight_spec,

    ## clustering specs
    method = "median",

    ## cluster validation
    validClimR = TRUE,
    rawStats = TRUE,
    minSize = 1,
    k = k_clusters,

    ## create plot
    plot = FALSE,
    dendrogram = TRUE,
    colPalette = NULL,
    labels = FALSE,
    hang = 0,
    pch = 15,
    cex = 1,

    ## preview status
    verbose = FALSE
)
```

8.5.14 SCWMHC Complete Linkage Implementation

```
## spatially constrained weighted multivariate hierarchical clustering
clust_complete <- HiClimR(

    ## input matrix actuals and imputations
    x = sales_matrix_predict[4:8],

    ## do not conduct coarsening spatial resolution
    lon = sales_matrix_predict$lon, lonStep = 1,
    lat = sales_matrix_predict$lat, latStep = 1,

    ## spatial contiguity constraint
    contigConst = spatial_const,

    ## pre-processing
    standardize = list(TRUE, TRUE, TRUE, TRUE, TRUE),
    weightMVC = multi_weight_spec,

    ## clustering specs
    method = "complete",

    ## cluster validation
    validClimR = TRUE,
    rawStats = TRUE,
    minSize = 1,
    k = k_clusters,

    ## create plot
    plot = FALSE,
    dendrogram = TRUE,
    colPalette = NULL,
    labels = FALSE,
    hang = 0,
    pch = 15,
    cex = 1,

    ## preview status
    verbose = FALSE
)
```

8.5.15 SCWMHC Centroid Linkage Implementation

```
## spatially constrained weighted multivariate hierarchical clustering
clust_centroid <- HiClimR(

    ## input matrix actuals and imputations
    x = sales_matrix_predict[4:8],

    ## do not conduct coarsening spatial resolution
    lon = sales_matrix_predict$lon, lonStep = 1,
    lat = sales_matrix_predict$lat, latStep = 1,

    ## spatial contiguity constraint
    contigConst = spatial_const,

    ## pre-processing
    standardize = list(TRUE, TRUE, TRUE, TRUE, TRUE),
    weightMVC = multi_weight_spec,

    ## clustering specs
    method = "centroid",

    ## cluster validation
    validClimR = TRUE,
    rawStats = TRUE,
    minSize = 1,
    k = k_clusters,

    ## create plot
    plot = FALSE,
    dendrogram = TRUE,
    colPalette = NULL,
    labels = FALSE,
    hang = 0,
    pch = 15,
    cex = 1,

    ## preview status
    verbose = FALSE
)
```

8.5.16 SCWMHC Ward's Method Implementation

```
## spatially constrained weighted multivariate hierarchical clustering
clust_ward <- HiClimR(

    ## input matrix actuals and imputations
    x = sales_matrix_predict[4:8],

    ## do not conduct coarsening spatial resolution
    lon = sales_matrix_predict$lon, lonStep = 1,
    lat = sales_matrix_predict$lat, latStep = 1,

    ## spatial contiguity constraint
    contigConst = spatial_const,

    ## pre-processing
    standardize = list(TRUE, TRUE, TRUE, TRUE, TRUE),
    weightMVC = multi_weight_spec,

    ## clustering specs
    method = "ward",

    ## cluster validation
    validClimR = TRUE,
    rawStats = TRUE,
    minSize = 1,
    k = k_clusters,

    ## create plot
    plot = FALSE,
    dendrogram = TRUE,
    colPalette = NULL,
    labels = FALSE,
    hang = 0,
    pch = 15,
    cex = 1,

    ## preview status
    verbose = FALSE
)
```

SCWMHC Ward's Method Implementation - Optimal Number of Clusters

```
## spatially constrained weighted multivariate hierarchical clustering
clust_ward_43 <- HiClimR(

    ## input matrix actuals and imputations
    x = sales_matrix_predict[4:8],

    ## do not conduct coarsening spatial resolution
    lon = sales_matrix_predict$lon, lonStep = 1,
    lat = sales_matrix_predict$lat, latStep = 1,

    ## spatial contiguity constraint
    ## apply heavy geo weight on the multivariate dissimilarity index
    contigConst = spatial_const,

    ## pre-processing
    standardize = list(TRUE, TRUE, TRUE, TRUE, TRUE),
    weightMVC = multi_weight_spec,

    ## clustering specs
    method = "ward",

    ## cluster validation
    validClimR = TRUE,
    rawStats = TRUE,
    minSize = 1,
    k = 43,

    ## create plot
    plot = FALSE,
    dendrogram = TRUE,
    colPalette = NULL,
    labels = FALSE,
    hang = 0,
    pch = 15,
    cex = 1,

    ## preview status
    verbose = FALSE
)
```

8.5.17 Point Data Implementation

```
## store coordinate range
clust_coords <- as.data.frame(clust_ward_43$coords)
clust_coords$Lon <- as.numeric(as.character(clust_coords$Lon))
clust_coords$Lat <- as.numeric(as.character(clust_coords$Lat))

## store submarket id and address of property geocoordinate range
submrkt <- as.data.frame(clust_ward_43$region) # create data frame
names(submrkt) <- "SubmarketID"        # rename column headers
submrkt$Address <- sales_matrix_predict$address

## create spatial points data frame for spatial join with polygons
submrkt_pts <- SpatialPointsDataFrame(coords = clust_coords,  # wgs84 proj
               data = submrkt, proj4string = CRS(proj4string(pluto_shape)));
```

Point Data Implementation - Optimal Number of Clusters

```
## store coordinate range
clust_ward_coords_43 <- as.data.frame(clust_ward_43$coords)
clust_ward_coords_43$Lon <- as.numeric(as.character(clust_ward_coords_43$Lon))
clust_ward_coords_43$Lat <- as.numeric(as.character(clust_ward_coords_43$Lat))

## store submarket id and address of property coordinate range
submrkt_ward_43 <- as.data.frame(clust_ward_43$region) # create data frame
names(submrkt_ward_43) <- "SubmarketID" # rename column headers
submrkt_ward_43$Address <- sales_matrix_predict$address

## create spatial points data frame for spatial join with polygons
submrkt_ward_pts_43 <- SpatialPointsDataFrame(coords = clust_ward_coords_43,
                            data = submrkt_ward_43, # wgs84 proj
                            proj4string = CRS(proj4string(pluto_shape)))
## check on work
submrkt_ward_pts_check_43 <- cbind(clust_ward_coords_43, submrkt_ward_43)
head(submrkt_ward_pts_check_43, 10)
```

8.5.18 Spatial Join Implementation

```
## spatial join and subset land lot polygons to intersecting points
## keep only land lots that spatially intersect with points
submrkt_poly <- pluto_shape[!is.na(sp::over(pluto_shape,
                           sp::geometry(submrkt_pts))), ]

## restore submarket id info by spatial joining point info to polygons
submrkt_poly@data <- cbind(submrkt_poly@data,
                                over(submrkt_poly, submrkt_pts))
```

Spatial Join Implementation - Optimal Number of Clusters

```
## spatial join and subset land lot polygons to intersecting points
## keep only land lots that spatially intersect with sales geocode
submrkt_ward_poly_43 <- pluto_shape[!is.na(sp::over(pluto_shape,
                                      sp::geometry(submrkt_ward_pts_43))), ]

## restore submarket id info by spatial joining point info to polygons
submrkt_ward_poly_43@data <- cbind(submrkt_ward_poly_43@data,
                          over(submrkt_ward_poly_43, submrkt_ward_pts_43))
```

8.6 Appendix F: Results & Discussion

8.6.1 Average Linkage Table

```
## preview correlation results
kable(clust_mean$statSum,
      caption = "Average Linkage Correlation Summary");
```

Average Linkage Cluster Correlation Differences Histogram

```
## plot number of clusters
plot(abs(clust_mean$diffCor),
     ylab = "Absolute Value of Correlation Differences",
     xlab = "Number of Clusters",
     type = "h",
     ylim = c(0, 0.12))
```

Average Linkage Dendrogram

```
## create dendrogram
plot(clust_mean,
     hang = -1,
     label = FALSE,
     xlab = "",
     sub = "",
     main = "")

## viz cut line
abline(h = 1005,
       lty = "dotted")

## create cluster bounds
rect.hclust(clust_mean,
            k = 22)
```

8.6.2 Median Linkage Table

```
## preview correlation results
kable(clust_median$statSum,
      caption = "Median Linkage Correlation Summary");
```

Median Linkage Cluster Correlation Differences Histogram

```
## plot number of clusters
plot(abs(clust_median$diffCor),
     ylab = "Absolute Value of Correlation Differences",
     xlab = "Number of Clusters",
     type = "h",
     ylim = c(0, 0.12))
```

Median Linkage Dendrogram

```
## create dendrogram
plot(clust_median,
     hang = -1,
     label = FALSE,
     xlab = "",
     sub = "",
     main = "")

## viz cut line
abline(h = 445,
       lty = "dotted")

## create cluster bounds
rect.hclust(clust_median,
            k = 39)
```

8.6.3 Complete Linkage Table

```
## preview correlation results
kable(clust_complete$statSum,
      caption = "Complete Linkage Correlation Summary");
```

Complete Linkage Cluster Correlation Differences Histogram

```
## plot number of clusters
plot(abs(clust_complete$diffCor),
     ylab = "Absolute Value of Correlation Differences",
     xlab = "Number of Clusters",
     type = "h",
     ylim = c(0, 0.12))
```

Complete Linkage Dendrogram

```
## create dendrogram
plot(clust_complete,
     hang = -1,
     label = FALSE,
     xlab = "",
     sub = "",
     main = "")

## viz cut line
abline(h = 2870,
       lty = "dotted")

## create cluster bounds
rect.hclust(clust_complete,
            k = 14)
```

8.6.4 Centroid Linkage Table

```
## preview correlation results
kable(clust_centroid$statSum,
      caption = "Centroid Linkage Correlation Summary");
```

Centroid Linkage Cluster Correlation Differences Histogram

```
## plot number of clusters
plot(abs(clust_centroid$diffCor),
     ylab = "Absolute Value Difference Between Correlations",
     xlab = "Number of Clusters",
     type = "h",
     ylim = c(0, 0.12))
```

Centroid Linkage Dendrogram

```
## create dendrogram
plot(clust_centroid,
     hang = -1,
     label = FALSE,
     xlab = "",
     sub = "",
     main = "")

## viz cut line
abline(h = 575,
       lty = "dotted")

## create cluster bounds
rect.hclust(clust_centroid,
            k = 20)
```

8.6.5 Ward's Method Table

```
## preview correlation results
kable(clust_ward$statSum,
      caption = "Ward's Linkage Correlation Summary");
```

Ward's Method Cluster Correlation Differences Histogram

```
## plot number of clusters
plot(abs(clust_ward$diffCor),
     ylab = "Absolute Value of Correlation Differences",
     xlab = "Number of Clusters",
     type = "h",
     ylim = c(0, 0.12))
```

Ward's Method Dendrogram

```
## create dendrogram
plot(clust_ward,
     hang = -1,
     label = FALSE,
     xlab = "",
     sub = "",
     main = "")

## viz cut line
abline(h = 375,
       lty = "dotted")

## create cluster bounds
rect.hclust(clust_ward,
            k = 43)
```

8.6.6 Comparing the Highest Performing Results Table Implementation

```
## create data frame
clust_num <- data.frame("Average", "Median", "Complete", "Centroid",
                        "Ward's");

colnames(clust_num) <- c("Average", "Median", "Complete", "Centroid",
                         "Ward's");

## store number of clusts
clust_num$Average <- which.max(abs(clust_mean$diffCor))
clust_num$Median <- which.max(abs(clust_median$diffCor))
clust_num$Complete <- which.max(abs(clust_complete$diffCor))
clust_num$Centroid <- which.max(abs(clust_centroid$diffCor))
clust_num$`Ward's` <- which.max(abs(clust_ward$diffCor))

## create data frame
clust_results <- data.frame("Average", "Median", "Complete", "Centroid",
                            "Ward's");

colnames(clust_results) <- c("Average", "Median", "Complete", "Centroid",
                             "Ward's");

## store clusting results
clust_results$Average <- clust_mean[["diffCor"]][[which.max(abs(
    clust_mean$diffCor))]]
clust_results$Median <- clust_median[["diffCor"]][[which.max(abs(
    clust_median$diffCor))]]
clust_results$Complete <- clust_complete[["diffCor"]][[which.max(abs(
    clust_complete$diffCor))]]
clust_results$Centroid <- clust_centroid[["diffCor"]][[which.max(abs(
    clust_centroid$diffCor))]]
clust_results$`Ward's` <- clust_ward[["diffCor"]][[which.max(abs(
    clust_ward$diffCor))]]

## combine data frames
clust_table <- rbind(clust_num, clust_results)
row.names(clust_table) <- c("Number of Clusters", "Correlation Difference")

## create results table
knitr::kable(clust_table,
             row.names = TRUE,
             caption = "Linkage Method Comparison of High Performing Results");
```

8.6.7 Comparing the Lowest Performing Results Table Implemenation

```
## create data frame
clust_num_min <- data.frame("Average", "Median", "Complete", "Centroid",
                            "Ward's");

colnames(clust_num_min) <- c("Average", "Median", "Complete", "Centroid",
                             "Ward's");

## store number of clusts
clust_num_min$Average <- which.min(abs(clust_mean$diffCor))
clust_num_min$Median <- which.min(abs(clust_median$diffCor))
clust_num_min$Complete <- which.min(abs(clust_complete$diffCor))
clust_num_min$Centroid <- which.min(abs(clust_centroid$diffCor))
clust_num_min$`Ward's` <- which.min(abs(clust_ward$diffCor))

## create data frame
clust_results_min <- data.frame("Average", "Median", "Complete", "Centroid",
                                "Ward's");

colnames(clust_results_min) <- c("Average", "Median", "Complete", "Centroid",
                                 "Ward's");

## store clusting results
clust_results_min$Average <- clust_mean[["diffCor"]][[which.min(abs(
    clust_mean$diffCor))]]
clust_results_min$Median <- clust_median[["diffCor"]][[which.min(abs(
    clust_median$diffCor))]]
clust_results_min$Complete <- clust_complete[["diffCor"]][[which.min(abs(
    clust_complete$diffCor))]]
clust_results_min$Centroid <- clust_centroid[["diffCor"]][[which.min(abs(
    clust_centroid$diffCor))]]
clust_results_min$`Ward's` <- clust_ward[["diffCor"]][[which.min(abs(
    clust_ward$diffCor))]]

## combine data frames
clust_table_min <- rbind(clust_num_min, clust_results_min)
row.names(clust_table_min) <- c("Number of Clusters", "Correlation Difference")

## create results table
knitr::kable(clust_table_min,
             row.names = TRUE,
             caption = "Linkage Method Comparison of Low Performing Results");
```

8.6.8 Submarket Maps of Manhattan Implementation

```
## submarket mapping

## map view centroid search
map_centroid_search <- "EMPIRE STATE BUILDING"

## geocode map view centroid
map_view <- geocode_OSM(map_centroid_search,
                        projection = "WGS84",
                        return.first.only = TRUE,
                        as.data.frame = TRUE,
                        server = "http://nominatim.openstreetmap.org")

## create color palette by submarket cluster id
submarket_colors <- colorNumeric(palette = "Set1",
                                 domain = submrkt_ward_poly_43@data$SubmarketID)
## initialize map
submarket_map = leaflet() %>%  # run leaflet

    # basemap appearance
    addProviderTiles(providers$CartoDB.Positron) %>%

    # default location
    setView(lng = map_view$x,     # longitude coord
            lat = map_view$y,     # latitude coord
            zoom = 11.5) %>%      # geo scale

    # display submarket polygons
    addPolygons(data = submrkt_ward_poly_43, # call ward optimal num of clusters
            fill = TRUE,          # polygon fill, option
            fillColor = ~submarket_colors(submrkt_ward_poly_43@data$SubmarketID),
            fillOpacity = 0.75,   # polygon fill, graphic transparency
            weight = 1.00,        # polygon stroke, outline weight
            color = ~submarket_colors(submrkt_ward_poly_43@data$SubmarketID),
            opacity = 1.00) %>%   # polygon stroke, graphic transparency

    # display nta polygons
    addPolygons(data = nta_shape, # call nta shapefile
            fill = FALSE,         # polygon fill, option
            fillColor = NULL,     # polygon fill, color
            fillOpacity = 0.00,   # polygon fill, graphic transparency
            weight = 0.90,        # polygon stroke, outline weight
            color = "black",      # polygon stroke, color
            opacity = 0.50) %>%   # polygon stroke, graphic transparency

    # display scale bar
    addScaleBar(position = "bottomleft")  # include scale bar

## display map
submarket_map
```

8.6.9 Small Area Fair Market Rent Maps of Manhattan Implementation

```
## hud small area fair market rent mapping

## convert zcta id data type to numeric
zcta_shape@data$zcta <- as.numeric(zcta_shape@data$zcta)

## create color palette by zcta
zcta_colors <- colorNumeric(palette = "Set3",
                            domain = zcta_shape@data$zcta)

## initialize map
safmr_map = leaflet() %>%  # run leaflet

    # basemap appearance
    addProviderTiles(providers$CartoDB.Positron) %>%

    # display location
    setView(lng = map_view$x,          # longitude coord
            lat = map_view$y,          # latitude coord
            zoom = 11.5) %>%           # geo scale

    # display zcta polygons
    addPolygons(data = zcta_shape,     # call zcta shapefile
                fill = TRUE,           # polygon fill, option
                fillColor = ~zcta_colors(zcta_shape@data$zcta),
                fillOpacity = 0.50,    # polygon fill, graphic transparency
                weight = 1.25,         # polygon stroke, outline weight
                color = ~zcta_colors(zcta_shape@data$zcta),
                opacity = 1.00) %>%    # polygon stroke, graphic transparency

    # display nta polygons
    addPolygons(data = nta_shape,      # call nta shapefile
                fill = FALSE,          # polygon fill, option
                fillColor = NULL,      # polygon fill, color
                fillOpacity = 0.00,    # polygon fill, graphic transparency
                weight = 1.00,         # polygon stroke, outline weight
                color = "black",       # polygon stroke, color
                opacity = 0.50) %>%    # polygon stroke, graphic transparency

    # display scale bar
    addScaleBar(position = "bottomleft")  # include scale bar

## display map
safmr_map
```

8.6.10 Complete Interactive Web Map Implementation

```
## initialize map
web_map = leaflet() %>%  # run leaflet

    # basemap appearance
    addProviderTiles(providers$CartoDB.Positron) %>%

    # default location
    setView(lng = map_view$x,      # longitude coord
            lat = map_view$y,      # latitude coord
            zoom = 11.5) %>%       # geo scale

    # display real estate sales locations
    addCircleMarkers(data = sales_subset_geo,   # call pluto
                     ~LON, ~LAT,
                     fill = TRUE,           # marker fill, option
                     radius = 0.75,         # marker fill, size
                     fillColor = "black",   # marker fill, color
                     fillOpacity = 0.50,    # marker fill, graphic transparency
                     weight = 0.00,         # marker stroke, outline weight
                     opacity = 0.00,        # marker stroke, graphic transparency
                     color = "black",       # marker stroke, color
                     popup = ~as.character(pluto_shape_cent_add$ADDRESS.USPS),
                     group = "Real Estate Sales") %>% # toggle

    # display pluto polygons
    addPolygons(data = pluto_shape, # call nta shapefile
                fill = FALSE,          # polygon fill, option
                fillColor = NULL,      # polygon fill, color
                fillOpacity = 0.00,    # polygon fill, graphic transparency
                weight = 0.50,         # polygon stroke, outline weight
                color = "black",       # polygon stroke, color
                opacity = 0.50,        # polygon stroke, graphic transparency
                group = "Land Lot Polygons") %>% # toggle

    # display zcta polygons
    addPolygons(data = zcta_shape, # call nta shapefile
                fill = FALSE,          # polygon fill, option
                fillColor = NULL,      # polygon fill, color
                fillOpacity = 0.00,    # polygon fill, graphic transparency
                weight = 0.75,         # polygon stroke, outline weight
                color = "black",       # polygon stroke, color
                opacity = 0.50,        # polygon stroke, graphic transparency
                group = "Zip Code Tabulation Areas") %>% # toggle

    # display safmr polygons
    addPolygons(data = zcta_shape,     # call zcta shapefile
                fill = TRUE,           # polygon fill, option
                fillColor = ~zcta_colors(zcta_shape@data$zcta), # polygon fill, color
                fillOpacity = 0.50,    # polygon fill, graphic transparency
```

```
                weight = 1.25,        # polygon stroke, outline weight
                color = ~zcta_colors(zcta_shape@data$zcta), # polygon stroke, color
                opacity = 1.00,       # polygon stroke, graphic transparency
                group = "Small Area Fair Market Rent Areas") %>% # toggle

    # display nta polygons
    addPolygons(data = nta_shape, # call nta shapefile
                fill = FALSE,         # polygon fill, option
                fillColor = NULL,     # polygon fill, color
                fillOpacity = 0.00,   # polygon fill, graphic transparency
                weight = 1.00,        # polygon stroke, outline weight
                color = "black",      # polygon stroke, color
                opacity = 0.50,       # polygon stroke, graphic transparency
                group = "Neighborhood Tabulation Areas") %>% # toggle

    # display submarket polygons
    addPolygons(data = submrkt_ward_poly_43, # call ward optimal num of clusters
                fill = TRUE,          # polygon fill, option
                fillColor = ~submarket_colors(submrkt_ward_poly_43@data$SubmarketID),
                fillOpacity = 0.75,   # polygon fill, graphic transparency
                weight = 1.00,        # polygon stroke, outline weight
                color = ~submarket_colors(submrkt_ward_poly_43@data$SubmarketID),
                opacity = 1.00,       # polygon stroke, graphic transparency
                group = "Submarkets of Manhattan") %>% # toggle

    # layer control toggle
    addLayersControl(options = layersControlOptions(collapsed = FALSE),
                     overlayGroups = c("Real Estate Sales",
                                       "Land Lot Polygons",
                                       "Zip Code Tabulation Areas",
                                       "Small Area Fair Market Rent Areas",
                                       "Neighborhood Tabulation Areas",
                                       "Submarkets of Manhattan")) %>%

    # layer control toggle unselected by default
    hideGroup(c("Real Estate Sales",
                "Land Lot Polygons",
                "Zip Code Tabulation Areas",
                "Small Area Fair Market Rent Areas",
                "Neighborhood Tabulation Areas",
                "Submarkets of Manhattan")) %>%

    # display scale bar
    addScaleBar(position = "bottomleft")  # include scale bar

## display map
web_map
```