

BETTER ACCESS TO PARKS TO IMPROVE POPULATION HEALTH IN
MECKLENBURG COUNTY, NC

by

Coline Christiane Dony

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Geography and Urban Regional Analysis

Charlotte

2016

Approved by:

Dr. Eric M. Delmelle

Dr. Elizabeth C. Delmelle

Dr. Janni Sorensen

Dr. A. Suzanne Boyd

Dr. Ramesh ("Rudy") Shankar

ABSTRACT

COLINE CHRISTIANE DONY. Better access to parks to improve population health in Mecklenburg County, NC. (Under the direction of Dr. ERIC M. DELMELLE)

Better access to parks can increase our level of physical activity. Increasingly aware of this association and its impact on community health; urban leaders started incorporating access to parks as an indicator of quality of life. However, two studies that evaluate geographic access to parks in Mecklenburg County (NC), each identify different areas of low access. This demonstrates the challenge to quantify the notion of “access”.

In this study, geographic disparities in park access are evaluated for Mecklenburg County, which encompasses the City of Charlotte. The location of parks managed by the county and those managed by other entities such as homeowner associations (HOA) were collected. Collecting HOA park locations is an important contribution because it represents about 60% of all park locations in the county and that data is left out in all previous studies on this topic. Another contribution is the comparison of access between four modes of transport rather than focusing on access by car. Then, results are shown from surveys that were carried out to understand perceptions on park access from visitors and understand their use of parks for physical exercise. Lastly, online park reviews are presented to show real-time monitoring capabilities of park satisfaction.

Results show that neighborhoods with a high percentage of blacks had a lower number of parks per square mile; indicating a form of environmental injustice. Surveys indicate that most visitors came to a park to engage in physical exercise. Finally, I dispute previous studies for their weak evaluation of park access and their use of incomplete data.

ACKNOWLEDGMENTS

Several people contributed to this dissertation. First, I must thank my advisor, Dr. Eric Delmelle (UNCC, Department of Geography & Earth Sciences), for his positive presence and academic support which had great influence on me and allowed me to complete this thesis based on my own interests and in line with my beliefs.

Particularly, I want to thank the members of my committee, who have agreed to devote their time and expertise to examine and evaluate my final work as a doctoral student. First, Dr. Elizabeth Delmelle (UNCC, Department of Geography & Earth Sciences) whose research collaboration gave a boost to my critical thinking. Second, Dr. Janni Sorenson (UNCC, Department of Geography & Earth Sciences) who sharpened my knowledge about existing social injustices in Mecklenburg County. Third, I would like to thank Dr. A. Suzanne Boyd (UNCC, Department of Social Work) for her continued positive support and for improving my understanding of social determinants of health. Finally, Dr. Rudy Shakhar (UNCC, Director of EPIC Power Modernization Cluster) who has shown me possible applications of my research outside of geography and health.

I owe thanks to Dr. Wenwu Tang (UNCC, Department of Geography & Earth Sciences) for advice and for providing me with an office space and access to computational resources that were necessary for my research.

I want to acknowledge the Graduate School at The University of North Carolina at Charlotte for supporting my studies financially through the Graduate Assistance Support Plan. Special thanks to the Joanna R. Baker Memorial Graduate Fellowship, who provided additional research funds which helped carry out surveys at public parks and to KIND® who provided 800 free snack bars to give to survey respondents, which likely

boosted our response rate.

I especially want to thank the volunteers that offered their time to collect surveys at parks with me; Mike Desjardins, William Sun, Jonathan Cagle, Collin Wood, Kelly O'Connor, Kalee Wilson, Manal Mahmoud, Kaitlin Colandrea, Danny Yonto and Joshua Leslie.

I also would like to thank my study companions who have helped me improve as a doctoral student and whom I sincerely respect, namely Whalen Dillon, Dr. Monica Dorning, Danny Yonto, Tonya Farrow-Chestnut, Wenpeng Feng, Dr. Amos (Zhaoya) Gong, Stephan Hoche, Alex Hohl, Kelly Brawn, Jing Deng, Meijuan Jia, Mike Desjardins and Adam Griffith.

Lastly, I would also like to express my gratitude to my family. Thank you to Anaïs, Julia, Noëlie and Berenger, as well as my mother for the support they provided me from across the Atlantic Ocean. Last but not least, I want to thank my husband, Keith Waters, for the countless support he gave me during the pursuit of my doctoral degree.

Charlotte,

August 2016

INTRODUCTION

Today, fewer jobs require physical labor and our spare time is spent on more sedentary activities such as watching television (Hill, Wyatt, Reed, & Peters, 2003). Additionally, placing elevators, escalators and automatic sliding doors or planning with a car-centric vision have additionally cut down how much physical exercise we are faced with on a daily basis. These changes make our lives easier and more accessible, but also require us to supplement our day with artificial physical exercise to maintain a healthy physiology. In this dissertation, I am particularly interested to find ways to get people to engage in regular physical exercise and to understand the influence of access to public places on people's engagement in physical activity within their communities.

Bedimo-Rung, Mowen and Cohen (2005) argue that living near public open spaces contributes to higher levels of physical activity and to lower levels of stress resulting in fewer mental health problems. Given both physical and mental health benefits for residents living nearby public open spaces, improving access to public open spaces can become a prevention strategy to reduce heart disease. Since 2012, the Trust for Public Land evaluates the availability of public parks in cities of the United States (U.S.) and updates each city's ParkScore® annually on their interactive webpage. The score is calculated based on median park size, percent parkland within city limits, spending per resident on park and recreation, median availability of amenities, and percent of the population living within a ten-minute walk of a public park. Compared to other cities, Charlotte (North Carolina), ranked at the very bottom for five consecutive years. In my dissertation I take a closer look at access to parks and recreation in Mecklenburg County, North Carolina, which encompasses the City of Charlotte.

First, my dissertation identifies areas that have limited or no access to parks applying a novel variation on the Floating Catchment Area (FCA) method, which builds upon earlier methods used in access studies. Another contribution I make is to compare access to parks between four modes of transportation (driving, transit, walking, and bicycling) rather than focusing on access by car. I then compare how much these areas coincide with those identified (1) by the Trust for the Public Land's *ParkScore*® study and (2) by the Charlotte-Mecklenburg's Quality of Life Study. This comparison shows significant result differences between both studies and show additional differences with results of my analysis. The lack of robustness between study results demonstrates one of the main weaknesses of spatial access studies, which I discuss.

Second, I present results from surveys administered to visitors at 18 public parks within Mecklenburg County. The survey instrument was designed to better understand travel behavior to public parks and utilization of public parks for physical exercise. This part of my dissertation contributes to the empirical understanding of parks as a place for physical activity. During the survey collection process, one additional weakness of previous studies on park access in Mecklenburg County was identified. One significant data source seems to be left out in both studies, namely the location of non-public parks. For example, many parks and recreational spaces are managed by homeowner associations (HOAs) in Mecklenburg County. Previous studies have limited their data to the location of public parks managed by the county to evaluate access. For my dissertation, I use tax parcel data in combination with OpenStreetMap data – a freely available source of volunteered geographic information, to create an inventory of non-public parks. Non-public parks count about twice as much locations compared to public

parks, which constitutes a significant data source to evaluate park access. Therefore, this data collection is an important contribution to the literature and disputes the completeness of the data used in previous studies.

Lastly, I present the sentiment (either negative or positive) of reviews left about parks in Mecklenburg County by users on Google and on Foursquare¹ using sentiment analysis. Data from these online commenting platforms have the benefit to be readily accessible and user's activity is continuous, which could help public park officials monitor parks where improvements are necessary. Since collecting information on visitor satisfaction through public surveys is time consuming, social media platforms could be a useful complementary tool to monitor satisfaction. Charlotte is an urban area that is rapidly growing in population and that needs more strategic decision-making tools that can keep up with the pace at which their neighborhoods are developing.

My contribution to the field of Geography is twofold. First, I make a contribution to spatial modeling by modifying the state-of-the-art model for spatial accessibility. My critical analysis of this modeling approach highlights many of its remaining weaknesses, such as inconsistent definitions of “access” and the importance of complete inventories of service locations to evaluate accessibility appropriately. I argue that geographic modeling approaches have an incredible potential in the identification of environmental injustices, but have not been widely used so far because of these limitations. Second, I contribute to our understanding of the notion of “access”, which is a key concept in many branches of the field of Geography, especially that of Health Geography. I call attention to the

¹ Foursquare is a mobile application that keeps track of user's visits at shops, restaurants, parks and other places. The application asks users to leave ratings and comments about the places they visited. There is a social aspect about this application, which lets users search for nearby places their friends have been visiting.

possibility that the notion of “access” will be interpreted differently depending on the service (e.g. parks vs. grocery stores) and depending on the local context (e.g. urban vs. suburban context). These differences should influence the way we model and evaluate spatial accessibility, so that findings are appropriate for the local context at hand. Taking into account the local context is key to make geographic modeling a useful and effective tool for decision making.

TABLE OF CONTENTS

TABLE OF CONTENTS	x
LIST OF TABLES	xiv
LIST OF FIGURES	xv
LIST OF ABBREVIATIONS	xvii
LITERATURE REVIEW	1
1. The Rise of Physical Inactivity	2
2. Disease Prevention and Urban Planning	4
3. Spatial Justice of Park Access	8
4. Concept of Access	11
5. Monitoring Approaches for Rapidly Growing Urban Areas	18
STUDY AREA: MECKLENBURG COUNTY	21
RESEARCH OBJECTIVES AND CONTRIBUTIONS	27
1. Research Objective 1	27
2. Research Objective 2	28
3. Research Objective 3	29
4. Assumptions	30
DATA	32
1. Data on Parks and Recreation from the Department of Park and Recreation	32
2. Data on Parks and Recreation from Mecklenburg County	35
3. Data on Parks and Recreation from OpenStreetMap	40
4. Public Park Inventory	41
5. Non-Public Park Inventory	41
6. Park Reviews	42

	xi
7. Survey of Public Park Visitors	42
8. Neighborhood and Block Group Level Characteristics	47
9. Criminal Activity Nearby Parks	48
10. Data Limitations	48
METHODS	50
1. Modeling Spatial Access (Potential): the Floating Catchment Area (FCA) Method	50
2. Analyzing Access (Revealed) to Parks	58
3. Calculating Park Availability within the Local Context	60
4. Estimating Park Satisfaction of Online Reviews	60
5. Methodological Limitations	61
RESULTS	63
1. Spatial Access (Potential) to Public Parks in Mecklenburg County	63
2. Access (Revealed) to Parks in Mecklenburg County	68
3. Park Availability in Mecklenburg County	80
4. What is the Sentiment of Online Comments at Parks in Mecklenburg County?	85
DISCUSSIONS	90
1. Modeling Spatial Accessibility to Parks Using the VFCA Method	90
2. Accessibility to Parks in Mecklenburg County: Potential vs. Revealed	91
3. Willingness to Travel to Parks	94
4. The Use of Parks as a Place for Physical Activity	95
5. Access to Parks and Environmental Justice	96
6. Real Time Monitoring of Park Satisfaction	97
RECOMMENDATION	98
CONCLUSIONS	100
FUTURE RESEARCH	102

	xii
REFERENCES	103
APPENDIX A: IRB APPROVALS AND AMENDMENTS	110
1. Initial Approval	110
2. IRB Amendment Approval (September 2015)	111
3. IRB Amendment Approval (October 2015)	112
4. IRB Renewal Approval	113
APPENDIX B: SURVEY INSTRUMENT	114
1. Questionnaire Page 1	114
2. Questionnaire Page 2	115
APPENDIX C: INVITATION TO FILL-OUT WEB-BASED SURVEY	116
APPENDIX D: INVITATION TO VOLUNTEER	117
APPENDIX E: CHARACTERISTICS OF PARKS INCLUDED IN SURVEY	118
APPENDIX F: SURVEY DATA – CLEANING PROCESS (STATA)	119
APPENDIX G: NORMALITY AND SKEWNESS TEST ON VISITOR’S ACCESS SCORE AND IMPORTANCE OF PARK CHARACTERISTICS	131
APPENDIX H: SURVEY DATA – DESCRIPTIVE STATISTICS AND STATISTICAL ANALYSIS (STATA)	136
APPENDIX I: PARKS AND RECREATIONAL FACILITIES DATA – CLEANING PROCESS (PYTHON)	139
1. Cleaning Framework	139
2. Clean Greenways	143
3. Extract Greenway Entrances	146
4. Clean Park Location Points	148
5. Clean Park Properties	151
6. Extract Additional Parks and Recreational Facilities	163
7. Link Data Together	172

	xiii
8. Clean Attribute Information	178
9. Edit Attribute Information	183
10. Add Park Weblinks	188
11. Extract Clean Datasets	193
12. Generate Crime Location Points	197
13. Scrape Crime Data in Mecklenburg County	199
14. Scrape Geocodes of Crime Locations in Mecklenburg County	200
15. Clean Quality of Life	201
16. Extend Survey Data with Location Information	202
17. Calculate Park Densities and Convert to Contours	204
APPENDIX J: COLLECT REVIEWS	206
1. Collect Google Reviews	206
2. Collect Foursquare Reviews	209
APPENDIX K: MEASURE SPATIAL ACCESSIBILITY (VFCA)	213

LIST OF TABLES

TABLE 1: Number of surveys collected per park type	43
TABLE 2: Questions from the survey that will be used to respond to research questions 3 through 5	59
TABLE 3: Descriptive statistics on the walk time to the closest park	71
TABLE 4: Summary of reported travel times	72
TABLE 5: Summeray of reported travel time per park type	73
TABLE 6: Summary of reported travel time per park type after types were aggregated	75
TABLE 7: Summary of responses about engagement in physical activity	77
TABLE 8: Summary of responses about frequency of engagement in physical activity at parks	78
TABLE 9: Summary of responses about engagement in physical activity by park type	78
TABLE 10: Summary of responses about frequency of engagement in physical activity by park type	79

LIST OF FIGURES

FIGURE 1: Population reporting no engagement in physical activity	3
FIGURE 2: The health impact pyramid (Frieden, 2010)	5
FIGURE 3: Determinants of health and associated intervention and prevention strategies	6
FIGURE 4: Forms of healthcare strategies to alleviate heart disease	7
FIGURE 5: Parks and recreational facilities in Mecklenburg County	21
FIGURE 6: Level of need for parks in Charlotte (TPL, 2016)	23
FIGURE 7: Proximity to public parks and recreational facilities in Mecklenburg County (Quality of Life Study, 2015)	25
FIGURE 8: Location of parks and recreational facilities managed by organizations other than the Department of Park and Recreation	26
FIGURE 9: Causal assumption made to support studies that model and identify areas with low access to parks	30
FIGURE 10: Neighborhood and community parks managed by or in collaboration with the Department of Parks and Recreation of Mecklenburg County	33
FIGURE 11: Regional parks and mature preserves managed by or in collaboration with the Department of Parks and Recreation of Mecklenburg County	34
FIGURE 12: Content of the “commercial impervious surfaces” dataset made available by Mecklenburg County’s Open mapping portal	39
FIGURE 13: Example of reviews left on Google about Freedom Park in Charlotte	42
FIGURE 14: Population pyramid of surveys versus Mecklenburg County	43
FIGURE 15: Park selection in Mecklenburg County, where surveys would be conducted in the month of September	44
FIGURE 16: Residence of surveyed visitors in and around Mecklenburg County	47
FIGURE 17: Illustration of the first step in the Two-step Floating Catchment Area (2SFCA) method, which calculates each park-to-population ratio	51
FIGURE 18: Illustration of the second step in the Two-step Floating Catchment Area (2SFCA) method, which calculates access to parks for each geographic unit	52
FIGURE 19: Illustration of the concept of the variable-width floating catchment area	54

(VFCA) method

FIGURE 20: Spatial accessibility for scenarios I and II, for four different modes of transportation	64
FIGURE 21: Distribution of spatial accessibility scores for scenario I for four different modes of transportation	65
FIGURE 22: Spatial accessibility for scenario III, for four different modes of transportation	66
FIGURE 23: Distribution of spatial accessibility scores for scenario III for four different modes of transportation	67
FIGURE 24: Score reported by park visitors on their overall parks score and the importance of four park characteristics.	68
FIGURE 25: Results of the ordinal logistic regression for research question 3	69
FIGURE 26: Median and 95 th percentile travel times	72
FIGURE 27: Visitor's reported travel time per park type	74
FIGURE 28: Results of the ordinal logistic regression for research question 4	76
FIGURE 29: Results of the logistic regression for research question 5	79
FIGURE 30: Number of public parks and recreational facilities and their accessibility	81
FIGURE 31: Number of private parks and recreational facilities against racial diversity	82
FIGURE 32: Number of parks and recreational facilities against population density	83
FIGURE 33: Number of parks and recreational facilities (offset by attraction) and health insurance	84
FIGURE 34: Online sentiment about parks on Google and arts and cultural participation	86
FIGURE 35: Online sentiment about parks on Foursquare and resident adopted streams	87
FIGURE 36: Positive reviews left about parks on Google	88
FIGURE 37: Negative reviews left about parks on Google	88

LIST OF ABBREVIATIONS

2SFCA – Two-step Floating Catchment Area

CDC – Centers for Disease Control and Prevention

DHHS – United States Department of Health and Human Services

FCA – Floating Catchment Area

GIS – Geographic Information Systems

HOA – Homeowner Association

IAPD – Illinois Association of Park Districts

MCHD – Mecklenburg County Department of Health

NC – North Carolina

TPL –Trust for Public Land

U.S. – United States of America

VFCA – Variable Floating Catchment Area

LITERATURE REVIEW

Living near public open spaces contributes to higher levels of physical activity and to lower levels of stress and fewer mental health problems (Bedimo-Rung, Mowen, & Cohen, 2005), which has been confirmed by a number of studies (e.g., Lopez & Hynes, 2006). In my dissertation, I evaluate disparities regarding access to parks and recreational facilities in Mecklenburg County (NC), which encompasses the City of Charlotte. This year – in 2016, Charlotte ranks 95 out of 98 U.S. cities based on a park score measured by the Trust for Public Land. With such a low ranking (total score of 30 out of 100; TPL, 2016), there seems to be opportunity for the City of Charlotte to improve its public park system, which in turn could translate into a greater participation in physical activity from its residents.

In this literature review I address five major topics. In the first section, I summarize the rise of physical inactivity in Mecklenburg County, the U.S. and worldwide. Then, in the second section, I review a number of planning or community-based solutions, such as planning parks and recreational facilities to influence more people to engage in regular physical exercise. In the third section, I address social justice issues of access to parks and recreation from a social determinants of health perspective. In my dissertation, I evaluate access to parks and recreation using spatial accessibility methods used in the field of geography. Therefore, in the fourth section, I provide a review of these methods and their limitations. Finally, in the last section of this literature review, I summarize recent studies that explore the use of social media to monitor and improve urban public places.

1. The Rise of Physical Inactivity

There has been a worldwide decrease in physical activity which has been associated with a global increase in noncommunicable disease, which includes heart disease and chronic diseases (Bauman & Craig, 2005). This decrease in physical activity has been reported since our shift to a more sedentary lifestyle. Fewer jobs require physical labor and our spare time is spent on more sedentary activities such as watching television (Hill, Wyatt, Reed, & Peters, 2003). Placing elevators, escalators and automatic sliding doors and car-centric planning have further cut down how much physical exercise we are faced with on a daily basis. These changes make our lives easier and more accessible, but require us to supplement our day with artificial physical exercise to maintain a healthy and functional heart.

Yusuf et al. (2004) found that physical inactivity was responsible for 12.2% of the global rate of heart attacks, after adjusting for risk factors associated with cardiovascular diseases (as cited in Mozaffarian et al., 2016b). Today, less than half the population of U.S. adults meets the national guidelines for physical activity (Haskell et al., 2007). The most recent findings from the Physical Activity Council indicate that 27.7% of the U.S. population is inactive (Physical Activity Council, 2016). Residents of Mecklenburg County who report that they do not engage in physical activity has fluctuated between 17% and 22% since 2005 (MCHD, 2015). These figures are consistently lower compared to the North Carolina and nation-wide averages (see Figure 1).

In a prospective cohort study by Matthews et al. (2014) on all-cause mortality in Southern U.S. states (including North Carolina), it was found that individuals who engaged in more physical activity had a lower risk for cardiovascular disease. Among

whites, the most active individuals had a 31% lower risk of cardiovascular disease compared to those that were least active. Among blacks, the risk for cardiovascular disease was reduced by 19% for the most active individuals compared to the least active.

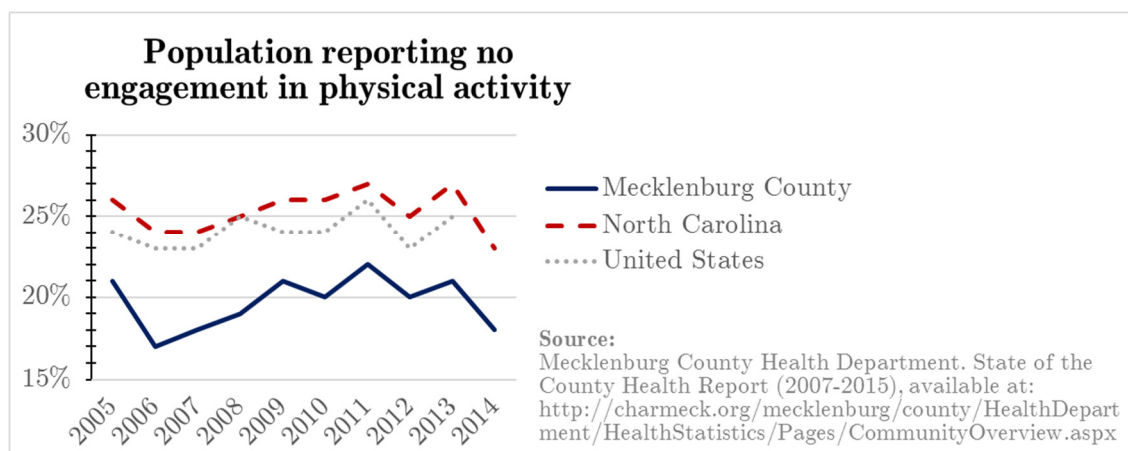


Figure 1: Population reporting no engagement in physical activity.

In a recent study, Mueller et al. (2016) show that 20% of preventable, natural, all-cause deaths in Barcelona (Spain) are attributed to a combination of (1) physical inactivity among residents, (2) their exposure to higher than recommended levels of air pollution, noise and heat and (3) their access to greenspace. Their study finds that physical inactivity contributes to the biggest share (7.9%) of preventable deaths at the census tract level. Also, the authors find that poor access to parks – which is defined as a lack of greenspace of 0.5-hectare or larger within 300 meters, contributes to less than one percent (0.8%) of preventable deaths. Although I speculate that there are confounding effects greenspaces may have on levels of physical activity, air pollution and heat, it is worth mentioning their finding within the scope of this dissertation.

Measuring the direct impact of better access to parks on community health is

challenging. A number of studies however, have presented findings that show evidence to believe there the impact of park access on hospital expenditures is significant.

Rosenberger et al. (2005) used spatial regression (spatial lag models) to understand the link between hospital expenditures, physical inactivity and recreation availability in West Virginia, controlling for differences in health care availability and socioeconomic status between its counties. Their study finds that counties with more active residents were associated with higher availability for recreation and with lower hospital expenditures. Also, the authors find that more recreation opportunities were associated with less health expenditures per county, which they use as a supporting argument to convince decision makers to invest in the supply of recreational opportunities.

In my dissertation, I do not test the impact of parks and recreation on community health. Instead, I use research findings such as those presented by Rosenberger et al. (2005), to justify the importance of my research. Thus, the positive impact of parks and recreation on community health is one of the assumption I make in my dissertation.

2. Disease Prevention and Urban Planning

The socio-ecological model is a popular framework in public health and prevention. This framework is based the biopsychosocial model (Engel, 1977) of George L. Engel's, an American psychiatrist who proposed to take a more holistic approach to prevent and cure diseases. Diseases are caused by a range of factors that play at different levels, from molecules, to the individual, and all the way to their communities, cultures and societies. The prevention of noncommunicable diseases will require action at each of these levels.

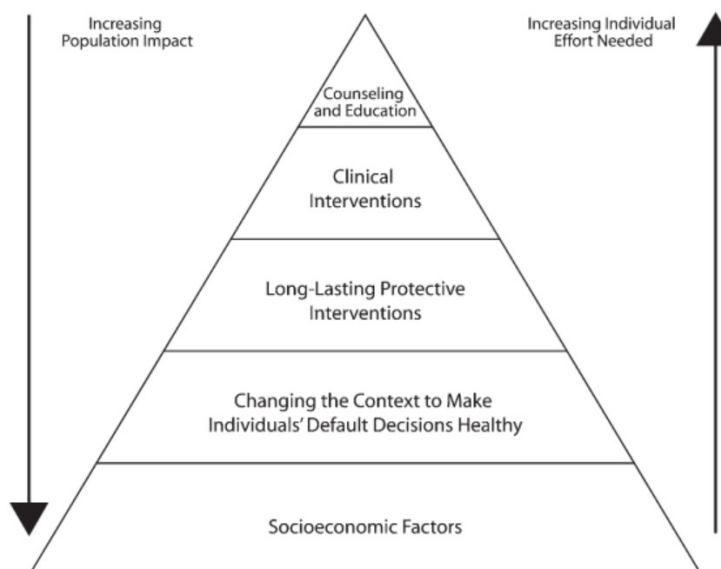


Figure 2: The health impact pyramid (Frieden, 2010)

Almost 25 years after Engel's suggestion, McGinnis, Williams-Russo and Knickman (2002) estimate that 95% of the national healthcare budget is still invested in medical treatments. McGinnis and Foege (1993) estimated that only 10-15% of mortality in the U.S. can be prevented by improving healthcare availability and treatments, while 40% could be prevented by behavioral changes.

To emphasize the impact of healthcare prevention at different levels of the socioecological model, CDC director Thomas Frieden (2010) developed the health impact pyramid (see Figure 2 on page 5). Efforts to address socioeconomic determinants of health requires the least effort at the individual level while impacting the population the most, making them the base of the pyramid. Raising the minimum wage is one example of a policy that would fall at the base of the health impact pyramid. The second layer of the pyramid, represent efforts to change the context to make individual's default decisions healthier. In my dissertation, I discuss efforts to plan parks and recreation to encourage more physical exercise within communities and thus it fits at this stage of the

pyramid.

In their article, Koohsari, Badland and Giles-Corti (2013) recommend re-introducing public health as a priority for urban planners. They argue that planners are well positioned to study impacts of the built environment on physical activity at different scales and for different populations and thus could develop various strategies targeting different levels of government and acting upon different factors influencing physical activity. The “public health planner” would find most cost-effective program(s) at each level of the socio-economic model that can generate the highest improvement in health outcomes (see example in Figure 3).

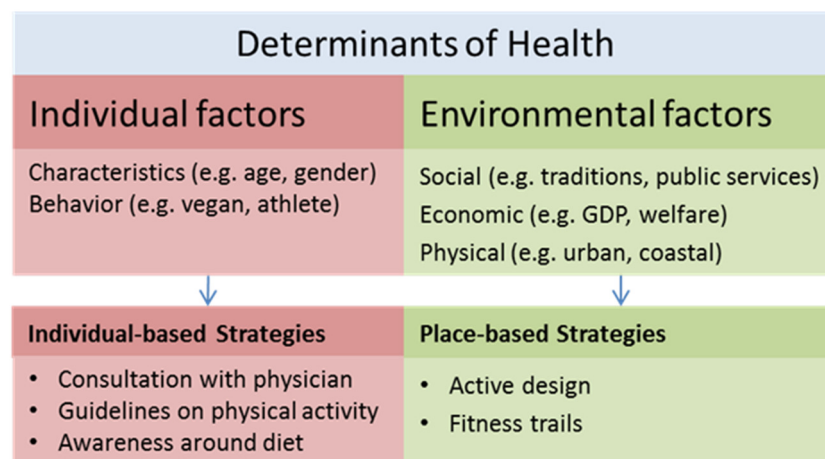


Figure 3: Determinants of health and associated intervention and prevention strategies

Strategies to improve population health generally focus on “individual-based” approaches (Koohsari, Kaczynski, Giles-Corti, & Karakiewicz, 2013) and the formal healthcare settings (Moon & Gillepsie, 1995). In a formal healthcare setting, a patient faced with health problems would be prescribed a treatment by a health professional. This one-on-one professional advice uses the trusted relationship between a health

professional and her/his clients (or patients) to raise awareness regarding dietary and physical activity recommendations to eventually establish healthier habits. An example of non-formal, individual-based healthcare would be receiving a recommendation from a family-member or friend that may have faced similar health issues.



Figure 4: Forms of healthcare strategies to alleviate heart disease

Other strategies make use of the community to impact our behavior. These strategies are referred to as "place-based" approaches towards health prevention (Koohsari et al., 2013) and one of their benefits lies in the ability to reach more individuals without direct intervention of health professionals, which can be more cost-effective and reach more socio-economic groups at once. Parks for example, can positively influence behavior by increasing our engagement in physical exercise, without direct consultation with a health professional. Planning parks in a way that encourages communities to be more physically active would be an example of a formal, place-based healthcare approach, which is where my dissertation fits.

In 2010, the first congress organized by the Healthy Parks Healthy People international movement was held in Melbourne, Australia. In 2011, the U.S. launched its own national Healthy Parks Healthy People program, which is coordinated by the National Park Services (NPS). This program aspires to reframe the role of public open space to become a strategy for health prevention. In the U.S., the availability of financial

resources to support various kinds of open space and conservation programs has increased over the past 20 years (Walls, 2009). The U.S. government also reports the need to improve access to facilities supporting physical activity and to the built environment (e.g.: sidewalks, bike lanes, trails and parks) as it recognizes the positive effect on physical activity (U.S. Department of Health and Human Services, 2010). Although government administrations and health organizations recognize the need to look at the design and planning of public infrastructure to improve community health, few studies evaluate the impact of better and effective planning on community health. Also, there does not seem to be a consistent approach to evaluate access to certain services, including access to parks. Many studies that use spatial accessibility methods also fail to report strengths and limitations of their approach, which is an important discussion to include if we want to slowly generate a consensus around appropriate methodologies to use in such studies.

3. Spatial Justice of Park Access

A growing body of literature has emerged to evaluate whether or not access to public open spaces is equivalent across socioeconomic and ethnic/racial groups and between rural and urban areas (e.g., Parks, Housemann, & Brownson, 2003; Dai, 2011; Wolch, Byrne, & Newell, 2014). Some findings are consistent across a number of papers, suggesting that access to public open spaces tends to be unequal. Dai (2011) quantified access to urban green spaces using GIS and evaluated disparities among racial/ethnic and socio-economic groups in Atlanta, GA, using linear regression. His study confirms significantly poorer access to urban green spaces among neighborhoods with a higher concentration of African Americans. Additionally, a poorer access to urban green spaces

was found among socioeconomically disadvantaged areas (Dai, 2011). In a cross-sectional study by Parks, Housemann and Brownson (2003) it is shown that U.S. residents living in lower income areas were reporting lower levels of physical activity. Furthermore, they found that levels of activity were highest among suburban residents and lowest among urban and rural residents, which they reported to coincide with other national cross-sectional studies (Parks, Housemann, & Brownson, 2003). In light of these findings, increasing the level of physical activity for individuals living in rural and urban areas should form a public health priority.

Local government might think about developing planning strategies that try to alleviate these disparities by introducing new public open spaces in disadvantaged neighborhoods. Unfortunately, the introduction of public open spaces in lower income neighborhoods has been reported to have an undesired gentrification effect. Indeed, public open spaces increase the desirability of neighborhoods in general, resulting in land and housing values to rise and eventually pushing current residents out of their neighborhoods which have become unaffordable (Wolch, Byrne, & Newell, 2014). This contradiction is referred to as “ecological gentrification” by Dooling (as cited by Wolch, Byrne, & Newell, 2014, p. 239). Nonetheless, in many studies and initiatives, the positive economic effect of public open spaces is embraced and used as an argument to attract more (financial) support for programs supporting the development or expansion of public open spaces. For example, the Association of Park Districts of Illinois (IAPD) is openly supporting private developers to implement open spaces in every new subdivision or housing development, using the increase of property values as main argument (IAPD, 2006). This practice might improve the overall availability of open space, yet exacerbate

already existing disparities in access to such spaces and ultimately on physical activity.

Today, many local governments and planners are faced with this intricate matter of providing health benefits to disadvantaged neighborhoods by increasing the availability of public open space yet understanding the gentrifying effects of these developments.

The “just green enough” planning strategy (Curran, & Hamilton 2012) attempts to remedy the undesired gentrification effects of new urban green space. In that respect, the authors are predominantly addressing the social injustice linked to this issue instead of trying to influence the population in general. This strategy is driven by concerns voiced by the community itself, its needs and wishes rather than taking conventional approaches towards urban design or ecological restoration. In a Brooklyn community, the “just green enough” approach focused on stream cleanups and development of small-scale green spaces along a creek nearby residences of the working class (Curran & Hamilton 2012). The involvement of the community is central to this planning / intervention approach. The authors suggest that some current planning practices have a narrow focus on the aesthetics rather than on the functionality of places. Neighborhoods with a high concentration of minorities and a lower income population often face other, more stringent environmental injustices than access, availability or aesthetics of public open spaces. Residing in closer proximity to polluted sites and industrial sites can have more effect on health than living on a remote distance from public open spaces. Yet, oftentimes low income and minority populations are concentrated in areas closer to industrial sites because they work at these industries and thus these unhealthy environments are part of their livelihood (Curran & Hamilton, 2012). Cleaning toxic waste sites and other polluted sites has a higher priority regarding the health of these residents. Through their activism

and contestation against environmental gentrification, the community in Brooklyn was able to shed an open mind on this notion of green and healthy (Curran & Hamilton 2012). The community was closely involved in the planning and implementation of the planning project, ensuring that the needs and concerns of the residents are met. Their approach suggests a more qualitative approach when analyzing or evaluating interventions.

Koohsari, Badland and Giles-Corti (2013) argue that planners have a better understanding of urban issues and would be able to target different levels of the socioecological model more effectively. They suggest that a more top-down approach would result in better health outcomes and provide an overall increase in physical activity not just for particular neighborhoods, but for a city as a whole. Curran and Hamilton (2012) on the other hand, suggest that a bottom-up approach is more appropriate to deal with this public health issue. Few studies so far, have tried to include the local population into community planning efforts and I agree that the local context needs to be understood to make effective changes in the built environment. In this study I address this gap in the literature by surveying park visitors to get a sense of the local perceptions on access to parks in order to provide more meaningful recommendations at the county level.

4. Concept of Access

The concept of access to any good or service has been studied for a long time and is being used in many different settings. From access to education to accessing drinking water, the concept of access is important in many studies that try to identify population disparities in the distribution of goods or services. Donabedian (1973) distinguished two different aspects of access; geographic and socio-organizational access. Geographic access refers in many cases to a physical distance that needs to be traveled. Socio-

organizational access refers to the effort required to obtain the right services. The availability of specialized services at a facility or the language in which services are provided would be examples of socio-organizational access. Khan & Bhardwaj (1994) further investigated the question of access within a healthcare context and unfold this concept into two dichotomous categories, namely (1) potential versus realized access and (2) spatial versus nonspatial access. Guagliardo (2004) refers to these two dichotomies respectively as “stages” and “dimensions”. The first “stage” develops when there is a population in need and a willingness to provide a service for it. It is called “potential” access because there is potential to fill a need. The final stage of access is “realized” access and occurs when all barriers to provision are overcome (Guagliardo, 2004, p. 4). In other words, the need has been met. The other dichotomy, namely spatial vs. nonspatial access, is referred to by Guagliardo (2004) as “dimensions” of access. This dichotomy was already brought up by Donabedian (1973). In this study potential and realized access are measured using models of accessibility that incorporate both spatial and nonspatial dimensions of access.

4.1. Models of Spatial Accessibility

Spatial accessibility to a good or service (i.e. commodity) is defined as a function of the availability (supply) of this commodity and of the costs (e.g. distance, price) separating the population in demand from this commodity. Several spatial accessibility metrics are derived from so-called “container” approaches. The original container method identifies whether a commodity (e.g. parks) is located within some geographic unit (e.g. census block, block group, or tract). This binary way of assessing accessibility is very limited as geographic, administrative boundaries are artificial barriers to access; in

reality, individuals are often able to access parks in adjoining units. Moreover, these administrative boundaries often divide a study area into subunits of varying sizes. Having “containers” of varying sizes further impacts the results: the location of a certain commodity has a lower probability to fall within smaller containers than to fall within larger ones. As a consequence, smaller geographic units have a higher probability to receive lower accessibility scores when using container methods. Regardless of these shortcomings, this approach is still employed in neighborhood-scale environmental justice and social equity-based analyses (e.g. Vaughan, Kaczynski, Stanis, Besenyi, Bergstrom, & Heinrich, 2013), mainly because of its simplicity and ease of implementation.

Buffer analysis (Nicholls, 2001), kernel density estimation (Moore et al., 2008), and network constrained service area methods (Miyake et al., 2010) have all been proposed as alternatives to container methods (Cromley & McLafferty, 2012). These latter methods fall under the classification of “coverage” models (Talen, 2003) as they assess the population that falls within a specified distance from a commodity. The population falling within this defined distance is considered having access to (or being “covered” by) this commodity. The results of coverage models are often driven by the definition of this distance. Because it is often difficult to set an appropriate distance at which a service would stop being accessible, the results of coverage models could be considered to be somewhat arbitrary.

The distance-threshold limitation of coverage models motivated the use of Thiessen polygons (Boone et al., 2009; Sister et al., 2010), which generates polygons (often asymmetric in shape) delimiting the area of influence or so-called “service area” of

each commodity. None of the polygons overlap each other, so that a population center (demand point) is always located within exactly one service area. Summing the population falling inside each Thiessen polygon gives a sense of total demand for each commodity. This technique makes it possible to estimate potential crowding at certain locations and helps identify underserved areas. It also assumes that the population will utilize a facility that is located closest to their residence, however this assumption may not be realistic in the case of public parks, since individuals may interact with larger regional parks located further away (Boone et al., 2009; Sister et al., 2010).

4.2. Models Incorporating Spatial and Non-Spatial Factors

Realizing that coverage methods have shortcomings in estimating access to parks that might have different service areas (e.g. neighborhood vs. regional parks), more complex measures have been developed. Based on literature that has found empirical support for the idea that park amenities play a role in attracting visitors willing to travel a greater distance beyond their neighborhood park (McCormack, Rock, Toohey, & Hignell, 2010), gravity-based models use notions of attraction and friction to measure our willingness to travel to a particular location. These models offer some conceptual improvement upon the simpler metrics. On the other hand, gravity-based models are continuous metrics that incorporate the full range of destination options. This tends to produce an overly smoothed accessibility landscape (Luo & Wang, 2003; McGrail & Humphreys, 2009).

Floating catchment area (FCA) methods represent another category of accessibility measurements and were initially conceived in a healthcare context. In the FCA approach, a catchment area is estimated around a service facility based on some

maximum distance that individuals are likely to travel; any demand point within that catchment area is deemed to have access to that facility, while all others do not. It is a dichotomous technique as contrasted to the continuous, gravity measures (Luo & Wang, 2003; McGrail & Humphreys, 2009). Combined measures such as the two-step floating catchment area (2SFCA) method with a distance decay function have been proposed as a superior alternative for identifying potential disparities in accessibility (Dai, 2011). This measure, widely employed in the healthcare literature, specifies a given catchment distance around a facility and evaluates both supply and demand (i.e. attraction and crowding) within that region. A major limitation of the 2SFCA is that each catchment area is set at a fixed distance, regardless of the type of facility, which does not adhere to the way that some commodities are planned.

The limitations of previous methods have prompted a number of improvements including the incorporation of a distance-decay parameter within each catchment – to either the population or supply side (Dai, 2011; Luo & Qi, 2009) and the use of variable-width catchments (Luo & Whippo, 2012). Luo and Whippo (2012) suggested the use of variable catchment sizes in a healthcare-specific context. In their approach, catchment sizes are incrementally expanded until a minimum, specified provider-to-population ratio is reached. While this may make conceptual sense in assessing healthcare access, determining an optimal park-to-population ratio is much less intuitive. One challenge when using this approach lies in the calibration of the park-to-population ratio.

Two recent methodologies have been proposed as theoretically superior towards assessing park access as compared to well-established techniques, such as the 2SFCA method and its derivations. These include the population-weighted distance (PWD)

method developed by Zhang, Lu and Holt (2011) and an “accessibility in the context of spatial disparity” measure (ASD) put forth by Lee and Hong (2013). Both of these measures are based on gravity-based spatial interaction considerations whereby larger, more attractive parks are expected to draw a larger share of the population. Zhang et al.’s (2011) national study and Vaughan et al.’s (2013) local study (in Kansas City, MS) on park accessibility both incorporated notions of choice sets; modeling supply and demand as a probability function (based on Huff’s (1964) market area segmentation model). Lee and Hong’s (2013) ASD approach involves discretizing the urban area into a continuous grid and computing a gravity-model inspired supply-demand ratio. The distance-decay parameter helps to distinguish the intended usage and expected demand for various types of parks: neighborhood parks have a service coverage area of 250 meters, medium sized parks have a service area of 1000 meters, and parks of a larger size do not have a coverage limit.

4.3. Modeling Access to Parks

With a specific set of functions in mind for each park, local Park and Recreation Departments plan and anticipate a certain level of demand. Neighborhood parks are intended to serve residents living in their immediate vicinity; they are typically smaller and often provide limited parking accommodations. Regional parks on the other hand, are larger, offer more or distinct amenities and are planned to attract residents from further away. In planning, differences in service levels are also referred to as “normative standards”. Páez et al. (2012) define “normative accessibility” to reflect a level of accessibility considered to be acceptable from the viewpoint of a planner or policy maker. The authors distinguish this from the notion of “positive accessibility”, which they define

to be the level of impedance perceived acceptable and reasonable by the individuals themselves (Páez et al., 2012, p. 142). For modeling purposes, a critical differentiation must be made with regards to the type of accessibility being measured. If the specified distance is intended to reflect actual travel behavior, then a positive approach must be apprehended, which often requires surveying the public. On the other hand, in a normative approach, a certain level of access is set and reflects the distance at which planners and policy-makers have agreed all individuals should have an acceptable access to a particular facility. These notions – normative and positive accessibility, are similar to the notions of potential and revealed accessibility that have been formulated by Khan and Bhardwaj (1994) in the context of access to healthcare. They refer to “potential access” as the prescribed level of access provided by the supply and refer to “revealed access” as the level of access actually experience by the demand.

In my dissertation, I evaluate the potential (or normative) accessibility to public parks using the Variable Floating Catchment Area (VFCA) method (Dony, Delmelle, & Delmelle, 2015). This model incorporates a flexible attraction index based on size, and amenities of a park, which defines the catchment size of each park. To date, the attraction parameter has either narrowly focused on acreage, or treated all parks equally. For each demand point the level of accessibility to a park is weighted based on the potential crowding at the park measured by a park-to-population ratio. Finally, I compare spatial accessibility of each demand point between four modes of transport; (1) driving, (2) public transit, (3) bicycling and (4) walking. Although network-constrained distances are widely recognized as superior approximations of travel as compared to their Euclidean counterpart (Gutiérrez & García-Palomares, 2008), the ease of computing Euclidean

distances has contributed to their persistent use. The inclusion of alternative modes of transportation in accessibility studies is a burgeoning field of study. Assessment of accessibility by public transit has been implemented by Delmelle and Casas (2012) and Mavoa, Witten, McCreanor and O'Sullivan (2012), while Reyes, Páez and Morency (2014) examined pedestrian access based on revealed walking trip lengths. Recently, Mao and Nekorchuk (2013) proposed a multi-modal 2SFCA method where the specified catchment area is modified according to a designated transport mode. Along the same vein, in the VFCA, the size of the variable-width catchments is discounted according to the mode of transportation. Another contribution my dissertation makes to the literature is the use of social surveys to evaluate revealed access to parks in Mecklenburg County.

5. Monitoring Approaches for Rapidly Growing Urban Areas

Broader regional changes such as adding a new highway stretch, adding a new light rail line, adding a new ballpark, adding a new bike share program, experiencing fast real-estate development can make the overall context of a city shift pretty quickly. In Charlotte, NC, all the examples listed have happened in the past five years; the region is experiencing a rapid urban growth. This population growth can lead to a change in needs, life-styles and opinions. In this context, it is important that urban leaders develop a clear vision and make decisions in accordance with the evolution of their urban area and its changing population. That includes taking into account needs of the incoming population. In consequence, it is extremely important for those urban centers to acquire tools that can collect data quickly (or continuously) and can automatically interpret incoming data in order to make planning decisions that keep up with the pace at which the region is changing.

The use of social media and other online commenting platforms to make real-time policy recommendations. In their book, Ciuccarcelli, Lupi, and Simeone (2014), explore social media as a source of knowledge for urban planning and management. They argue that time-based and geo-located social media data should be complemented by the more traditional data collection methods such as surveys to provide more complete insights into the social life of urban spaces. Garcia Esparza, O'Mahony, and Smyth (2010) make the observation that real-time data from the web is far from structured, but offer an additional and valuable source of data that can improve recommendations for decision-making. As an example, Barry (2014) used photographs shared by online users of Flickr (a photo sharing platform of Google) to better understand public perceptions of livestock grazing in public spaces. Interestingly, this study showed that opinions and concerns shared on Flickr provided a perspective that is seldom expressed at public meetings or in surveys. Social media has been described as a ubiquitous tool for social interaction. While most new users are between 16–24 year olds, the use of social media by individuals between 25 and 45 years old has increased in recent years (NM Incite, 2012). In a study of Afzalan and Muller (2014), social media was tested as a communication tool to improve public participation in the planning of local green spaces. They concluded that these web-based communication tools helped significantly to create a dialogue and to build consensus. Moreover, it extended participation from groups that do not typically engage in planning processes.

Sentiment analysis (a form of data mining), has been increasingly used to measure attitudes towards certain topics on social media, especially from tweets. For example, Paul and Dredze (2011) followed a number of Twitter users and tried to extract messages

that were related to disease symptoms. To those tweets, they linked diseases these users could likely be diagnosed with, such as allergies, obesity or depression and mapped the emergence of certain diseases at the U.S. state-level. Twitter messages have also been used as a predictor for stock markets (Bollen, Mao, & Zeng, 2011) and to better understand the public opinion regarding certain topics, such as vaccination (Salathé, & Khandelwal, 2011) or the Affordable Care act (Wong et al., 2015).

Based on a dictionary in which each word is classified as negative or positive - a sentiment dictionary, it is possible to derive the sentiment of a sentence based on the words it constitutes (Wilson, Wiebe, & Hoffmann, 2005). The average sentiment scores from users tweeting about a certain topic determines the overall public sentiment towards that topic.

In essence, web-based citizen data and social media are important avenues to explore in the context of urban planning and decision-making. They have the potential to extend current sources of data rather than replace them. Finally, these data sources offer constant inflow of citizen data which can help decision-making processes in rapidly growing urban areas, such as Mecklenburg County.

STUDY AREA: MECKLENBURG COUNTY

The Mecklenburg County Department of Park and Recreation manages 210 parks and facilities, accounting for over 21,000 acres of parkland (see Figure 5).

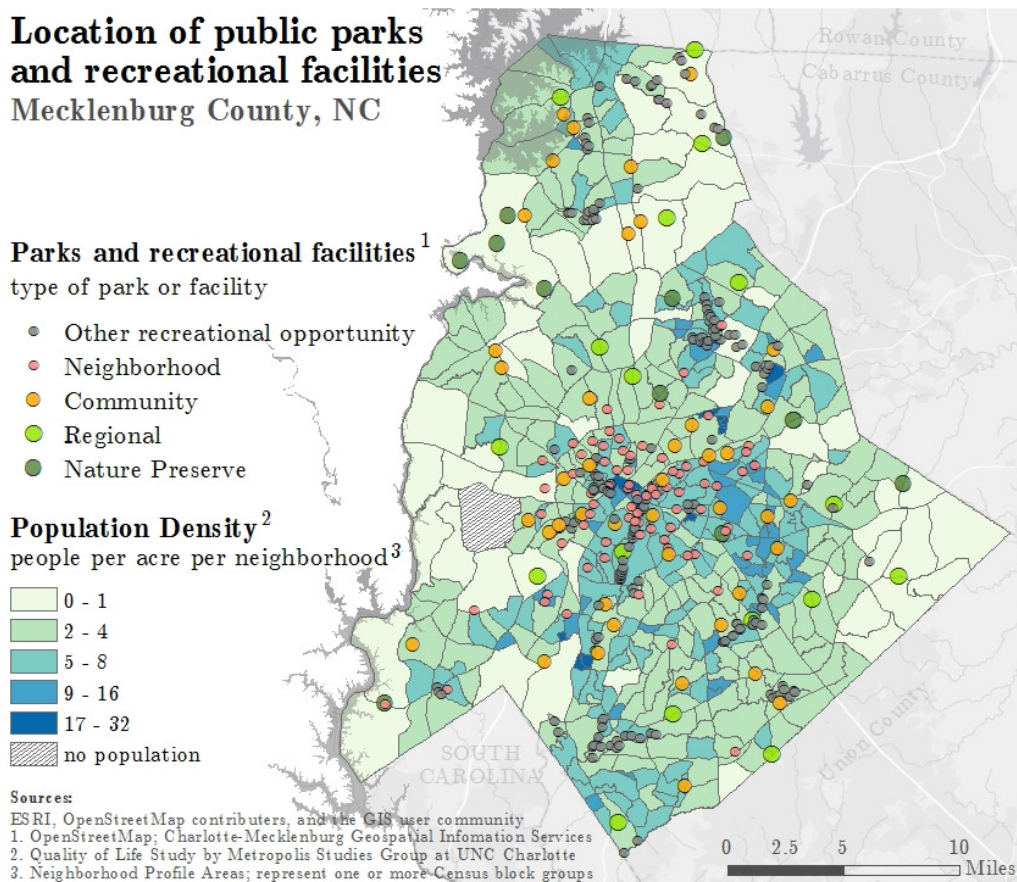


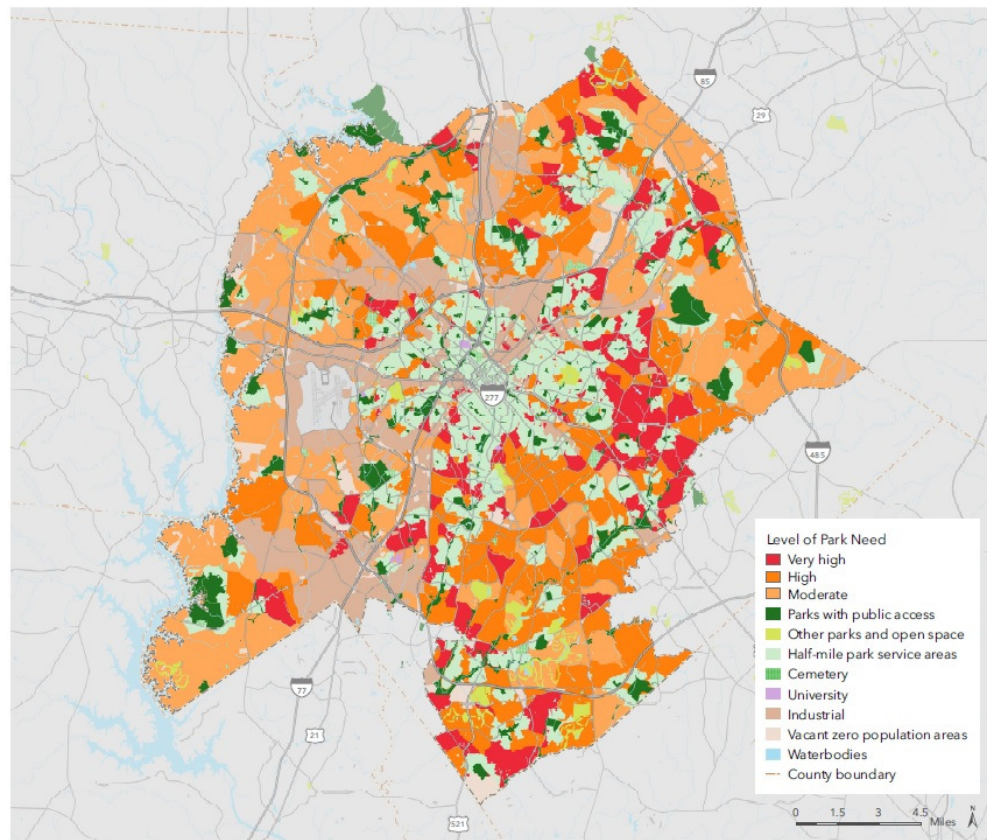
Figure 5: Parks and recreational facilities in Mecklenburg County.

The Department of Park and Recreation distinguished 5 difference public park types, namely neighborhood parks, community parks, regional parks, nature preserves and recently they added urban parks. Neighborhood parks are meant to be proximity parks, they are usually small in size and do not have large parking space. Community

parks have more amenities, are larger, often have a recreational center attached to them and provide parking. Community events often take place at these parks. Regional parks are large and offer many amenities including trails. Many learning activities for youth are organized at regional parks. Some nature preserves are accessible to the public and offer trails. However, many of them are there for conservation purposes or watershed quality assurance. Finally, Romare Bearden park is the first and only urban park so far in Mecklenburg County. It is located in the business district of Charlotte nearby the baseball stadium in a walkable area. Figure 5 also shows locations of other recreational opportunities. These include entrances to greenways, recreational centers, public golf courses and pools.

The Trust for Public Land, a U.S. non-profit organization, estimated that over \$80 million in health costs were saved in 2009 alone, thanks to the use of parks by residents of Mecklenburg County (TPL, 2010). Since 2012, the Trust for Public Land ranks U.S. cities based on scores of public park availability. Every year since their first publication, Charlotte (NC) has ranked at the very bottom. The latest scores published in 2016 rank Charlotte 95 out of 98 U.S. cities. The total acreage of public parkland in Charlotte exceeds the median for all cities, however the percent of the population that has access to a park within ½ mile is one of the lowest. The latter carries the most weight in the overall park score – which is understandable, and is one of the main reasons why Charlotte receives a consistently poor park score. Figure 6 shows the level of need for park in areas of Charlotte. From this map, neighborhoods in the eastern part of the City of Charlotte seem to have a high (dark orange) or very high (red) need for public parks. The southeastern part seems to show a cluster of very high need areas, while neighborhoods

south of the business district (south of I-277), which consists of the wealthiest population in Mecklenburg County, seems to fall in a high need area as well.



The Trust for Public Land 2016 *ParkScore*® index CHARLOTTE, NORTH CAROLINA

Figure 6: Level of need for parks in Charlotte (TPL, 2016)

In 2014, the director of Mecklenburg’s Park and Recreation, Jim Garges, was interviewed about the city’s poor park score (interview by McShane, 2014). Jim Garges admitted that Mecklenburg County needs to improve its park system and that they have been in a period of transition regarding public parks since they merged the services by the City of Charlotte with those of the County in 1992. Additionally, Jim Garges underlines that there are no guidelines that require developers to include park land in their

plans, even though this could be required under North Carolina state law.

The Quality of Life Study for Charlotte and Mecklenburg County – which started in 1993 as the *City Within A City Neighborhood Assessment*, provides neighborhood level information on social, housing, economic, environmental and safety conditions. In 1998, The University of North Carolina at Charlotte partnered with the Charlotte-Mecklenburg Planning Commission to continue and expand this assessment of neighborhoods. Their first report, renamed to the *Charlotte Neighborhood Quality of Life Study* was published in 2000 and has been published every other year since. In 2012 however, instead of a report, the format of the Quality of Life Study was transformed to an interactive dashboard called the Quality of Life Explorer.

Since the launch of the Quality of Life Explorer, proximity to parks and recreation is included to be one of the neighborhood quality of life indicators. Figure 7 shows the percentage of the population within each neighborhood that are within a 0.5-mile radius from a public park or recreational facility (see Figure 5 for a map showing the location of public park and facilities). Comparing this map with the one published by the Trust for Public Land (see Figure 6); areas of high and very high need do not necessarily coincide with neighborhoods that show a low percentage of household within 0.5 mile from a park or recreational facility. However, the differences between both studies regarding access to parks and recreation are not surprising. They are the perfect demonstration that two different definitions of “access” and different methods to measure access can lead to very different results. In the next section, I discuss the notion of access and provide a review of methods to measure spatial access to parks and recreation.

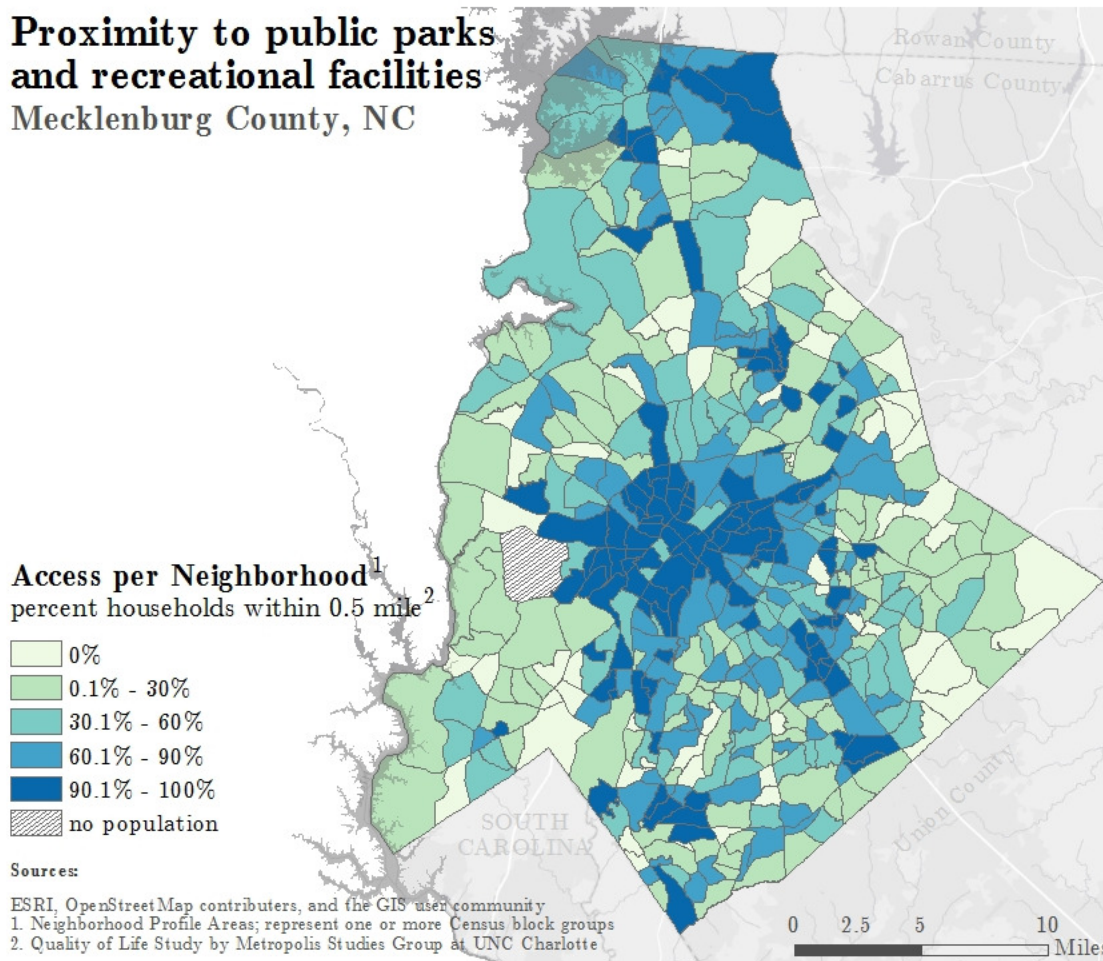


Figure 7: Proximity to public parks and recreational facilities in Mecklenburg County (Quality of Life Study, 2015)

One of the major issues with the outcomes from both the TPL and the Quality of Life Explorer, are the parks and recreational facilities that were taken into account. Both include parks managed by the Department of Park and Recreation, which is not a complete list of available parks in the County. In Mecklenburg County, many homeowner's associations manage their own outdoor recreational facilities (e.g. tennis court, pool, playground). Figure 8 shows the location of parks and recreational facilities that are managed by organizations other than the Department of Parks and Recreation. Using a combination of data from tax parcels and from OpenStreetMap, I was able to

extract an additional 636 locations where park and recreational facilities are available, which is three times the number of public parks in Mecklenburg County. These parks are not public as they are often only available to the residents of a homeowner's association.

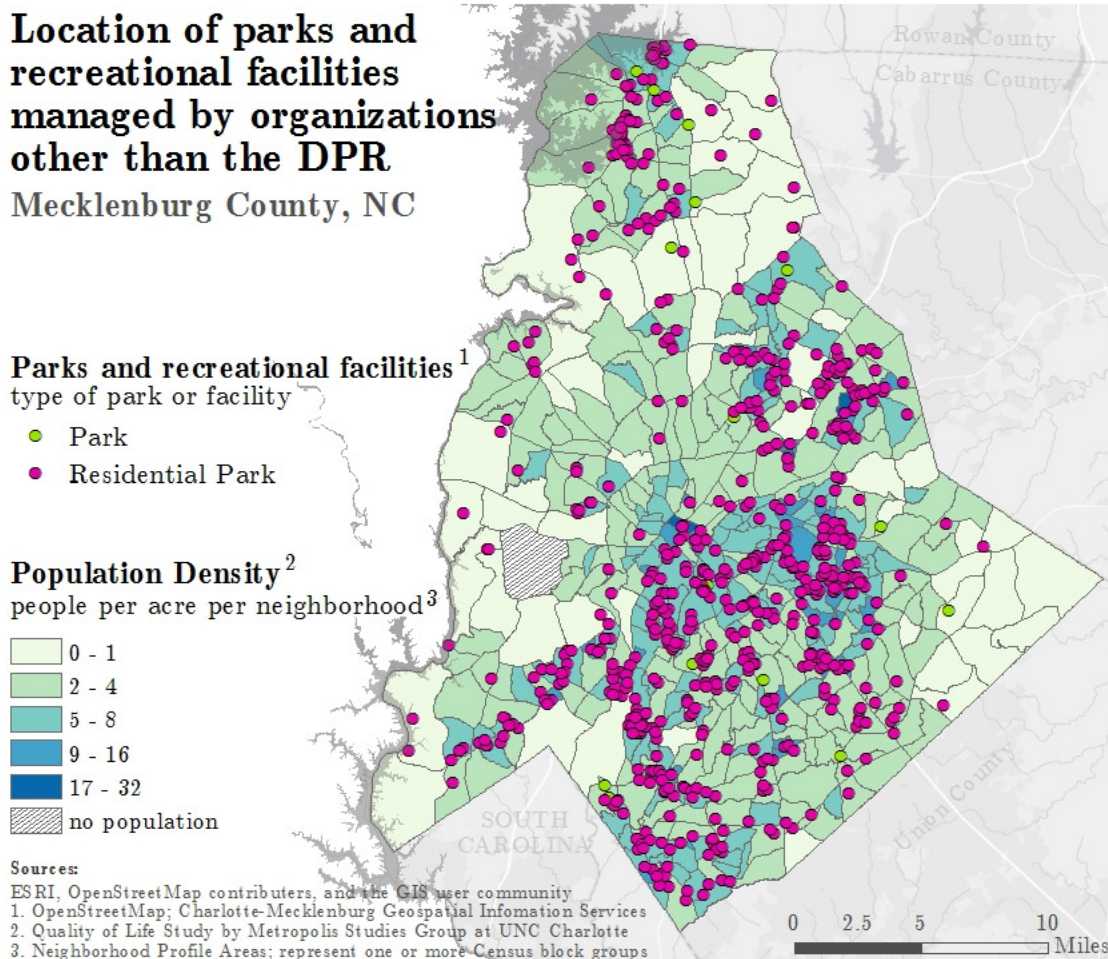


Figure 8: Location of parks and recreational facilities managed by organizations other than the Department of Park and Recreation.

In my dissertation, I include this data source to further evaluate access to parks and recreation in Mecklenburg County, which is an important contribution to the current literature.

RESEARCH OBJECTIVES AND CONTRIBUTIONS

The objectives of this dissertation are detailed in this section. Each research objective will be accompanied by its associated research contributions. Additionally, specific research questions are formulated which are addressed in this dissertation to achieve the overall research objective.

1. Research Objective 1

The first objective is to develop a model to measure spatial accessibility that can take into account certain planning standards and the local context. The goal is to use this model to measure spatial accessibility to public parks for each block group in Mecklenburg County and to compare the results with findings from the Trust for Public Land (TPL) and from the Charlotte-Mecklenburg Quality of Life Explorer (QOL).

1.1. Research Contributions

This objective contributes to the development of modeling approaches that can take into account the local context, which are critical for planners and policy-makers and transferable to a number of other applications (e.g. access to food, health care, jobs, information). The comparison with studies from the TPL and QOL is important because both studies evaluated access to public parks in Mecklenburg County and their outcomes have influenced local planners and policy-makers. By way of comparison, this study evaluates the trustworthiness of their findings.

1.2. Research Question 1

Using a novel derivation of the Variable Floating Catchment Area method (VFCA) – a state-of-the-art spatial access model; which areas of Mecklenburg County have the poorest and greatest spatial access to public parks?

1.3. Research Question 2

Are results from the VFCA method comparable to those of the Trust for Public Land and from the Charlotte-Mecklenburg Quality of Life Explorer?

2. Research Objective 2

The second objective is to survey perceived access to parks among Mecklenburg County public park visitors and improve our understanding of individual's travel behavior to public parks and their utilization of parks for physical exercise. Additionally, the goal is to map the availability of parks using insights from these surveys.

2.1. Research Contributions

Empirical evidence about the use of public parks as a place for regular physical activity may support park investments under prevention strategies for heart disease. In addition, since Charlotte scores poorly on park access compared to other U.S. cities (TPL, 2016), this study can validate this dissatisfaction by asking park visitors how they perceive their access to parks. Additionally, empirical evidence on park visitor's travel behavior and their satisfaction regarding access to parks provides a way to evaluate and revisit current normative planning standards. Fourth, the equality of access to public parks are visually identified on maps showing park availability within the local context. The latter makes this study more valuable to local planners and policy-makers.

2.2. Research Question 3

How do visitors rate their access to parks in Mecklenburg County? Are these self-reported access ratings associated with (1) the area in which individuals live, (2) their socio-economic status, (3) their available transportation options, (4) their Body Mass Index (BMI) or (5) the availability of a park within a 10-minute walking distance from

their residence?

2.3. Research Question 4

How much time are visitors willing to travel to a public park? Is travel time associated with (1) the park type they are visiting, (2) their available transportation options or (3) their socio-economic status?

2.4. Research Question 5

How frequently do park visitors engage in regular physical activity at public parks? Is this frequency associated with (1) the park type they are visiting (2) their socio-economic status, (3) the availability of a park within a short walking distance from their residence or (4) their membership with a gym?

2.5. Research Question 6

After re-evaluating outcomes from Q1 and based on data from visitors at public parks in Mecklenburg County, which areas of Mecklenburg County have the poorest spatial access to public parks?

3. Research Objective 3

The third and final objective is to explore capabilities of social media and online commenting platforms such as Foursquare or Google Maps to quantify the public opinion about public parks in Mecklenburg County.

3.1. Research Contributions

Monitoring visitor's satisfaction of parks with public surveys is costly and time consuming. Therefore, social media platforms are presented and discussed in this dissertation as a potential to provide additional data sources to urban planners in rapidly growing areas such as Charlotte (NC).

3.2. Research Question 7

Using messages left at park locations on social media platforms such as Foursquare and Google Maps and data mining techniques such as sentiment analysis, can satisfaction of park visitors in Mecklenburg County be monitored?

4. Assumptions

This study relies on a number of assumptions. First, I make the assumption that individuals living closer to parks will likely engage in physical activity more frequently compared to individuals who live further away. This assumption is based on studies suggesting a positive association between access to parks and individual's levels of physical activity (Bedimo-Rung, Mowen, & Cohen, 2005). Second, I assume that increased regular physical activity reduces the risk for non-communicable diseases, which is echoed in several studies indicating a negative relationship between regular physical activity and the risk for non-communicable diseases (Brownson, Boehmer, & Luke, 2005). Finally, both assumptions are linked, creating a chain of assumptions that form the basis for this study (see Figure 9 on page 30).

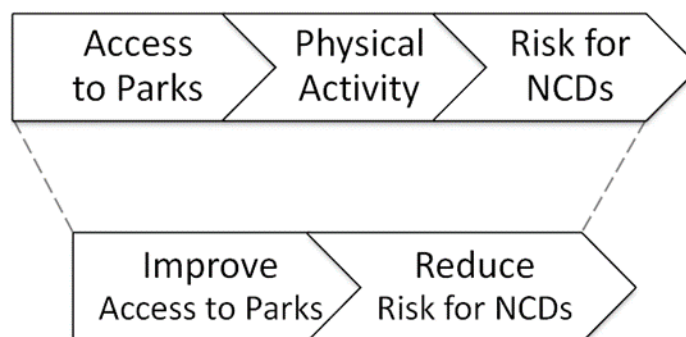


Figure 9: Causal assumption made to support studies that model and identify areas with low access to parks.

Based on this chain of assumptions, individuals living in areas where access to public parks is below a certain threshold can be diagnosed to have a higher risk for developing non-communicable diseases. As a consequence, improving access to public parks is a public health and social justice concern.

DATA

This study uses primary and secondary data sources. Surveys were filled out by park visitors 18 years and older at 18 selected public parks in Mecklenburg County with the aim to better understand local perception on access to parks, to identify travel behavior to parks and their use of parks for physical activity. This constitutes the only primary source of data for this study. The Department of Park and Recreation of Mecklenburg County shared a list of parks and recreational facilities they manage along with a number of their characteristics. Boundaries of Mecklenburg County block groups for the year 2010 were obtained from the U.S. Census Bureau and neighborhood characteristics defined by the Charlotte-Mecklenburg Quality of Life Explorer were used in this study as well. Finally, volunteered geographic information from Google Maps and Foursquare, which are online commenting platforms, were collected as well. Each data source is described in more detail in this section.

1. Data on Parks and Recreation from the Department of Park and Recreation

The Department of Park and Recreation (DPR) of Mecklenburg County shared the database they use, which essentially is a list of the parks they manage (in a Microsoft Excel format). The DPR distinguishes 4 different park types, namely neighborhood, community and regional parks as well as nature preserves. Figure 10 shows the location of neighborhood and community parks listed by in the DPR's database. Blue and red dots represent the location of a neighborhood park or community park, respectively. The size of each dot reflects the size of the park. From this figure, we see that a higher number of small neighborhood parks are available in and around the Charlotte business district (centrally located on the map), whereas larger neighborhood and community parks are

located further away from the city center.

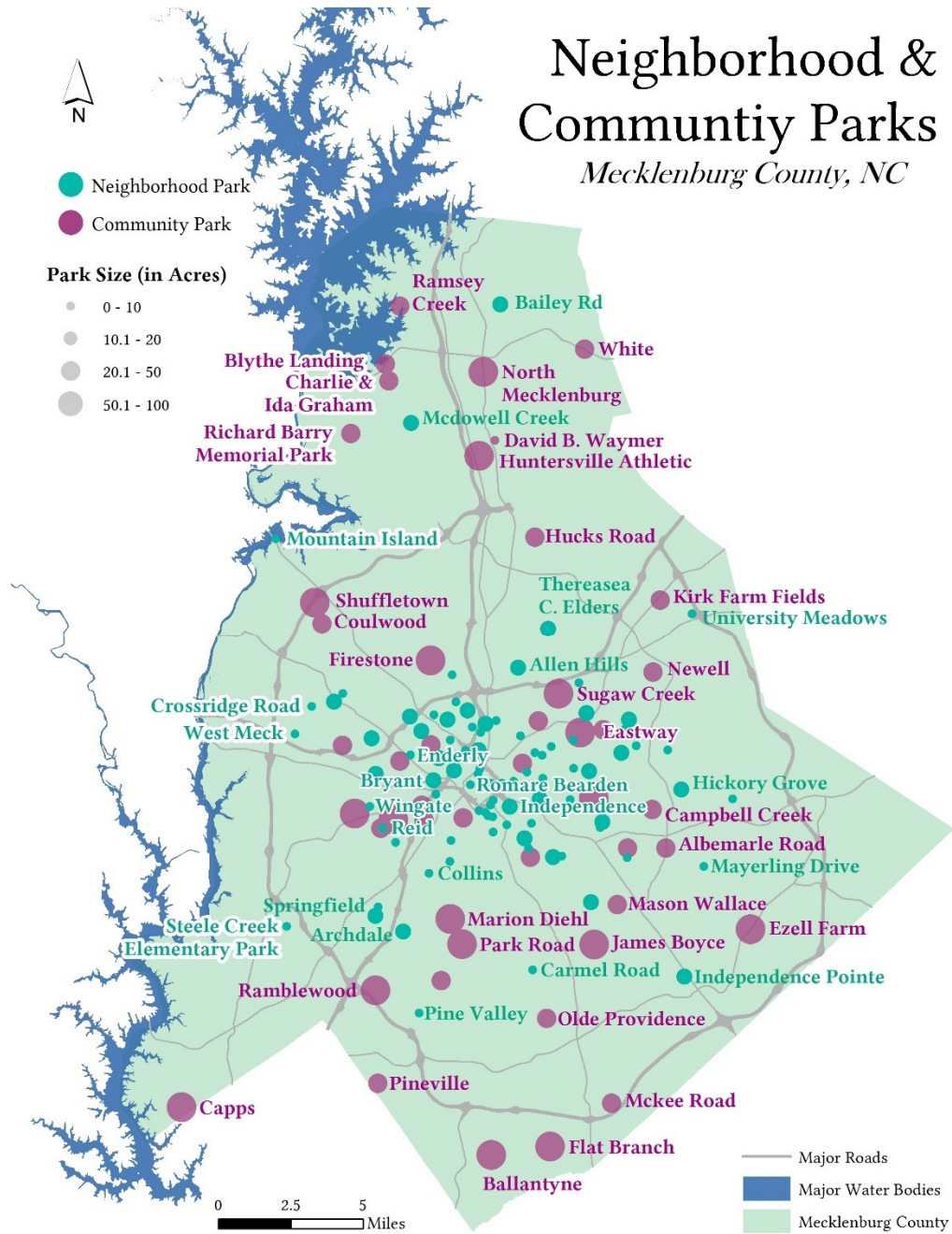


Figure 10: Neighborhood and community parks managed by or in collaboration with the Department of Parks and Recreation of Mecklenburg County.

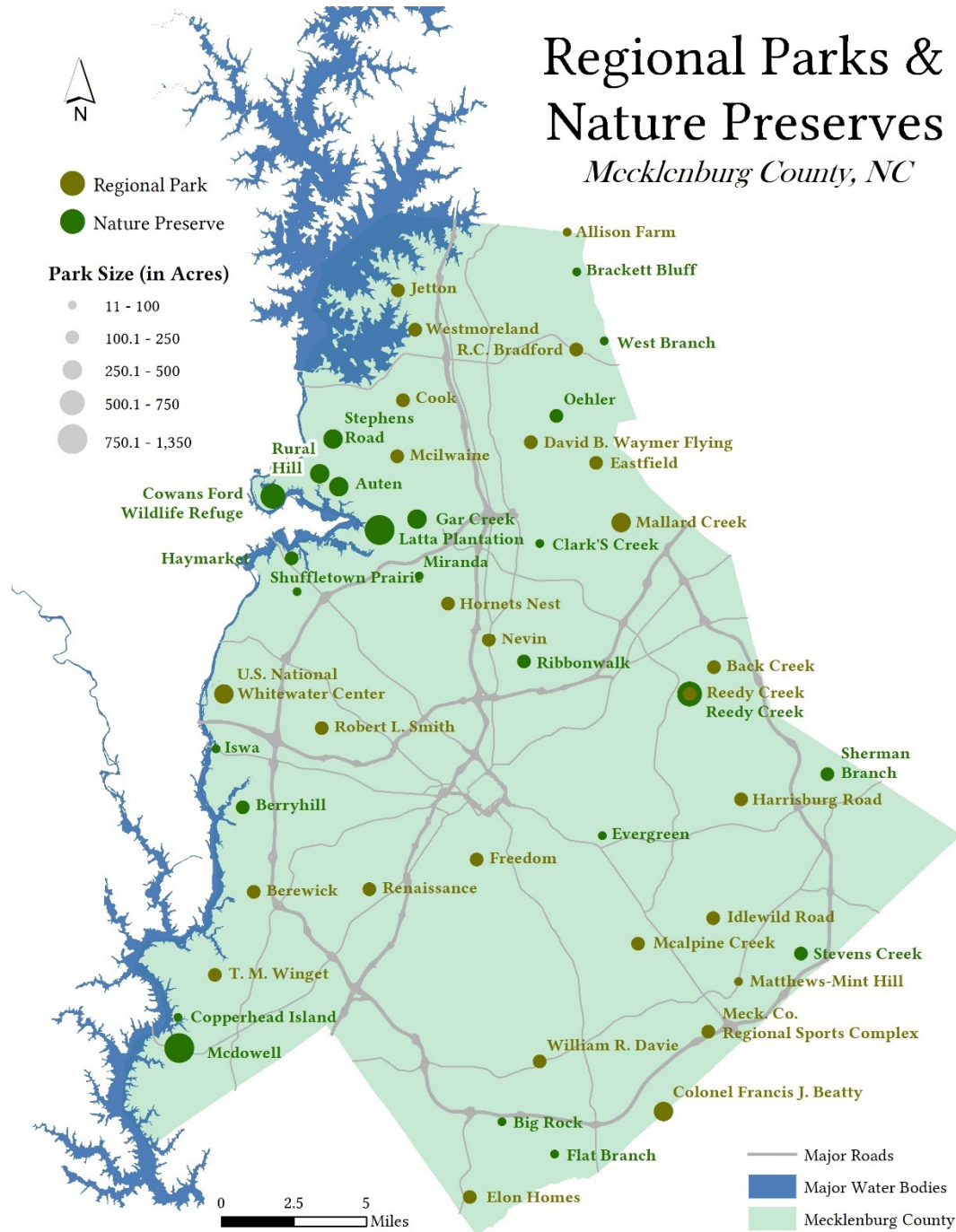


Figure 11: Regional parks and mature preserves managed by or in collaboration with the Department of Parks and Recreation of Mecklenburg County.

Regional parks and nature preserves are shown in Figure 11 and shows that these two types of park are not available near the city center and that the largest nature

preserves are located in the western part of the county (Cowans Ford, Latta Plantation and McDowell). Some parks are managed in collaboration with local governments; for example, Bradford regional park is managed in collaboration with the Town of Huntersville.

For each park in the DPR's database, the following information is available:

- (i) Park address
- (ii) Park type (e.g. neighborhood, community, etc.)
- (iii) Park status (developed, under construction, undeveloped, etc.)
- (iv) Park size (in acres)
- (v) Park amenities (amenity type and number per type)
- (vi) Park identifier

It is important to mention that the database provided by the DPR is not a complete and accurate inventory of public parks in Mecklenburg County. To generate a more complete and updated inventory of parks and recreational facilities, additional sources of data were used to complement the DPR's database.

2. Data on Parks and Recreation from Mecklenburg County

Mecklenburg County recently developed an online data portal, called "Open Mapping", where public data about the county is made available to the public for download (maps.co.mecklenburg.nc.us/openmapping). Their data is made available in a shapefile format, which is a file format compatible with Geographic Information Systems (GIS). Five datasets that were found under the "Park and Recreation" category on their webpage were used for this study: (1) park location points, (2) park property, (3) recreation centers, (4) greenways and (5) golf courses. Under the "environmental"

category the creeks and streams dataset was downloaded as well. Additionally, under the “impervious” category, the commercial impervious surfaces dataset was downloaded. Finally, under the “cadastral” category, the tax parcel with CAMA² data was downloaded as well. These eight additional sources of data were used to generate a more complete inventory of parks and recreational facilities.

2.1. Park Location Points

The “park location points” datasets contains the entrances (as points) of 251 locations. These locations include parks (neighborhood, community and regional), nature preserves, recreation centers, pools, special facilities, maintenance and administrative buildings, event venues and golf courses. A number of attributes are available for each location, including six attributes that were used for this study; (1) park name, (2) park address, (3) park type, (4) park status, (5) park size and (6) park identifier. The DPR’s database was linked to this dataset, making their non-spatial database a spatial one.

2.2. Recreation Centers

The “recreation centers” dataset contains the entrances (as points) of 33 recreation centers. These locations include recreation centers, recreational equipment on school sites and aquatic centers. A number of attributes are available for each property, including four attributes that were included in this study; (1) center’s name, (2) center’s type, (3) center’s address and (4) center’s status. This dataset was used to complement the inventory with recreation centers that do not appear in the DPR’s database nor in the park point locations dataset. Moreover, when certain attribute values of a recreation center such as its status or type were missing in the DPR’s database or in the park locations dataset, the attribute information of the recreation centers shapefile were used to fill the

² Real Estate Computer Assisted Mass Appraisal data

missing values.

2.3. Park Property

The “park property” dataset contains the boundaries (as polygons) of 257 properties. These properties include parks (neighborhood, community and regional), nature preserves, recreation centers, administrative buildings, historic sites, greenways and golf courses. A number of attributes are available for each property, including five attributes that were used for this study; (1) property name, (2) property type, (3) property status, (4) property acreage, and (5) property identifier. This dataset was used to complement the inventory with parks and recreational facilities that do not appear in the DPR’s database nor in the park point locations and recreation centers dataset. Moreover, when certain attribute values of a park or recreational facility such as name, size, status or type were missing in the DPR’s database or in the park locations dataset, the attribute information of the park property dataset were used to fill the missing values.

2.4. Golf Courses

The “golf courses” dataset contains the boundaries (as polygons) of 72 golf courses. These include greenway segments, overland connectors, greenway entrance segments, private segments, nature trails and greenway overlooks. A number of attributes are available for each segment, including two attributes that were used for this study; (1) course name and (2) owner’s last name. This dataset was used to complement the inventory with missing golf courses, especially public courses.

2.5. Greenways

The “greenways” dataset contains the outline (as lines) of 456 greenways segments. These include greenway segments, overland connectors, greenway entrance

segments, private segments, nature trails and greenway overlooks. A number of attributes are available for each segment, including four attributes that were used for this study; (1) segment's name, (2) segment's type, (3) segment's completion and (4) a detailed mention of the trail's start and end. This dataset was used to complement the inventory with greenways entrance points.

2.6. Creeks and Streams

The “creeks and streams” dataset contains the outline (as lines) of 3994 creeks and streams. This dataset was used to complement the inventory of greenways. Some greenways that are listed in the DPR's database, were not found in the greenways dataset. Since the name of greenways are usually attributed the creek or stream they are following, the outlines of missing greenways were extracted from this shapefile using the name of the creek provided in the DPR's database.

2.7. Commercial Impervious Surfaces

The “commercial impervious surfaces” dataset contains the boundaries (as polygons) of 119,153 impervious surfaces. These include buildings, paved surfaces “other” surface. Interestingly, the surfaces categorized as “other” in this dataset are surfaces such as warehouses, treatment facilities or recreational equipment. The latter is useful for this study as it contains the delineation of many recreational equipment such as tennis and basketball courts, outdoor pools, and so forth (see Figure 12). In this figure, the polygons that are part of the commercial impervious surfaces dataset are highlighted in red to demonstrate the type of information this dataset provides.

Mecklenburg County's Commercial impervious surfaces labeled as "other"



Turnberry

Cassington Court, Charlotte (28273)

Colonial Grand at Legacy Park

Legacy Park Drive, Charlotte (28269)

McDowell Creek Wastewater Treatment

4901 Neck Road, Huntersville (28078)

Figure 12: Content of the "commercial impervious surfaces" dataset made available by Mecklenburg County's Open mapping portal.

On the far left, a satellite image from the Turnberry neighborhood is shown. The red polygons delineate their tennis courts and their outdoor pool. To the right of the outdoor pool we can see a playground which is not delineated in the impervious surfaces dataset, since the surface is made of sand or grass. The Colonial Grand neighborhood is shown in the middle, where a basketball court and an outdoor pool is delineated by the commercial impervious surfaces dataset. Here too, a playground is available (next to the basketball court), but the surface is not impervious, which is why it is not found in the dataset. On the far right is an aerial photo of the McDowell Creek treatment plant. This was included to show that all polygons labeled "other" in this dataset are not only recreational equipment. To remove surfaces like treatment plants from the dataset, tax parcels were used to remove polygons that do not fall within residential parcels.

2.8. Tax Parcels with CAMA Data

The "tax parcels with CAMA data" dataset contains the boundaries (as polygons) of all parcels in Mecklenburg County. This dataset was used to label the impervious surface polygons (see previous section) and group recreational equipment together when

they fell in the same parcel. For example, the Turnberry neighborhood has two tennis courts and an outdoor pool (see Figure 12 on page 39). These two amenities belong to the same neighborhood; therefore, they were merged into one polygon. This way, multiple amenities belonging to one parcel are not accounted as two separate locations for recreation.

3. Data on Parks and Recreation from OpenStreetMap

OpenStreetMap provides volunteered geographic information such as streets, political boundaries, environmental boundaries (e.g. lakes) and other boundaries (e.g. parks and recreation) across the world. Mapzen, is an open source platform that makes the OpenStreetMap dataset available for download. On their metro extracts page (mapzen.com/data/metro-extracts) the OpenStreetMap data encompassing entire cities can be extracted in a shapefile format, which is a file format compatible with GIS. Their data for the Charlotte metropolitan area (which includes Mecklenburg County entirely) was used for this study. Their dataset includes an attribute called “leisure”, which include gardens, golf courses, nature preserves, parks, baseball fields, playgrounds, recreational grounds, sports centers, swimming pools and running tracks. Using this attribute, all parks and recreational facilities were extracted.

This dataset was used to complement the inventory with parks and recreational facilities that do not appear in the DPR’s database nor in the park point locations and recreation centers dataset. Moreover, when certain attribute values of a park or recreational facility such as amenities were missing in the DPR’s database the attribute information of the OpenStreetMap dataset were used to fill the missing values.

4. Public Park Inventory

Using the database shared by the DPR (see section 1.1), together with data made available by Mecklenburg County on their Open Mapping portal (see sections 1.2.1 through 1.2.6) an inventory of all parks, recreational facilities, public golf courses and greenway entrances was generated. It is important to mention that each source of data went through a data cleaning process and all features were normalized to link all data together. The data was cleaned using the python programming language (see APPENDIX I:), resulting in a park and recreational facilities inventory that is more complete compared to one of the sources individually (see Figure 5 on page 21).

5. Non-Public Park Inventory

Using the data extracted from commercial impervious surfaces (see section 1.2.7) and from the OpenStreetMap data (see section 1.3), an inventory of non-public parks was generated. This inventory reflects parks that are not managed by the Department of Park and Recreation. Instead they are likely to be managed by private associations such as homeowner's associations or local governments and do not overlap the inventory of public parks (section 1.4). Since these parks are not managed or monitored by one entity, it is harder to obtain a complete, reliable and updated database of locations together with their characteristics. The latter is an important limitation of this data source. It is important to mention that both sources of data went through a data cleaning process and all features were normalized to link all data together. The data was cleaned using the python programming language (see APPENDIX I:6), resulting in a park and recreational facilities inventory of non-public parks that is novel and thus adds to the innovation of this study. Figure 8 (on page 26) shows the locations of all locations in this inventory.

6. Park Reviews

Google Maps and Foursquare offer additional volunteered geographic information that is useful to this study. Using the “park” category on both platforms, park locations in Mecklenburg County were extracted together with their associated reviews from online users (see an example from Google reviewers in Figure 13).

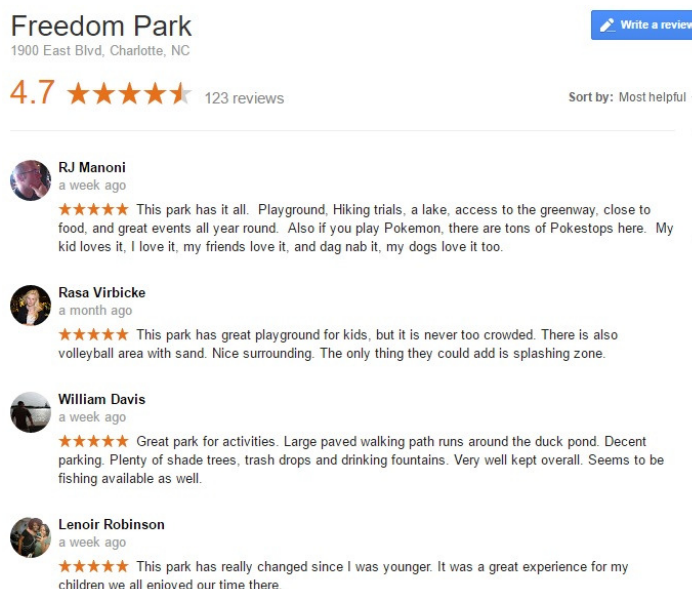


Figure 13: Example of reviews left on Google about Freedom Park in Charlotte.

Using the python language and web-scraping techniques all text reviews (comments) that were available for extraction were scraped in May of 2016 (see APPENDIX J:). Only a subset of reviews is available for extraction. It is not made clear by the providers how that selection process works.

7. Survey of Public Park Visitors

Surveys were collected by graduate and undergraduate students from UNC Charlotte (volunteers) at 18 public parks of different types across Mecklenburg County,

spanning from September to November 2015. At each park type, 35 - 150 visitors were surveyed, totaling 496 surveys (see Table 1):

Table 1: Number of surveys collected per park type

park type	Number of visitors surveyed		
	residents	non-residents	total
Neighborhood parks	29	10	39
Community parks	60	7	67
Regional parks	131	17	148
Nature preserves	92	42	134
Urban park	82	26	108

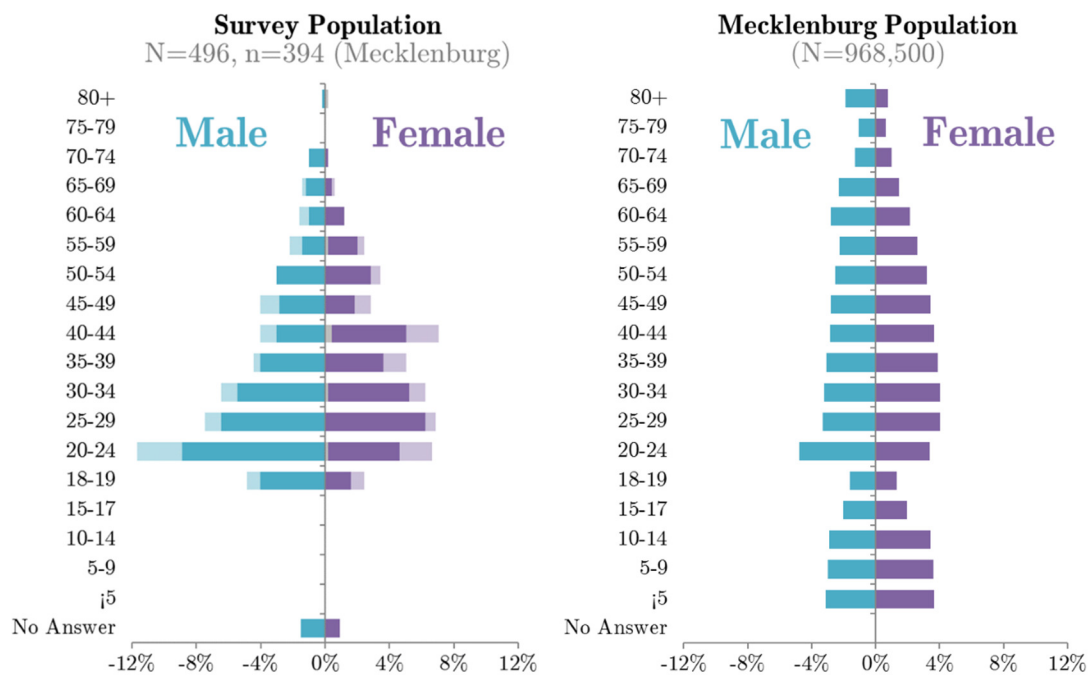


Figure 14: Population pyramid of surveys versus Mecklenburg County.

Figure 14 shows a comparison between the survey population (left) and the population of Mecklenburg County (right) broken down by age and by gender. Some park visitors that were surveyed indicated that they were not residents of Mecklenburg County (n=102). Respondents that were non-residents are shown in a lighter shade of

blue (males) and lighter shade of purple (females) in Figure 14 (on page 43) on the left.

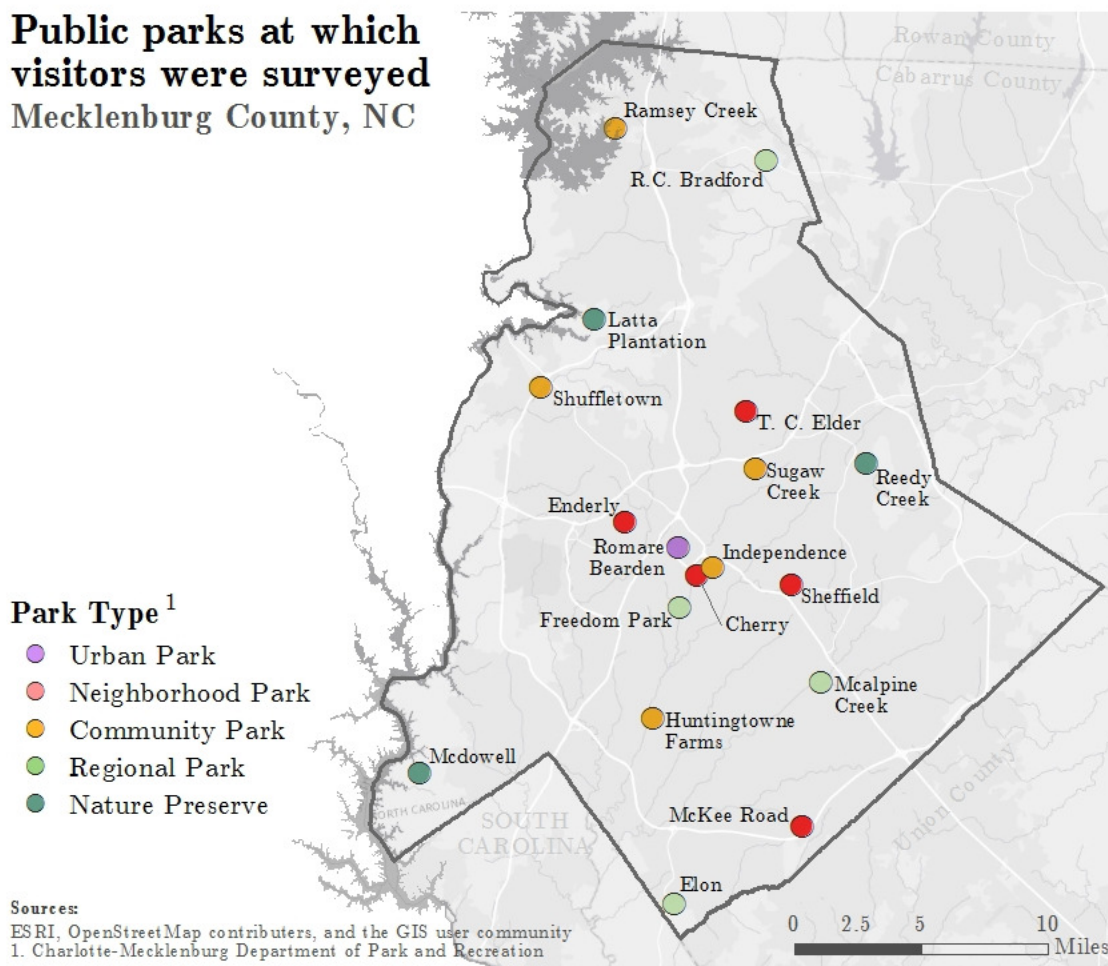


Figure 15: Park selection in Mecklenburg County, where surveys would be conducted in the month of September.

Parks were selected following a stratified sample based on their type, size and the availability of on-site amenities. Finally, parks were further stratified based on the geographic locations, guaranteeing a representative spatial coverage (North, East, South and West). Figure 15 (on page 44) shows the location and names of the parks where visitors were surveyed.

For each park, at least two days were scheduled to survey visitors; one during a

weekday and one during the weekend. The protocol for the data collection process was approved by the Institutional Review Board (IRB) for Research with Human Subjects of UNC Charlotte in May of 2015 (protocol number: 15-05-07, see APPENDIX A:1). The survey instrument was first tested (pilot study) on a select number of individuals at a couple of parks. Insights gained from this pilot study were used to revise the questionnaire and/or revise the list of park locations selected. Amendments were approved by the IRB of UNC Charlotte (see APPENDIX A:2), and shared with the Deputy Director of the DPR of Mecklenburg County. A team of 13 volunteers were recruited (see APPENDIX A:3) and trained before conducting the surveys. They were informed on the study objectives and were trained on recruiting techniques. All surveyors were required to take the IRB protocol training; therefore, they were informed on ethical and privacy requirements of this study. These requirements were repeated during the training. These were the inclusion criteria for participants of this study:

- (i) Individuals must be at the surveyed park location
- (ii) Individuals must be 18 years of age or older
- (iii) Individuals must speak English

Once a visitor agreed to participate, the surveyor offered to read each question out loud for the participant and record his/her answers or to have the participant fill-out the questionnaire alone. For park visitors that were unable to take the survey at the time, surveyors shared a flyer (see APPENDIX C:) that contained a QR-code and a webpage address linking to the online version of the survey. Participants could then fill-out the survey at a later time using a computer or a hand-held mobile device. Surveyors also left these flyers on windshields of cars parked at designated parking spaces. As a reward and

appreciation for their time, participants were offered a free snack from KIND (healthy snack company).

All survey responses (hard-copies and online surveys) were merged into a password-protected database. The park location where a participant has taken the survey as well as the closest intersection to her/his residence (if provided) was geocoded using a GIS. Participants were allowed to skip any questions they did not feel comfortable answering. Questions on the second page of the survey instrument (see APPENDIX B:) included more sensitive questions such as the annual household income, body weight, zip-code and street name of the participant. The survey data was cleaned before used for analysis and the cleaning process is written in scripts for the Stata programming language, which can be found in APPENDIX F:.

Figure 16 (on page 47) shows the distribution of the participants in this study. Based on how detailed their address was shared (at the intersection level, at the main street level, at the zipcode level, etc), a level of accuracy was identified for each geocode. In this figure we can see that most respondents shared address information that lead to high or good accuracy (dark and light green dots). The orange and red dots refer to addresses that did not provide sufficient detail and thus resulted in low or poor accuracy (e.g. only providing zipcode or only sharing that they do live in Mecklenburg County).

Place of residence of surveyed park visitors Mecklenburg County, NC

Accuracy of residence location conversion accuracy from addresses

- High accuracy
- Good accuracy
- Low accuracy
- Poor accuracy

Population Density² people per acre per neighborhood³

- 0 - 1
- 2 - 4
- 5 - 8
- 9 - 16
- 17 - 32
- no population

Sources:

- ESRI, OpenStreetMap contributors, and the GIS user community
 1. OpenStreetMap; Charlotte-Mecklenburg Geospatial Information Services
 2. Quality of Life Study by Metropolis Studies Group at UNC Charlotte
 3. Neighborhood Profile Areas; represent one or more Census block groups

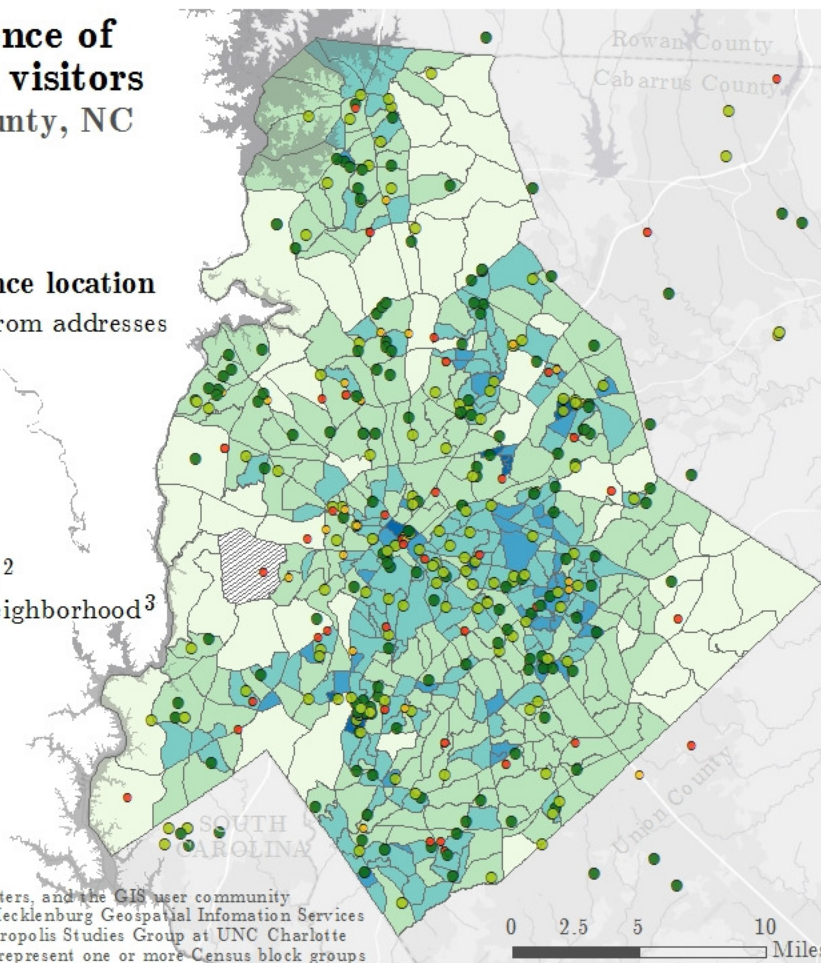


Figure 16: Residence of surveyed visitors in and around Mecklenburg County

8. Neighborhood and Block Group Level Characteristics

Block group boundaries and population counts were downloaded from the census bureau for the year 2010. Additionally, Neighborhood Profile Areas from the Charlotte-Mecklenburg Quality of Life Explore (mcmmap.org/qol) were downloaded as well as the following neighborhood characteristics:

1. Percent beneficiaries of Medicaid and statewide insurance
2. Population density
3. Street connectivity

4. Percent black
5. Park proximity
6. Household income
7. Percent streams adopted by residents
8. Participation in art and cultural activities.

9. Criminal Activity Nearby Parks

Criminal activity in Mecklenburg County is reported by the Charlotte-Mecklenburg Police Department. The location of each reported crime is provided as well. Using the python language, 2014-2015 crime data were extracted from the Charlotte spotcrime webpage (www.spotcrime.com/nc/charlotte), which keeps track of crime reports in Mecklenburg County.

10. Data Limitations

First, to generate the most complete and updated inventory of public parks and recreational facilities, several data sources were merged together, including data from the Department of Park and Recreation and from the county. The cleaning and merging process of these data sources is written in a script for the Python programming language. Certain parks showed conflicting information in different data sources. As an example, certain parks were labeled as developed in one source, labeled as undeveloped in another source and completely missing in another source of data. Because of these inconsistencies, the cleaning and merging process does not guarantee an inventory that is complete and accurate. Second, the inventory generated for non-public parks is a novel dataset and relies on data from the county and OpenStreetMap. The process of selecting impervious surfaces and OpenStreetMap polygons that are associated with recreational

activities has been developed by inspection of the data by myself only. The final dataset generated has not been approved by any local administration, therefore it is not guaranteed to be complete and accurate. Third, online reviews that were extracted from Google and from Foursquare do not represent all reviews posted by users. The selection of reviews that is made available through web scraping might not be a representative sample of all reviews posted on their respective platforms. Fourth, the surveys that were conducted at 18 public parks in Mecklenburg County were conducted in the months of September through November of 2015, which may not be representative of the behavior of individuals at other times of the year. Fifth, participants of the survey were allowed to skip questions they did not feel comfortable answering. Therefore, this data has missing responses to questions. Although many participants shared their zip-code and nearest intersection to their residence, other participants only shared their zip-code. These missing entries in the data are an important limitation for the analyses used in this study. Sixth, the inclusion criteria of participants in our survey limits the generalizability of our results. Individuals under 18 years of age were not included in our study, therefore their opinion is not being reflected in our results. Additionally, resident of Mecklenburg County who never visit parks could not be represented in our sample. However, their input on park accessibility is an important aspect, especially from a social justice perspective. Seventh, the sample size that was collected at neighborhood parks is small and therefore lacks explanatory power to generalize experiences for that park type. Neighborhood parks however are an important aspect of this study since their primary purpose is to provide close access to parks.

METHODS

To evaluate spatial access to public parks in Mecklenburg County, two variations of the floating catchment area (FCA) method are applied using the public park inventory as input data (see data description on page 41). Then, descriptive and statistical analyses are applied on surveys data gathered from public park visitors in the Fall of 2015 in Mecklenburg County. Analysis with the FCA method give insights on potential access to parks, whereas analysis of surveys provides a better understanding of perceived access to parks in Mecklenburg County. Finally, sentiment analysis applied to data from reviews left by park users on online commenting platforms show new possibilities to monitor park visitor satisfaction in real-time. Each method used in this study is described and explained in detail in this section.

1. Modeling Spatial Access (Potential): the Floating Catchment Area (FCA) Method

Floating catchment area (FCA) methods represent a category of spatial access methods that were initially conceived in a healthcare context. A catchment area is estimated around a service facility based on some maximum distance or time individuals are willing to travel; any individual within that catchment area is deemed to have access to that facility, while all others do not.

1.1. 2SFCA, Two-Step Floating Catchment Area Method

The two-step floating catchment area (2SFCA) method is a more advanced FCA method because it evaluates access taking into account supply and demand within each catchment. It was originally developed by Luo and Wang (2003) in the context of access to general practitioners. Accounting for the size of the demand (e.g. population within a distance of a practice) and the size of the supply (e.g. the number of physicians working

at a practice) allows to assess practices where the demand surpasses the supply and vice-versa. This model has been used in many other contexts since its onset, including access to parks. The 2SFCA method by Luo and Wang (2003) can be described in two steps.

1.1.1 First Step: Calculate the Park-to-Population Ratio

For each park location j , identify all the population locations (k) that are within a threshold travel time or distance, d_0 (or the travel cost) from location j (that is, catchment area j), and compute the park-to-population ratio, R_j , within the catchment area:

$$R_j = \frac{S_j}{\sum_{k \in \{d_{kj} \leq d_0\}} P_k}, \quad (1)$$

where P_k is the population of geographic unit k whose centroid falls within the catchment (that is, $d_{kj} \leq d_0$), S_j is the size of the park at location j , and d_{kj} is the travel cost between k and j . Step one of the 2SFCA method is illustrated in Figure 17.

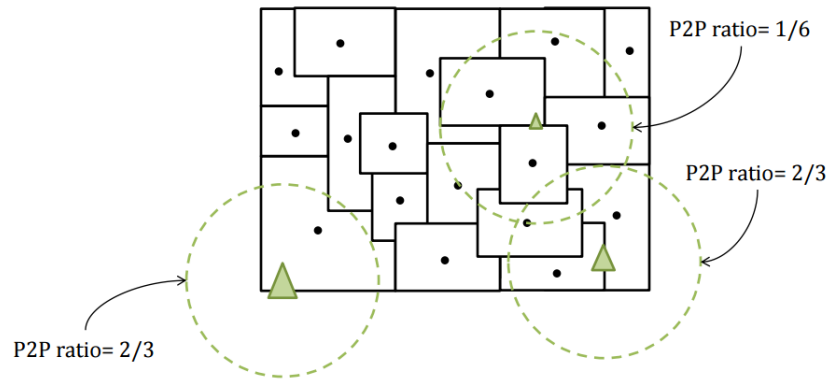


Figure 17: Illustration of the first step in the Two-step Floating Catchment Area (2SFCA) method, which calculates each park-to-population ratio

1.1.2 Second Step: Calculate Spatial Accessibility

For each population location i , identify all park locations (j) within the threshold travel cost (d_0) from location i (that is, the catchment area i), and sum the park-to-

population ratios, R_j , at these locations:

$$A_i^F = \sum_{j \in \{d_{ij} \leq d_0\}} R_j = \sum_{j \in \{d_{ij} \leq d_0\}} \frac{S_j}{\sum_{k \in \{d_{kj} \leq d_0\}} P_k}, \quad (2)$$

where A_i^F represents the accessibility at the resident location i based on the 2SFCA method, R_j is the park-to-population ratio at the park location j whose centroid falls within the catchment centered at i (that is, $d_{kj} \leq d_0$), and d_{ij} is the travel time between i and j . Step one of the 2SFCA method is illustrated in Figure 18.

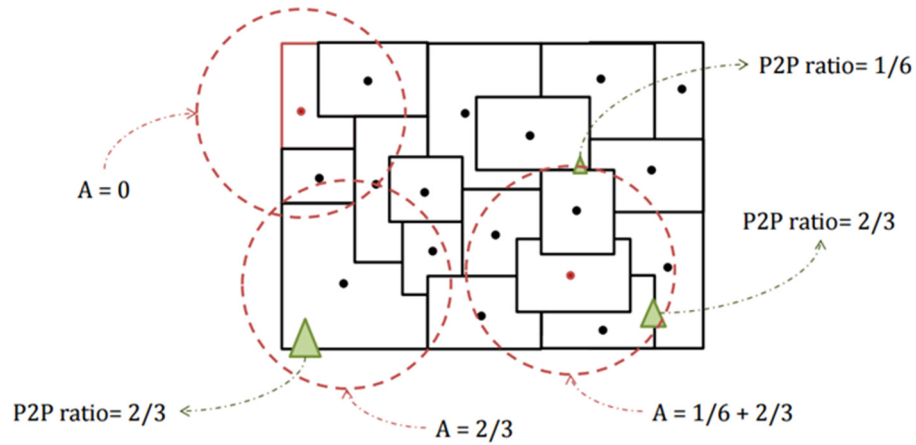


Figure 18: Illustration of the second step in the Two-step Floating Catchment Area (2SFCA) method, which calculates access to parks for each geographic unit

1.1.3 Limitations

One of the major limitations of the 2SFCA method is that each catchment is set at a fixed distance, regardless of the type of facility, which does not adhere to the way that some commodities are planned. In this method, catchment areas around neighborhood parks for example have the same size than those around regional parks, even though the latter are intended to attract individuals from across the region while neighborhood parks mainly attract nearby residents. Second, Luo and Wang (2003) only consider the car as a

way of transport. Many studies evaluating access to parks limit their analysis to this mode of transportation. Therefore, the evaluation only accounts for individuals that own or have access to a car. In this study, I develop a variation of the 2SFCA, called the Variable Floating Catchment Area (VFCA) method to address both limitations.

1.2. Variable Floating Catchment Area (VFCA) Method

The Variable Floating Catchment Area (VFCA) method is a variation of the 2SFCA I develop for this study to measure spatial accessibility to parks in Mecklenburg County. Specifically, I re-conceptualize the 2SFCA (Luo & Wang, 2003) to allow park catchments to vary their size depending on the park size and the number of available amenities. The VFCA method is not intended to replace other methods such as the two- or three-step floating catchment area methods, however this approach provides an alternative model to capture facility attraction. Moreover, since all parameters are designed to be flexible, this approach has the capability to support scenario analysis, which can be extremely useful for planners (Xiang & Clarke, 2003). The VFCA method can be described in three steps.

1.2.1 First Step: Define Catchment Sizes

In Dony et al. (2015), I suggest that the attractiveness of a park be a function of its size and the number of available amenities. The attraction coefficient, S_j , of the supply at node j is estimated using a weighted sum approach (Equation 3), where the weights γ_A and γ_K reflect the importance of park acreage (S_j^A), and on-site amenities (S_j^K), respectively and where $\gamma_A + \gamma_K$ equals 1.

$$S_j = \left[\gamma_A \frac{S_j^A}{\max_{j \in J} S_j^A} \right] + \left[\gamma_K \frac{S_j^K}{\max_{j \in J} S_j^K} \right] \quad \forall j \in J \quad (3)$$

Note that S_j^A and S_j^K can be changed to any other set of characteristics when spatial access to commodity other than parks is being assessed. The normalized attraction coefficient, \bar{S}_j , redistributes all the attraction values between 0 and 1 (Equation 4):

$$\bar{S}_j = \frac{S_j - \min_{j \in J} S_j}{\max_{j \in J} S_j - \min_{j \in J} S_j} \quad \forall j \in J \quad (4)$$

Based on this attraction value, each park is assigned a catchment area of a certain size depending on the travel mode. The higher the attraction value of a park (e.g. large park with multiple amenities), the larger the extent of its catchment. The catchment area of each park, T_{jm}^{crit} , is defined using the normalized attraction coefficient, \bar{S}_j , and is dependent on the travel mode m that is used.

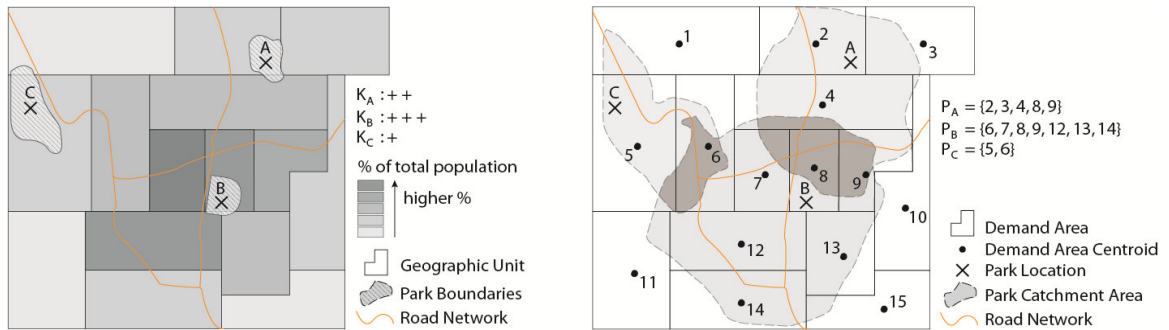


Figure 19: Illustration of the concept of the variable-width floating catchment area (VFCA) method

This definition of catchments in the VFCA method is illustrated in Figure 19. Three parks (A, B and C) of varying sizes and number of amenities ($|K_A|=2$, $|K_B|=3$, $|K_C|=1$) are distributed across a hypothetical study region (Figure 19, left). After computing the attraction coefficient of each park (see equations 3 and 4), the extents of their catchment are calculated (Figure 19, right). In this study, I compare spatial accessibility of each demand point between four modes of transport: (1) driving, (2)

public transit, (3) bicycling and (4) walking. The travel mode coefficient, ζ_m (where $\zeta_{car}=1$; $\zeta_{transit} = 4/3$; $\zeta_{cycling} = 6/5$ and $\zeta_{walking} = 8/7$), translates each normalized attraction coefficient into a catchment area expressed in minutes (Equation 3).

$$T_{jm}^{crit} = \begin{cases} \zeta_m * 5 \text{ min}, & \bar{S}_j < 0.1 \\ \zeta_m * 15 \text{ min}, & 0.1 \leq \bar{S}_j < 0.3 \\ \zeta_m * 30 \text{ min}, & 0.3 \leq \bar{S}_j < 0.7 \\ \zeta_m * 45 \text{ min}, & 0.7 \leq \bar{S}_j < 0.9 \\ \zeta_m * \max(t_{ijm}), & 0.9 \leq \bar{S}_j \end{cases} \quad (5)$$

I consulted with staff from the Mecklenburg County Department of Park and Recreation in order to determine appropriate travel mode coefficients, ζ_m and maximum travel budgets (in minutes) per transportation mode. In this respect, I am adhering to a normative accessibility assessment (Páez et al., 2012).

1.2.2 Second Step: Calculate Park-to-Population Ratio

Crowding at a park can decrease our willingness to travel to a park, and as a consequence, it can reduce its attractiveness. This latter consideration is in the same spirit as Lee and Hong's (2013) ASD metric. The park's acreage is divided by the total population within its catchment and gives a sense of potential crowding (park-to-population ratio). A low park-to-population ratio (e.g. small park surrounded by large population) indicates there is a higher likelihood for crowding. Based on the catchment area of each public facility, T_{jm}^{crit} , the set of geographic units that fall within this catchment is selected ($N_{im} = \{j: t_{ijm} < T_{im}^{crit}\}$). The total demand, V_j for each public facility j is computed by summing up the respective populations, g_i in this set (N_{im}) and using a distance decay coefficient, β , to give a higher importance to populations living in geographic units located closer to public facility j (Equation 6).

$$V_j = \sum_{i \in N_{jm}} \frac{g_i}{t_{ijm}^\beta} \quad \forall j \in J \quad (6)$$

The distance decay coefficient, β , governs the shape of the decay function; when β is high (> 1), the demand for parks will decrease faster with increasing distance. For instance, a higher distance decay coefficient might be used for the elderly, as they would be expected to travel shorter distances to visit parks (see Schwanen & Páez, 2010).

The park-to-population ratio, R_j (crowding index) of park j is then computed by dividing the acreage by the total population within the catchment area, V_j (Equation 7).

$$R_j = \frac{\bar{s}_j}{\sum_j V_j} \quad \forall j \in J \quad (7)$$

When studying commodities other than public parks, this ratio can be re-defined using alternative parameters.

1.2.3 Third Step: Calculate Spatial Accessibility

When estimating the spatial accessibility, the park-to-population ratios, R_j , are summed and weighted by the distance that separates the geographic unit from the park. A distance decay function gives a higher weight to the park-to-population ratio of a park when they are located closer to the geographic unit's centroid. The spatial accessibility at geographic unit i is then defined as:

$$A_i = \sum_{j \in N_{im}} \frac{R_j}{t_{ijm}^\beta} \quad \forall i \in I \quad (8)$$

The spatial accessibility score of each block group is estimated four times; once for each mode of transport. The higher the accessibility score, the greater the accessibility of that geographic unit to parks, compared to all other geographic units in the study area. The VFCA method was written in a script for the programming language (see APPENDIX K:)

1.3. Scenarios

One of the advantages of the VFCA method is its ability to generate results under different scenarios by letting the parameter values vary. For example, giving more weight to the size of the park than to the number of available amenities might result in different accessibility patterns. Moreover, the current set of parks and associated characteristics can be changed (e.g. adding a new park location or including more amenity types), allowing planners and decision makers to compare accessibility under various scenarios. To illustrate this capability and to gain insights on the sensitivity and the spatial structure of the model, different scenarios of the VFCA are run. Each scenario will be run four times, once for each mode of transport: car, public transit, bicycling, and walking. The travel mode coefficients, ζ_m , can be adapted as well. A Department of Park and Recreation usually has particular standards they need to reach (normative standards). As an example, a county might strive to give everyone access to a neighborhood park within a 10-minute walk and to a regional park within a 20-minute drive. Finally, the three different scenarios run for this study use the same distance decay coefficient ($\beta = 1.2$).

1.3.1 Scenario I: Two-Step Floating Catchment Area (2SFCA)

In the first scenario, I evaluate accessibility using parameters that approximate the original 2SFCA method by Luo and Wang (2003). The catchment of each park is fixed and set using a 15-minute catchment area. Thus, all catchments are uniform regardless of each park's size and number of amenities.

1.3.2 Scenario II: VFCA, Attraction Based on Park Size Only ($\gamma_A = \mathbf{1}$; $\gamma_K = \mathbf{0}$)

In the second scenario, park size is the only factor used to compute the attraction of a park and its catchment area. Thus, compared to Scenario I (2SFCA), I now explicitly

incorporate the concept of variable-width catchments. Consequently, comparing scenarios I and II will show the effect of using variable widths on accessibility results.

1.3.3 Scenario III: VFCA, Equal Weighting ($\gamma_A = \gamma_K$)

In the third scenario we look at accessibility outcomes when park size and number of on-site amenities are equally weighted. Comparing scenarios II and III will illustrate the effect on accessibility results when using an additional variable (namely, park amenities) to estimate park attraction.

2. Analyzing Access (Revealed) to Parks

To respond to research questions 3 through 5, seven survey questions will be used; they are listed in Table 2 (on page 59). The access scores gathered from the first question and the four importance scores gathered from questions 2 through 5 (see Table 2), are all categorical variables of the ordinal type. The responses to those questions, together with demographic data reported by the respondents are used to analyze perceived access to parks in Mecklenburg County (research question 3). Tests on skewness and normality, show that these five variables are not normally distributed and are skewed, even after transformation of the data (see APPENDIX G:). Therefore, it is not advised to use any statistical methods that assume linearity or normality of the dependent variable (e.g. analysis of variance or linear regression models). Instead, ordinal logistic regression is used to answer research question 3.

Table 2: Questions from the survey that will be used to respond to research questions 3 through 5

Survey Question	possible answers	data type
1- "How would you rate your access to parks?"	scale from 1 to 5 (1 = very poor, 5 = very good access)	ordinal
2- "When you go to any park, how important are park amenities?"	scale from 1 to 5 (1 = not important, 5 = very important)	ordinal
3- "When you go to any park, how important is the park design and aesthetic?"	scale from 1 to 5 (1 = not important, 5 = very important)	ordinal
4- "When you go to any park, how important is the size of the park?"	scale from 1 to 5 (1 = not important, 5 = very important)	ordinal
5- "When you go to any park, how important is the safety of the park?"	scale from 1 to 5 (1 = not important, 5 = very important)	ordinal
6- "How much time did it take to travel to this park?"	minutes	positive, continuous
7- "Did you come to this park to engage in physical activity today?"	yes / no	dichotomous

The travel times gathered from the sixth question (see Table 2), is a positive and continuous variable of the interval type. The responses to this question, together with demographic data reported by the respondents are used to analyze willingness to travel to parks in Mecklenburg County (research question 4). Tests on skewness and normality, show that this variable is not normally distributed and is skewed, even after transformation of the data (see APPENDIX G:). Therefore, it is not advised to use any statistical methods that assume linearity or normality of the dependent variable (e.g. analysis of variance or linear regression models). Instead, ordinal logistic regression is used to answer research question 4.

The engagement in physical activity gathered from question 7 (see Table 2) is a categorical variables of the dichotomous type. Therefore, logistic regression is used to answer research question 5. Scripts that cover all descriptive statistics and statistical analyses, which were written for the Stata programming language, can be found in APPENDIX H:.

3. Calculating Park Availability within the Local Context

Using kernel density estimation, the spatial availability of parks in Mecklenburg County was calculated. Kernel density is a non-parametric method that converts a random variable into a continuous probability density surface. Using a cell size of one square mile and based on the location of parks, the continuous surface estimates the number of parks or recreational facilities per square mile for every location in Mecklenburg County. Kernel density estimation provides a way to visualize the distribution of parks in the county and has been used in studies on park accessibility (e.g. Maroko et al., 2009).

The kernel density estimation is used on the inventory of public parks and on the inventory of private parks separately. Additionally, a kernel density surface is estimated when both inventories are taken together. Finally, based on insights from park visitors, weights will be given to each park based on size, amenities and/or safety.

4. Estimating Park Satisfaction of Online Reviews

The opinionfinder algorithm (Wilson, Wiebe, & Hoffmann 2005), which is a form of data mining, will be used to extract sentiment from reviews written by online users on Foursquare and Google about parks. Opinionfinder is a system that was developed by students and faculty at the University of Pittsburg (PA), which – among other uses, identifies whether certain words are classified to be positive or negative. Based on this sentiment dictionary, it is possible to derive the sentiment of a sentence based on the words it constitutes. The sentiment score x_t , of a comment left on online commenting platforms can be analyzed with Equation 9.

$$x_t = \frac{\text{percent}_t(\text{pos.words})}{\text{percent}_t(\text{neg.words})}, \quad (9)$$

Where $percent_t(pos. words)$ is the number of positive words divided by the total number of words constituting the comment and where $percent_t(neg. words)$ represents the number of negative words divided by the total number of words constituting the comment. The average sentiment scores from comments posted about one park will determine the overall public sentiment at that park. If this ratio is above 1, the sentiment is considered positive, if it is below 1 it is considered negative and if it is equal to 1 it is considered neutral.

5. Methodological Limitations

First, accessibility with the VFCA method is estimated using block group centroids as a point of origin for all travel and assumes that subsequent accessibility values are uniform across individuals residing within the boundaries of that block group. In reality of course, some segments of the population face lower levels of mobility, such as seniors (Schwanen & Páez, 2010) or those with disabilities (Casas, 2007). Moreover, it is important to note the limitations of this analysis due to the well-known Modifiable Unit Area Problem (MAUP) as coined by Openshaw (1983). Second, for public transit, we used travel time estimates during a typical workday, and as such did not explore the change in accessibility at different times of day, nor during the weekend. Third, we used nine types of recreational amenities, which is not an exhaustive list, nor does it reflect the quantity or quality of each amenity. However, this model can easily be modified to incorporate more amenities or account for other factors such as the presence of trails. Moreover, it is possible to adjust weights to each of the amenities. For example, unique amenities that attract users from a greater distance may be assigned higher weights. As an example, Grayson Park is an average sized neighborhood park (12 acres) in the southern

part of Mecklenburg County. However, it is the only park in the county with a skate park, making it unique and an incredibly popular destination for skateboarders all over the county and beyond. It is also important to note that not all groups of the population use or seek amenities in a similar manner. For instance, adolescents may prefer active sports (e.g. soccer, baseball, tennis) while the elderly may favor parks with walking trails and more passive recreation. Similarly, preferences may vary based on cultural differences. Fourth, the park-to-population ratio was calculated only taking into account the population in block groups within Mecklenburg County. This does not take into account individuals that live outside of Mecklenburg County, but that do live close enough to the boundary making them potential park users. Therefore, the park-to-population ratio of parks that are close to the boundary of Mecklenburg County may be overestimated due to unaccounted users living outside the county. Fifth, the surveys have limitation, which were detailed in the data limitation (see page 48). Sixth, the sentiment estimated by the opinionfinder system is based on a dictionary of English words that is not exhaustive. Moreover, the Spanish-speaking community is growing in Mecklenburg County and comments left in Spanish cannot be interpreted using this dictionary. Finally, the opinionfinder system uses an algorithm to put each word within the context of the entire sentence. Although their algorithm has high accuracy ratings, it cannot be guaranteed that each comment's sentiment is estimated correctly. Reviews written online often contain spelling mistakes and poor sentence structures, which may affect the accuracy rate of the opinionfinder system.

RESULTS

Results for the three research objectives will be addressed following the same order they are outlined in my dissertation. In the first section, potential access to public parks and recreational facilities in Mecklenburg County is presented for four modes of travel. In the second section, revealed access is presented for visitors surveyed at 18 public parks between September and November of 2015. Taking into account insights from this survey, the third section presents the availability of public and non-public parks and recreational facilities in Mecklenburg County. In the fourth and final section, the sentiment of online reviews towards parks in Mecklenburg County are presented. Many of the results are presented using maps. To sketch a more informative picture of neighborhoods characteristics in Mecklenburg County, each map will show a different socio-economic or civic variable in its background. These variables are all made available by the Charlotte-Mecklenburg Quality of Life Explorer.

1. Spatial Access (Potential) to Public Parks in Mecklenburg County

Using the method described in each of the three scenarios outlined earlier, relative accessibility scores are calculated. The resulting accessibility values are a score between 0 and 1, making it possible to compare patterns for different modes of transportation. Each map is presented using the same classification and color scheme (quintile classification and sequential colors). The dark, red color shades reflect block groups with higher accessibility to public parks using a particular mode of transport. The light, gray shades reflect block groups with low levels of spatial accessibility, compared to all other block groups in the study area.

Figure 20A shows the results of scenario I, which uses the two-step floating

catchment area method (2SFCA). For all modes of transport, but especially by car, census blocks with higher relative accessibility are found predominantly in the western and northern portions of the county. By transit, bicycle or foot, a more spatially dispersed pattern is observed however. Since the only weighing variable in the 2SFCA method is size, the western part, which contains most parks of a larger size result in higher accessibility scores (see Figure 10 and Figure 11 for park locations). It is important to note that the pattern for each respective mode of transportation largely follows the underlying transport infrastructure.

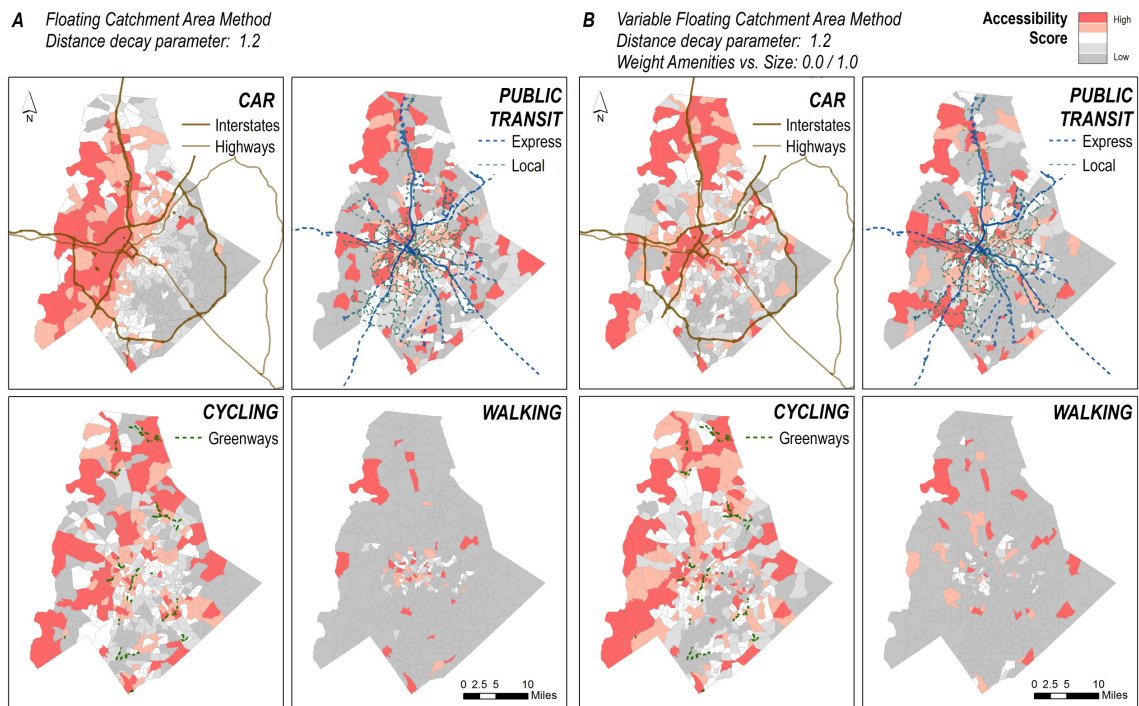


Figure 20: Spatial accessibility for scenarios I and II, for four different modes of transportation.

Figure 20B shows the results of scenario II, which uses the Variable Floating Catchment Area (VFCA) method based on park size only. Comparing the results of both scenarios make the differences in outcome clear. Results of accessibility scores by car in

Figure 20B show a more dispersed pattern; block groups that are well connected to several public parks or parks of larger sizes result in higher accessibility scores. We also note that the higher density of high-speed roads in the western portion of the county helps give rise to the accessibility patterns observed in Figure 20A. Results by public transit (Figure 20B) show higher scores for block groups connected to larger parks through transit express routes.

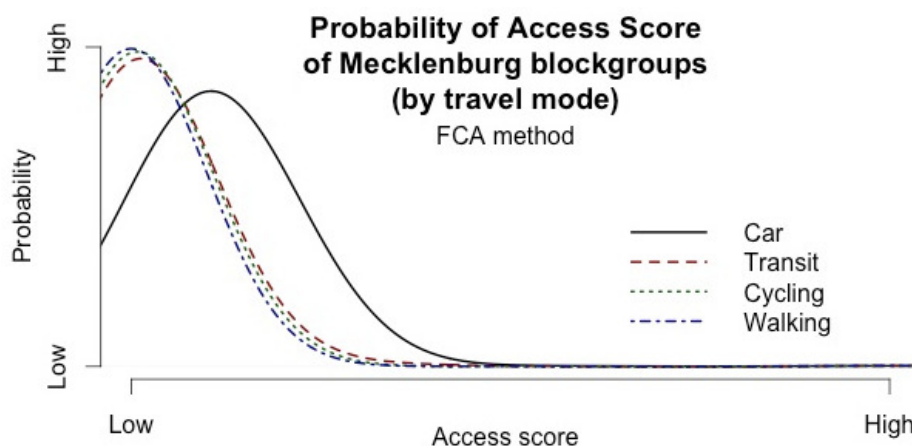


Figure 21: Distribution of spatial accessibility scores for scenario I for four different modes of transportation

The distribution of accessibility scores among block groups in Mecklenburg County are illustrated in Figure 21 under scenario I (2SFCA). This graph suggests that most block groups experience very low accessibility scores, regardless of the transport mode. When traveling by car, the probability that a block group has a higher accessibility score is slightly greater compared to the three remaining modes of transport.

Variable Floating Catchment Area Method
 Distance decay parameter: 1.2
 Weight Amenities vs. Size: 0.5 / 0.5

Accessibility Score


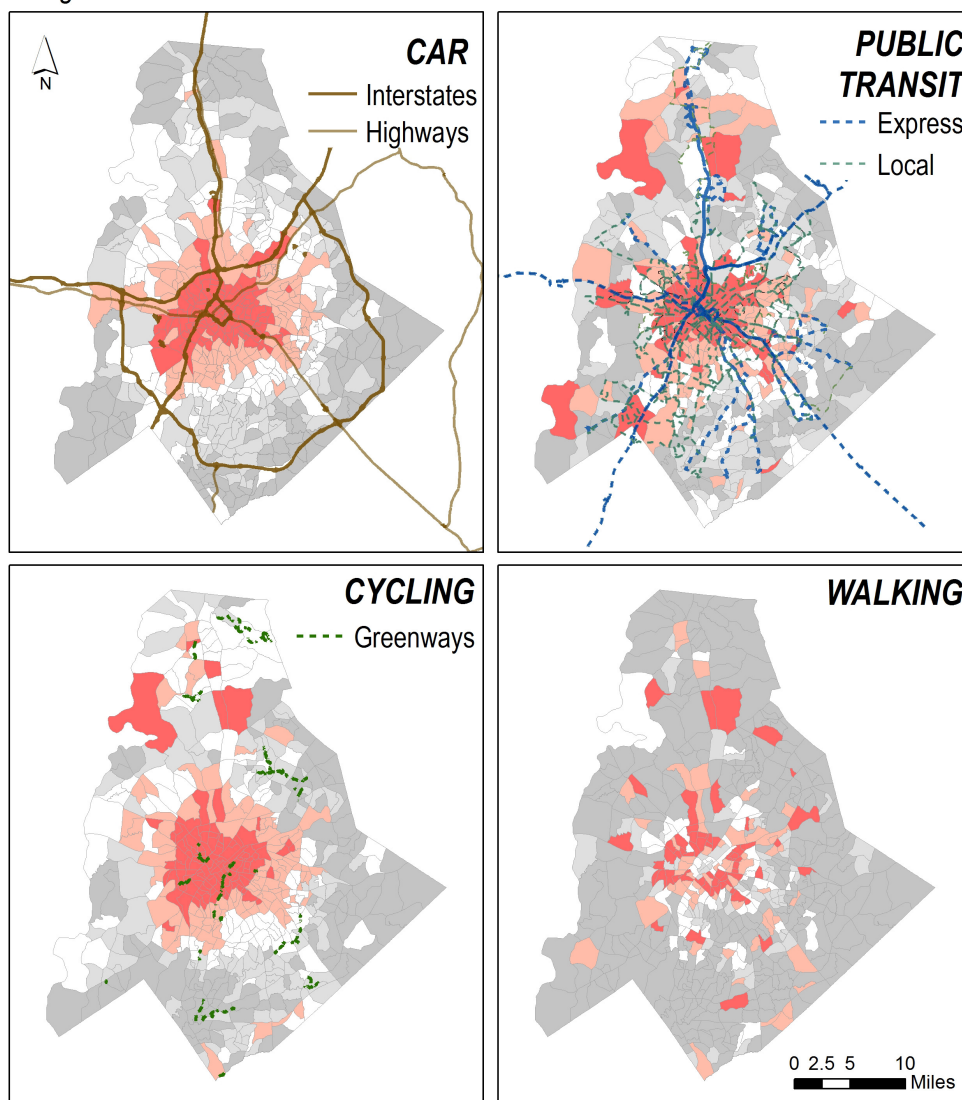



Figure 22: Spatial accessibility for scenario III, for four different modes of transportation

Figure 22 shows the results of scenario III, which uses the Variable Floating Catchment Area (VFCA) method based on park size and number of amenities. Accessibility patterns for all modes of transport seem to exhibit higher scores nearby Charlotte business district (centrally located in Mecklenburg County). Since a greater number of amenities is found in public parks located closer to the city center, the pattern

shows higher accessibility values for block groups located near or in-between parks with multiple amenities. When traveling by car, the city center and neighborhoods located along interstates have a much higher level of access. When traveling by public transit, high levels of accessibility are observed in the city center and along bus route corridors. A few block groups located at the periphery of the county consistently experience high levels of accessibility. This is partly due to the close proximity of the block group's centroid either to a park's entrance, a well-connected road or a bus stop. Accessibility levels for pedestrians appear relatively low and patchy in all scenarios.

The accessibility patterns in Figure 20 (scenarios I and II) drastically contrast with those in Figure 22 (scenario III), and this holds true for all modes of transportation. When comparing the results for car travel, we observe a near mirror image between Figure 20A and Figure 22. As larger parks tend to be located in the northern and western edge of Mecklenburg County (see Figure 10 and Figure 11), block groups in these areas experience higher levels of accessibility when more weight is given to park size.

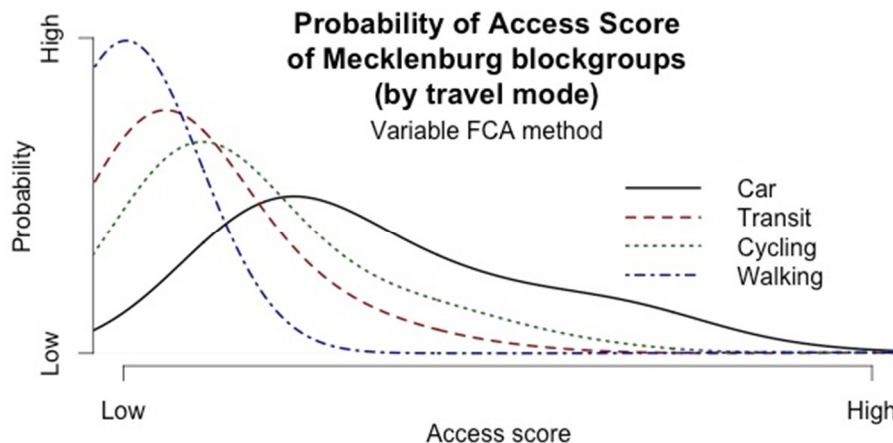


Figure 23: Distribution of spatial accessibility scores for scenario III for four different modes of transportation

The distribution of accessibility scores among block groups in Mecklenburg County, under scenario III is represented in Figure 23. The VFCA model is deemed to create outcomes that better reflect the barriers faced by residents when accessing public parks. Compared to the graph in Figure 5, accessibility scores show a broader distribution when using the VFCA method, particularly when traveling by car.

2. Access (Revealed) to Parks in Mecklenburg County

2.1. Perceived Access to Parks

Visitors were asked how they would rate their access to parks (between 1 and 5; 1 being very poor access and 5 very good access). They were also asked to rate the importance of four park characteristics (between 1 and 5; 1 being not important and 5 being very important); namely (1) the size of the park, (2) the amenities available at a park, (3) the aesthetic and design of the park and (4) the safety of the park.

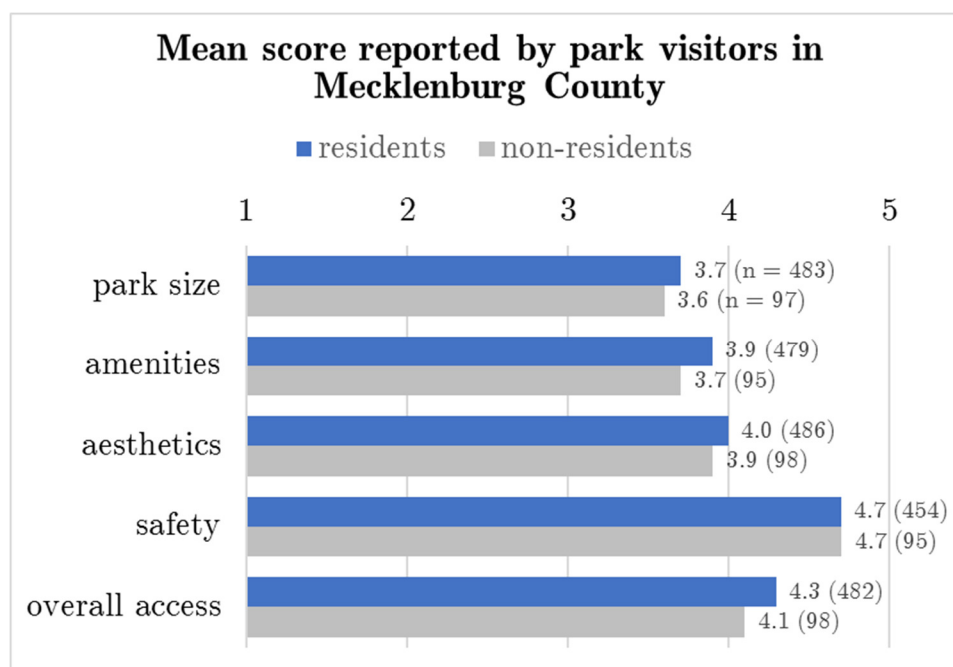


Figure 24: Score reported by park visitors on their overall parks score and the importance of four park characteristics.

In Figure 24, we can see that on average, residents of Mecklenburg County, experience a very good access to parks (4.3 out of 5). The importance of safety at parks is rated to be very important among all visitors (4.7 out of 5). The size and available amenities at parks were rated to be important (scores below 4 out of 5). Interestingly, most studies rely on the size of parks to weigh the access to a park.

```
Iteration 0:  log likelihood = -366.77821
Iteration 1:  log likelihood = -358.70268
Iteration 2:  log likelihood = -358.6635
Iteration 3:  log likelihood = -358.66349
```

```
Ordered logistic regression
Log likelihood = -358.66349
```

```
Number of obs   =      328
LR chi2(4)      =      16.23
Prob > chi2     =      0.0027
Pseudo R2      =      0.0221
```

access_score	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
cp_walk_time_simple						
10 minutes or less	.7146792	.243806	2.93	0.003	.2368281	1.19253
bmi	.0429789	.0213841	2.01	0.044	.0010669	.084891
gender						
Male	.3423605	.2154526	1.59	0.112	-.0799187	.7646398
car_availability_binary						
Yes	.1981949	.5588685	0.35	0.723	-.8971673	1.293557
/cut1	-3.054932	.9631573			-4.942686	-1.167178
/cut2	-1.471996	.8180787			-3.075401	.1314087
/cut3	.0450929	.7902174			-1.503705	1.593891
/cut4	1.652701	.7950732			.0943863	3.211016

Figure 25: Results of the ordinal logistic regression for research question 3

Homeownership, income, education level and ethnicity or race as reported by visitors were not associated with their perceived access to parks within a 90% confidence level. When testing for the impact of gender, car availability, bmi and walk time to the closest park however, the Likelihood Ratio Chi-Square showed that at least one of the

predictors' regression coefficient was not equal to zero in the model (95% confidence level, see Figure 25). This model was run on residents of Mecklenburg County; non-residents were omitted from the dataset in order to provide results that speak to perceived access by local residents.

At an 99% confidence level, the order logit (log-odd) for visitors who live a 10-minute walk or less from a park to perceive their access to be higher is 0.71 more than visitors who live more than a 10-minute walk from a park when the other variables in the model are held constant. At an 95% confidence level, a one-unit increase in a visitor's BMI would result in a 0.04-unit increase in the ordered log-odds of perceiving their park access higher while other variables are being held constant. At an 85% confidence level, the order logit for males perceiving their access to be higher is 0.34 more than females when the other variables in the model are held constant. Finally, the order logit for visitors who have access or own a car to perceive their access to be higher is 0.19 more than visitors who do not have access to a car when the other variables in the model are held constant. However, the latter logit is not significant at the 90% confidence level. When I run the same model based on all park visitors (residents and non-residents), we find that all factors that were significant, stay significant and the multinomial logits are in the same direction and have a similar size.

In Table 3 (on page 71), the median access score, age, BMI and time a visitor traveled to a park were calculated for different walking times to the closest park. This table suggests that visitors who walk 10 minutes or less to their closest park have a median access score of 5 (out of 5) compared to a median score of 4 (out of 5) for a walk time over 10 minutes. It is also worth mentioning that visitors who do have a lower

walking time to their closest park, had a lower median travel time to the park they were visiting when surveyed.

Table 3: Descriptive statistics on the walk time to the closest park

Walk time to closest park		access score	age	BMI	travel time
10 min. or less	p50	5	33.5	26	10
	N	108	108	101	110
11 to 30 min.	p50	4	33	25	15
	N	154	150	143	155
more than 30 min.	p50	4	34	25	15
	N	104	101	94	103

2.2. Willingness to Travel

Visitors were asked how much time it took them to travel to the park they were surveyed at that day. For all visitors, the median time traveled to a public park was 15 minutes. When accounting for Mecklenburg County residents only however, the median time traveled was 10 minutes.

Table 4 (on page 72) reports the time traveled to the park where visitors where surveyed is broken down for five individual's characteristics. We can see that the for these five characteristics, the median reported time traveled is between 10 and 15 minutes. The 95th percentile is also presented in this figure and suggests more variance in the willingness to travel for each individual's characteristic. In particular, age, income and car availability exhibit considerable variance (Figure 26 on page 72). The largest variance is clearly visible when breaking down reported travel times by car availability (far right). Visitors who report owning a car, have median reported travel time of 11.5 minutes and goes up to 30 minutes at the 95th percentile. Visitors reporting they do not

have a car (n=19) have a 90-minute travel time at the 95th percentile.

Table 4: Summary of reported travel times

Mecklenburg Residents Reported travel time (in minutes)			
Age	median	p95	N
18-24	10	35	95
25-29	10	30	63
30-44	15	30	130
45-59	10	30	67
60+	10	25	26
Income			
0-25K	14	60	72
25-49K	10	30	73
50-74K	15	30	64
75-99K	10	30	49
100-149K	10	30	37
150K+	12	45	31
Education			
No high school	15	30	18
High school	10	30	133
Higher-education	10	30	232
Car availability			
No car	10	90	20
Car access	10	75	36
Own car	11.5	30	332
Children at home			
1-2	13	30	93
3 or more	10	35	38
None	10	30	49

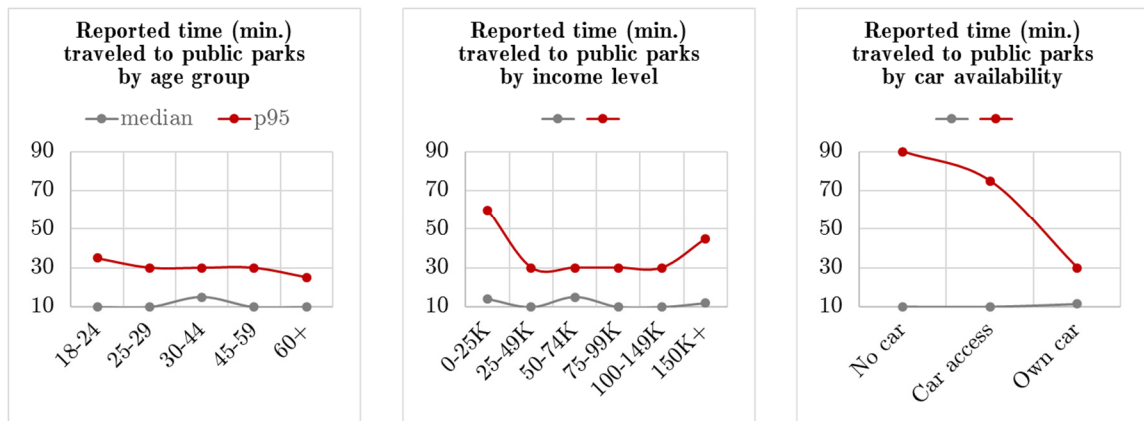


Figure 26: Median and 95th percentile travel times

It is important to mention that from these 19 respondents, 63% are ages 18-24, 26% are ages 25-44 and 11% are 45 or older and that a Pearson Chi Square test rejects the hypothesis that this distribution is random at a 99% confidence level. Looking at the break-down by age (far left), the youngest age group has the same median reported travel time than the oldest age group. However, the 95th percentile is 10 minutes higher for the youngest age group compared to the oldest, which has the lowest travel time at the 95th percentile.

Finally, the break-down by income (middle) shows that visitors in the lowest (21% of respondents) and highest (10% of respondents) income levels have similar median reported travel times, 14 and 12 minutes respectively. Visitors in the lowest income group however, has a 95th percentile travel time that is 15 minutes higher than that of the highest income group. Again, it is important to mention that a Pearson Chi square test on age and income rejects the null hypothesis that the distribution would be random, at a 99% confidence level.

Table 5: Summeray of reported travel time per park type

Mecklenburg Residents Reported travel time (in minutes)			
Park type	median	p95	N
Neighborhood	10	30	28
Community	10	30	60
Regional	15	30	131
Nature Preserve	15	32.5	92
Urban	17.5	45	80

It is likely that different park types attract residents for different reasons. In Table 5, the reported travel time is presented for different park types. Neighborhood and

Community parks have the same median and 95th percentile. Regional parks and nature preserves however, have a 5-minute higher median. The reported travel time at the 95th percentile is highest for urban parks, for which the median is highest as well. To visualize this information differently, Figure 27 shows the location of each public park at which surveys were collected. The color of each dot represents the type of the park, whereas its size represents the median travel time reported by surveyed visitors.

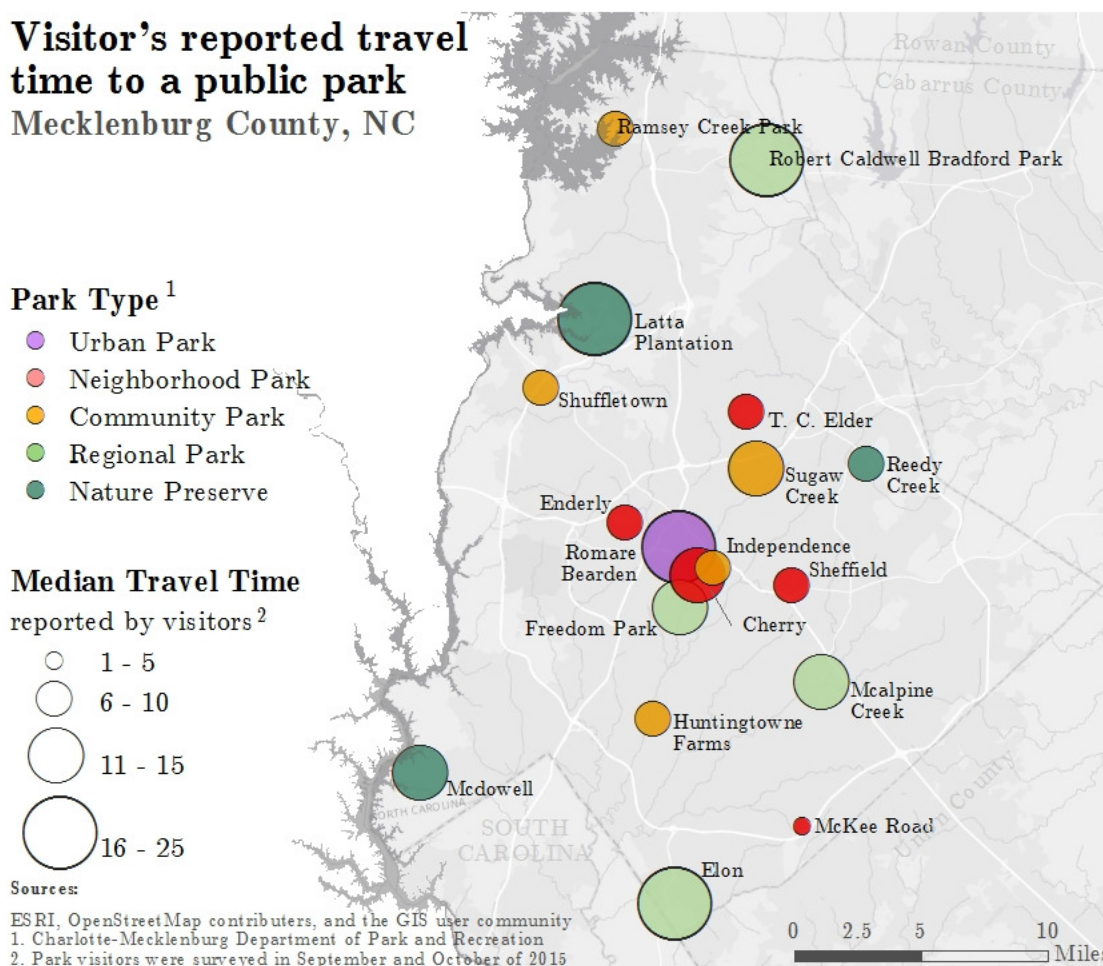


Figure 27: Visitor's reported travel time per park type

Homeownership, income, age, education level, gender, body mass index and

having a gym membership as reported by visitors were not associated with their reported time traveled to parks within a 90% confidence level. When testing for impact of race or ethnicity, the park type they were surveyed at and the walk time to the closest park however, the Likelihood Ratio Chi-Square suggests that at least one of the predictors' regression coefficient is not equal to zero in the model (99% confidence level). To make the interpretation of the ordinal logistic regression easier, I use three park types instead of five. For that, neighborhood and community parks are merged into one category and regional parks and nature preserves into a second category. Urban parks are kept as a separate category.

Table 6: Summary of reported travel time per park type after types were aggregated

Mecklenburg Residents			
Reported travel time (in minutes)			
Park type	median	p95	N
Neighborhood and Community	10	30	191
Regional and Nature Preserves	15	30	172
Urban	17.5	45	28

In Table 6 we can see that the medians and 95th percentiles stay robust even after reducing the data to three categories. The results of the Multinomial Logistic Regression (see Figure 28 on page 76) are robust when comparing the results using three categories instead of five.

At the 99% confidence level, the order logit for residents that were surveyed at neighborhood and community parks to report a higher travel time to the park they were surveyed at is 0.72 less than residents surveyed at regional parks and nature preserves when the other variables in the model are held constant. At a 99% confidence level, the order logit for visitors who live within a 10-minute walk of a park to have a higher travel

tine to the park they were surveyed is 0.86 less than visitors who live more than a 10-minute walk from a park when the other variables in the model are held constant.

```
Iteration 0:  log likelihood = -850.68193
Iteration 1:  log likelihood = -825.54161
Iteration 2:  log likelihood = -825.16585
Iteration 3:  log likelihood = -825.16558
Iteration 4:  log likelihood = -825.16558
```

```
Ordered logistic regression      Number of obs   =      364
                                LR chi2(4)           =      51.03
                                Prob > chi2           =      0.0000
                                Pseudo R2            =      0.0300

Log likelihood = -825.16558
```

travel_time	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
park_type_simple						
Neighborhood or Community Park	-.7183887	.1981595	-3.63	0.000	-1.106774	-.3300033
Urban Park	.7404482	.4239779	1.75	0.081	-.0905333	1.57143
cp_walk_time_simple						
10 minutes or less	-.8591016	.2103303	-4.08	0.000	-1.271341	-.4468619
ethnicity_hisp_latino						
Yes	.8813915	.3016517	2.92	0.003	.2901651	1.472618
/cut1	-5.193504	.5250987			-6.222678	-4.164329
/cut2	-3.570988	.2776475			-4.115167	-3.026809
/cut3	-3.270224	.2529523			-3.766001	-2.774446
/cut4	-3.226422	.2497363			-3.715896	-2.736948
/cut5	-3.184038	.2467058			-3.667572	-2.700504
/cut6	-1.718138	.177322			-2.065683	-1.370593
/cut7	-1.685831	.1763482			-2.031468	-1.340195
/cut8	-1.431607	.1696352			-1.764086	-1.099128
/cut9	-1.376195	.168419			-1.706291	-1.0461
/cut10	-.3925818	.1548874			-.6961554	-.0890081
/cut11	-.3805837	.1548399			-.6840643	-.0771032
/cut12	-.3565144	.1547456			-.6598101	-.0532186
/cut13	-.3444669	.1547028			-.6476789	-.0412549
/cut14	.5316084	.1583316			.2212841	.8419327
/cut15	.5466156	.1585329			.2358967	.8573345
/cut16	.5768881	.1589606			.2653311	.8884452
/cut17	1.16003	.1721695			.8225836	1.497476
/cut18	1.564267	.1877104			1.196362	1.932173
/cut19	2.966426	.2952687			2.38771	3.545142
/cut20	3.325534	.3408001			2.657578	3.99349
/cut21	3.561099	.3761899			2.823781	4.298418
/cut22	3.861567	.4286224			3.021483	4.701652
/cut23	4.569877	.5924322			3.408731	5.731023
/cut24	4.977318	.7194549			3.567212	6.387424
/cut25	5.673024	1.00874			3.695931	7.650118

Figure 28: Results of the ordinal logistic regression for research question 4

Finally, at a 99% confidence level, the order logit for Hispanics and Latinos to report a higher travel time to the park they were surveyed at is 0.88 more than non-Hispanics and non-Latinos when the other variables in the model are held constant. When I run the same model based on all park visitors (residents and non-residents), all factors that were significant remain significant and the multinomial logits are in the same direction and have a similar size.

2.3. Physical Exercise at Public Parks

Visitors were asked if they came to the park they were surveyed at to engage in physical activity that day. Table 7 shows that out of 385 Mecklenburg County residents, 253 (66%) responded they did come to the park that day to engage in physical activity. The median age and BMI is similar for visitors who did and did not come for physical activity.

Table 7: Summary of responses about engagement in physical activity

Engagement in physical activity	Mecklenburg Residents	
	Yes	No
number of respondents (N)	253	132
median age	34	33
median bmi	25.2	25.9
median access score	4	5

The median perceived access score for residents who came to do physical activity was 1 point lower compared to the median for residents who did not come to engage in physical activity.

Visitors were also asked how often they do engage in physical activity at parks. Table 8 shows that out of 383 Mecklenburg County residents, 227 (59%) responded they

used parks for physical activity regularly, 126 (33%) used them sometimes and 30 (8%) never went to parks to engage in physical activity. The median age and BMI is similar among the three respondent groups. The median perceived access score is 0.5 point lower for residents who sometimes go to parks to engage in physical activity.

Table 8: Summary of responses about frequency of engagement in physical activity at parks

Mecklenburg Residents			
Engage in physical activity at parks	Regularly	Sometimes	Never
number of respondents (N)	227	126	30
median age	33.5	31	33
median bmi	25.4	24.7	26
median access score	5	4.5	5

Breaking down the response by park type (see Table 9), the majority of residents did engage in physical activity at regional parks and nature preserves the day they were surveyed. Urban parks on the other hands attract mostly residents that do not come to engage in physical activity.

Table 9: Summary of responses about engagement in physical activity by park type

Mecklenburg Residents			
Engagement in physical activity	Yes	No	N
Neighborhood parks	67%	33%	58
Community parks	57%	43%	127
Regional parks	72%	28%	92
Nature preserves	83%	17%	81
Urban parks	28%	72%	29

Similar results are presented in Table 10, which shows that a majority of residents engage in physical activity at neighborhood, community and regional parks on a regular basis (defined as once a week or more). For nature preserves and urban parks, half of the

respondents did say they engaged in physical activity regularly and the other half sometimes engaged in physical activity or never.

Table 10: Summary of responses about frequency of engagement in physical activity by park type

Mecklenburg Residents				
Engage in physical activity at parks	Regularly	Sometimes	Never	N
Neighborhood parks	67%	23%	10%	60
Community parks	60%	32%	8%	128
Regional parks	62%	30%	8%	92
Nature preserves	53%	41%	6%	80
Urban parks	48%	44%	8%	25

Iteration 0: log likelihood = -245.83094
 Iteration 1: log likelihood = -231.58793
 Iteration 2: log likelihood = -231.52272
 Iteration 3: log likelihood = -231.52271

Logistic regression

Number of obs = 381

LR chi2(3) = 28.62

Prob > chi2 = 0.0000

Pseudo R2 = 0.0582

Log likelihood = -231.52271

pa_today	Coef.	Std. Err.	z	P> z	[95% Conf. Interval]	
park_type_simple						
Neighborhood or Community Park	-.7093627	.2431216	-2.92	0.004	-1.185872	-.2328531
Urban Park	-2.054372	.4587526	-4.48	0.000	-2.953511	-1.155234
ethnicity_afr_am_black						
Yes	-.2180931	.242034	-0.90	0.368	-.6924711	.2562849
_cons	1.22904	.1874307	6.56	0.000	.8616827	1.596397

Figure 29: Results of the logistic regression for research question 5

Homeownership, income, age, education level, gender, body mass index and having a gym membership as reported by visitors were not associated with whether or not they engaged in physical activity the day they were surveyed (90% confidence level). When testing for impact of the park type they were surveyed and ethnicity or race however, the Likelihood Ratio Chi-Square shows that at least one of the predictors' regression coefficient is not equal to zero in the model at a 99% confidence level (see

Figure 29).

At the 99% confidence level, the order logit for residents that were surveyed at Romare Bearden park (urban park) to report coming to that park to engage in physical activity that day is 2.05 less than residents surveyed at regional parks and nature preserves when the other variables in the model are held constant. At an 99% confidence level, the order logit for residents that were surveyed at neighborhood and community parks to report coming to that park to engage in physical activity that day is 0.71 less than residents surveyed at regional parks and nature preserves when the other variables in the model are held constant. Finally, the order logit for blacks and African Americans report coming to that park to engage in physical activity that day is 0.22 less than non-blacks and non-African Americans when the other variables in the model are held constant. However, the latter logit is not significant at the 90% confidence level. When we run the same model based on all park visitors (residents and non-residents), we find that all factors that were significant, stay significant and the multinomial logits are in the same direction and have a similar size.

3. Park Availability in Mecklenburg County

Using the kernel density estimation on the locations of parks, recreation facilities, golf courses and entrances to greenways that are public, Figure 30 shows contour lines of equal number of parks per square mile. It is important to mention that each location accounts equally in the density estimation. This map shows that the highest density of parks per square mile are found around the Charlotte business district. On the background are the percent households within each neighborhood that have a park or recreational facility within a 0.5-mile straight line as reported by the Charlotte-Mecklenburg Quality

of Life Explorer. The neighborhoods with high access do highly coincide with areas that offer a high density of parks and recreational facilities per square mile.

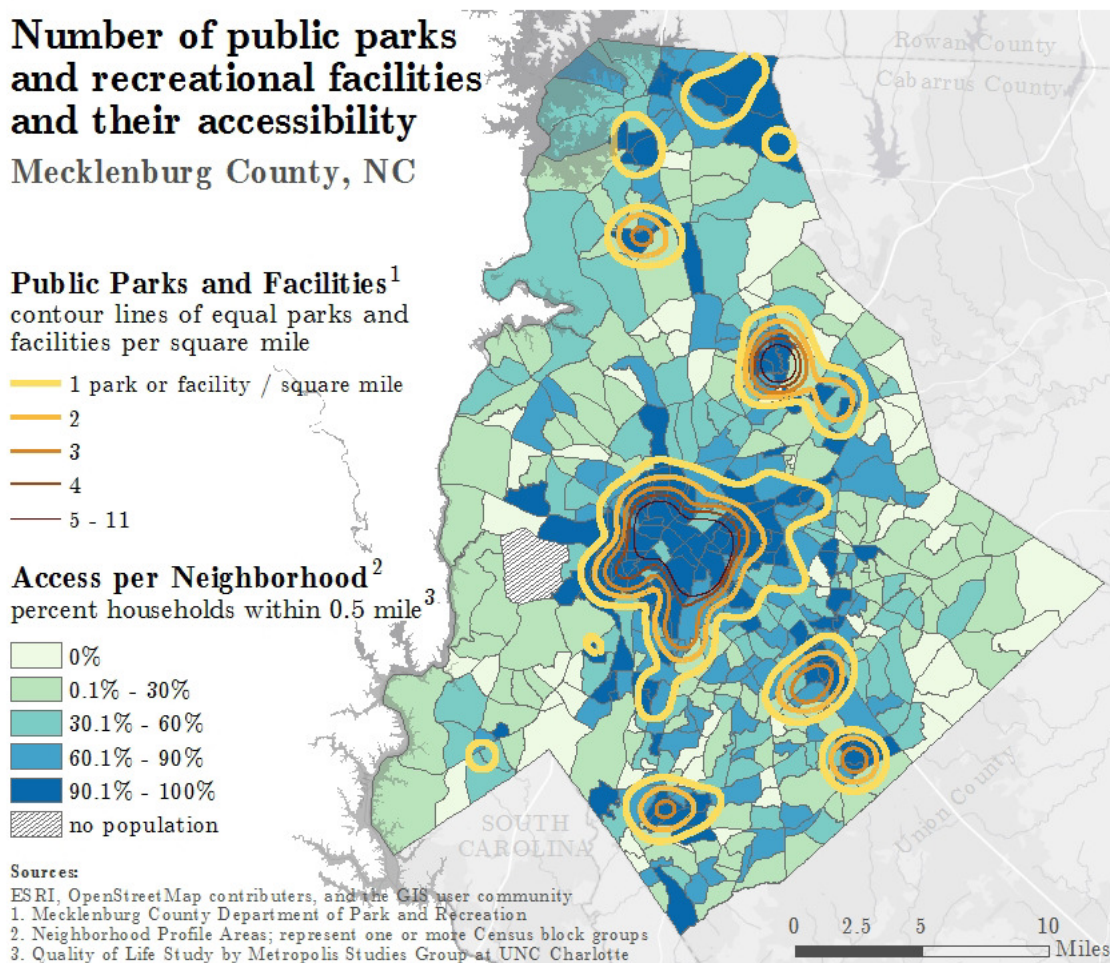


Figure 30: Number of public parks and recreational facilities and their accessibility

The outcomes of both analyses presented in Figure 30, however, do not account for non-public parks. Using the kernel density estimation on the locations of parks and other recreation opportunities that are not managed by the Department of Park and Recreation, Figure 31 shows contour lines of equal number of private parks per square mile. Here again, each location accounts equally in the density estimation. This map

suggests that the highest density of private parks and recreational opportunities per square mile are found in south and east of the Charlotte business district as well as at the northern end of Mecklenburg County. On the background are the percent blacks within each neighborhood as reported by the Charlotte-Mecklenburg Quality of Life Explorer. The neighborhoods with lower densities of private parks and recreational opportunities highly coincide with areas with neighborhoods that have a higher percent blacks.

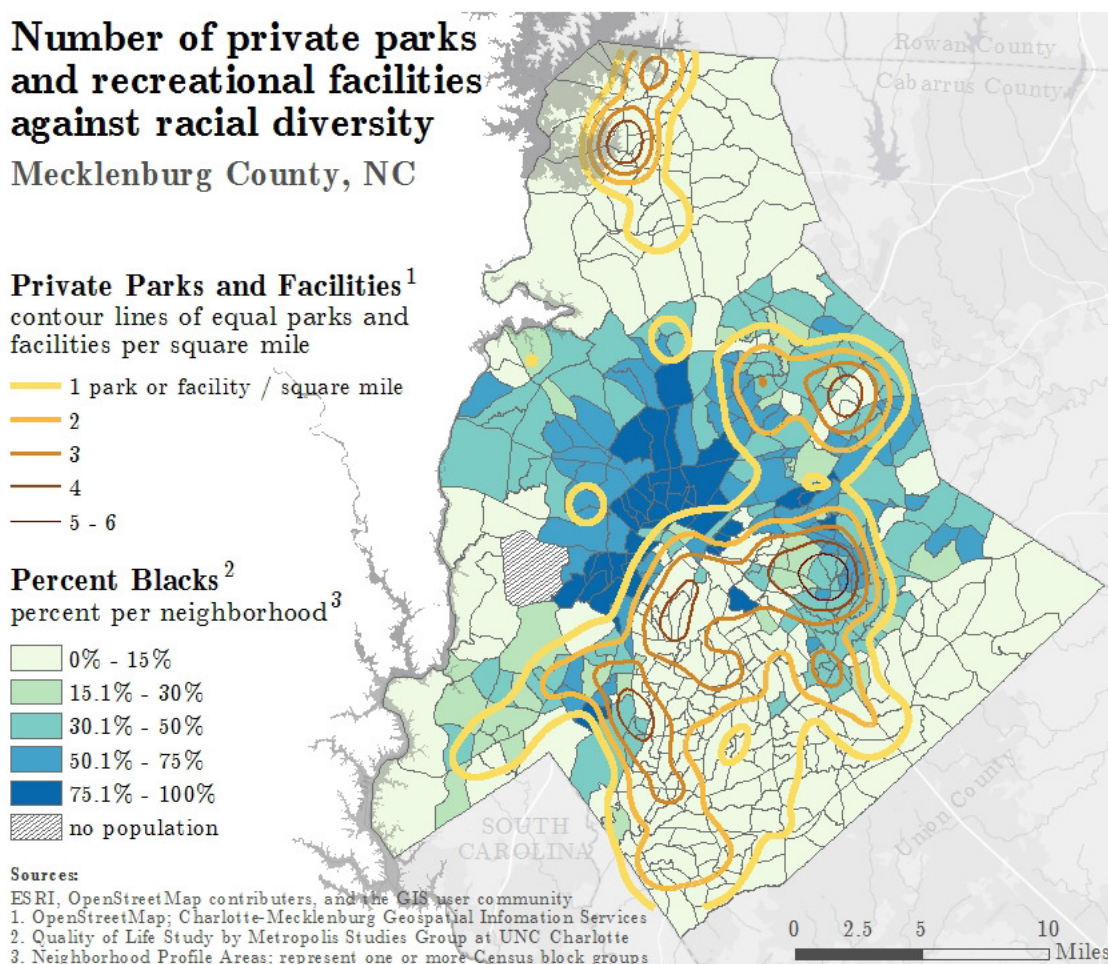


Figure 31: Number of private parks and recreational facilities against racial diversity

Number of parks and recreational facilities against population density

Mecklenburg County, NC

Public and Private Facilities¹
contour lines of equal parks and facilities per square mile

- 1 park or facility / square mile
- 2
- 3
- 4
- 5 - 13

Population Density²
people per acre per neighborhood³

- 0 - 1
- 2 - 4
- 5 - 8
- 9 - 16
- 17 - 32
- no population

Sources:

- ESRI, OpenStreetMap contributors, and the GIS user community
- 1. OpenStreetMap; Charlotte-Mecklenburg Geospatial Information Services
- 2. Quality of Life Study by Metropolis Studies Group at UNC Charlotte
- 3. Neighborhood Profile Areas; represent one or more Census block groups

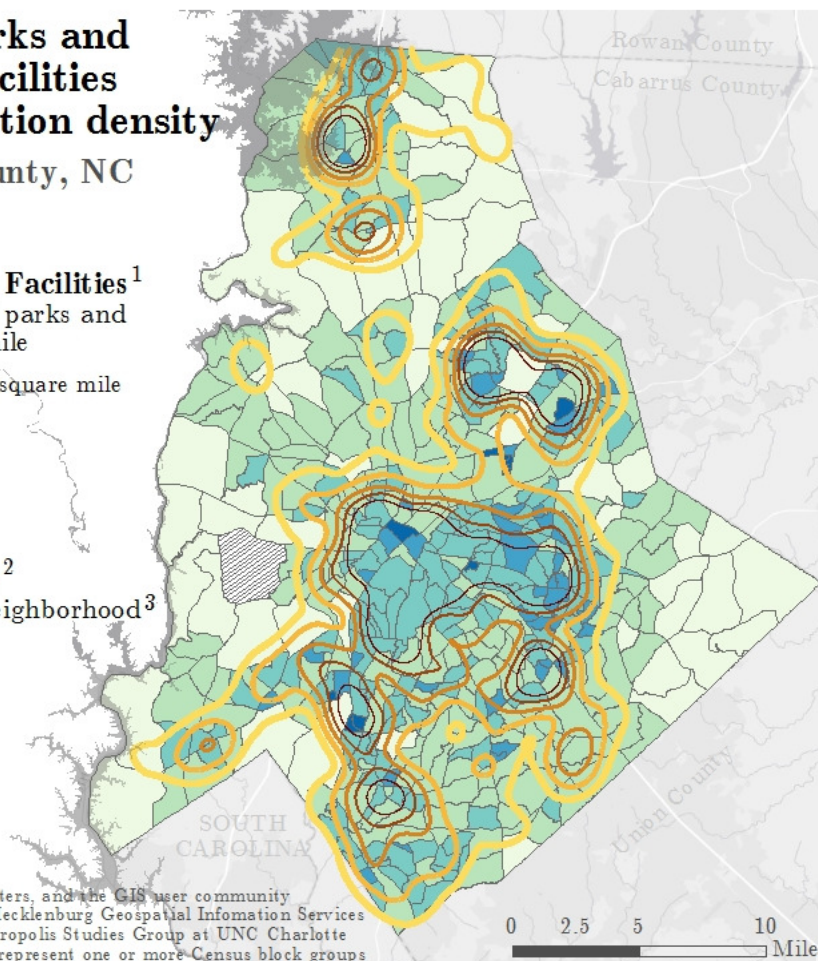


Figure 32: Number of parks and recreational facilities against population density

Using the kernel density estimation on the locations of parks and other recreation opportunities that are public and non-public, Figure 32 shows contour lines of equal number of parks per square mile. Each location accounts equally in the density estimation. This map shows a very different distribution of maps compared to the one where only public parks were accounted for. On the background is the population density as reported by the Charlotte-Mecklenburg Quality of Life Explorer. The neighborhoods with higher population densities tend coincide with areas with higher densities of parks and recreational facilities.

Number of parks and recreational facilities (offset by attraction) and health insurance Mecklenburg County, NC

Public and Private Facilities¹ contour lines of equal parks & facilities per sq. mile (offset by attraction²)

- 1 park or facility / square mile
- 2
- 3
- 4
- 5 - 14

Medicaid or Statewide Plan³ % beneficiaries per neighborhood⁴

- 0 - 10
- 10.1 - 20
- 20.1 - 30
- 30.1 - 50
- 50.1 - 94
- no population

Sources:

- ESRI, OpenStreetMap contributors, and the GIS user community
- 1. OpenStreetMap; Charlotte-Mecklenburg Geospatial Information Services
- 2. Measured based on park size, number of amenities and safety
- 3. Quality of Life Study by Metropolis Studies Group at UNC Charlotte
- 4. Neighborhood Profile Areas; represent one or more Census block groups

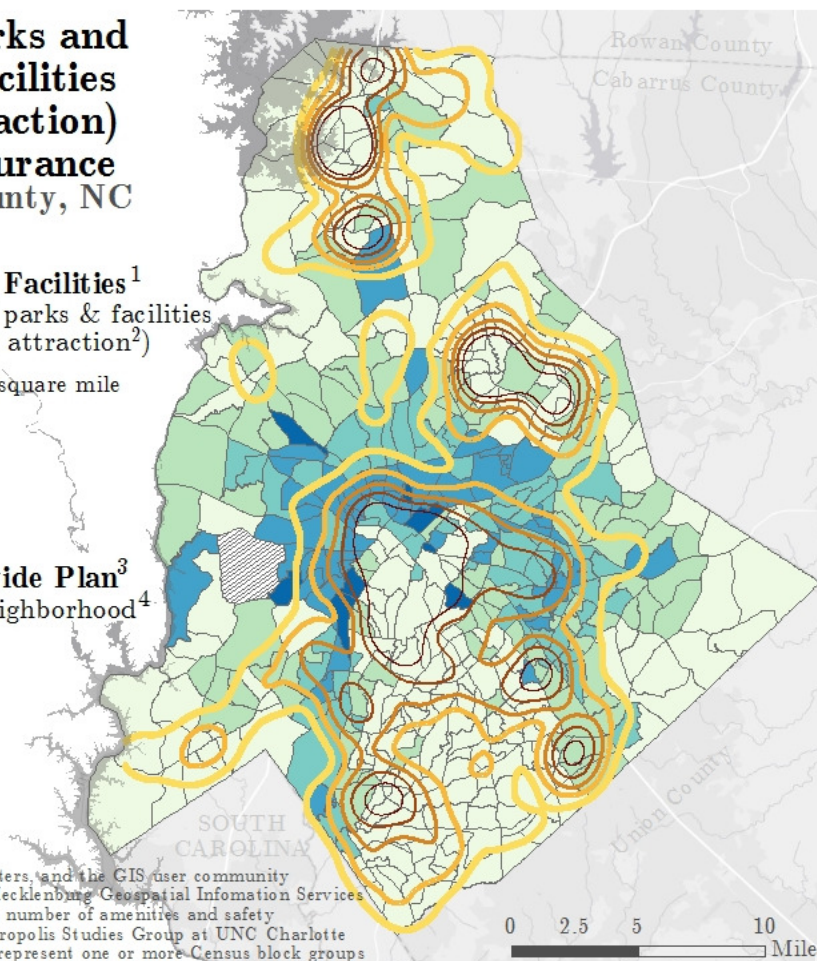


Figure 33: Number of parks and recreational facilities (offset by attraction) and health insurance

The surveys pointed out that safety, available amenities, aesthetics and size are important to many park visitors. Safety was reported to be the most important characteristic. To account for those characteristics, the kernel density estimation was weighed based on (1) violent crime within 200 meters of the park or recreational facility's entrance, (2) number of amenities available and (3) the size of the park. Instead of using absolute values, all locations were ranked based on these three characteristics. Based on their rank, each park was given a value between 0.25 and 1.75. For example, a park that is among those with the highest number of violent crimes will be given a value

of 0.25 and parks among those with the lowest number of violent crimes will be given a value of 1.75. In the kernel density estimation, the park with high crime will not account for 1 park, but for 25% of a park because it is likely that people want to avoid this park because of criminal activity. On the other hand, the park with low crime will not account for 1 park, but for 175% of a park because it will attract more people thanks to its safety. From the surveys, we learned that visitors value crime more than amenities and value amenities more than park size. Therefore, the overall attraction of each park was calculated as follows: To calculate attraction to each park, the average value of each park on crime, amenities and park size is calculated, which will be a value between 0.25 and 1.75. However, we weighted each value according to the survey results: the crime value is weighted three times higher than that of park size and the amenities value is weighted twice as much as that of park size. Figure 33 presents the map of contours with equal park and recreation densities including public and private parks and weighing for safety, amenities and size.

On the background is the percent beneficiaries of Medicaid or the statewide insurance plan as reported by the Charlotte-Mecklenburg Quality of Life Explorer. The neighborhoods with lower percentages of beneficiaries coincide with areas with higher densities of parks and recreation.

4. What is the Sentiment of Online Comments at Parks in Mecklenburg County?

Using the opinionfinder system (Wilson, Wiebe, & Hoffmann 2005) to process all reviews left by online users on Google and Foursquare about parks and recreational facilities, the overall sentiment was identified for each park.

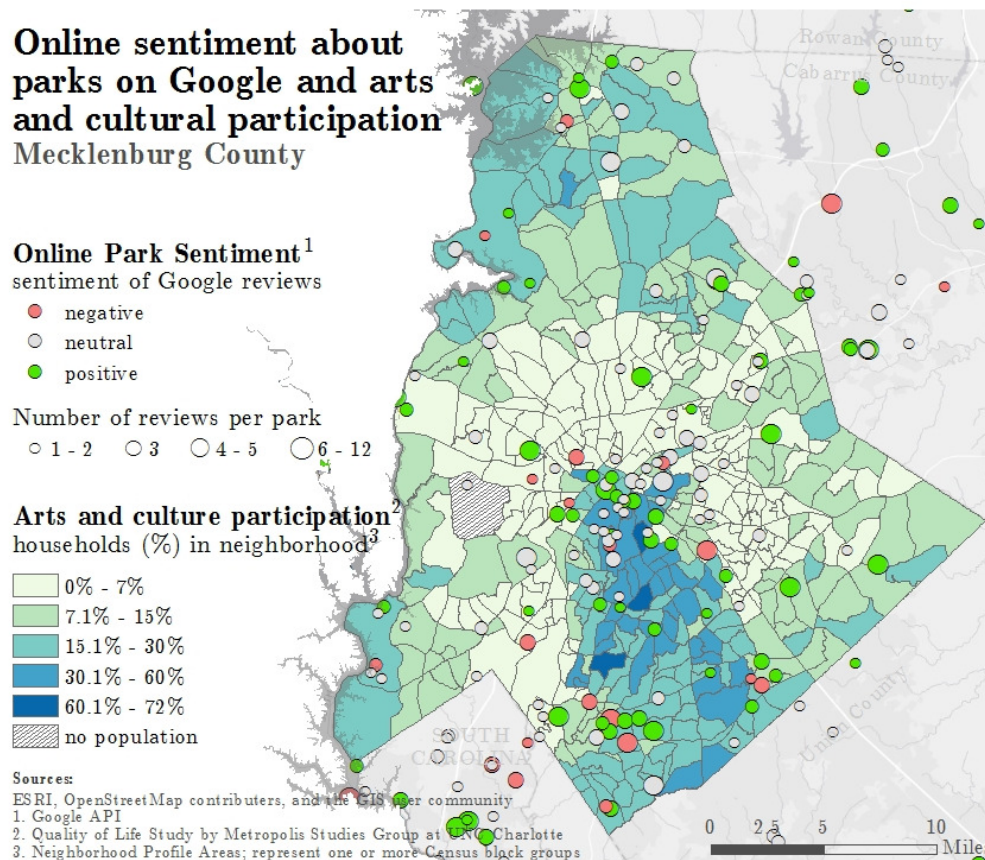


Figure 34: Online sentiment about parks on Google and arts and cultural participation

Figure 34 (on page 86) shows all locations in the Google Maps database that are categorized under “park” and at which at least one review was posted. The color of the dot refers to the overall sentiment at that location based on all reviews left by users. A red dot refers to a negative sentiment, a green dot to a positive sentiment and a grey dot refers to a neutral sentiment. The size of each dot represents the number of reviews left by users at that location. On the background is the percent households within each neighborhood that participate in arts and cultural activities sponsored by the NC Arts Council State Funds (ASC) as reported by the Charlotte-Mecklenburg Quality of Life Explorer. Participation in art and cultural activities may indicate a bigger engagement of residents

in their community.

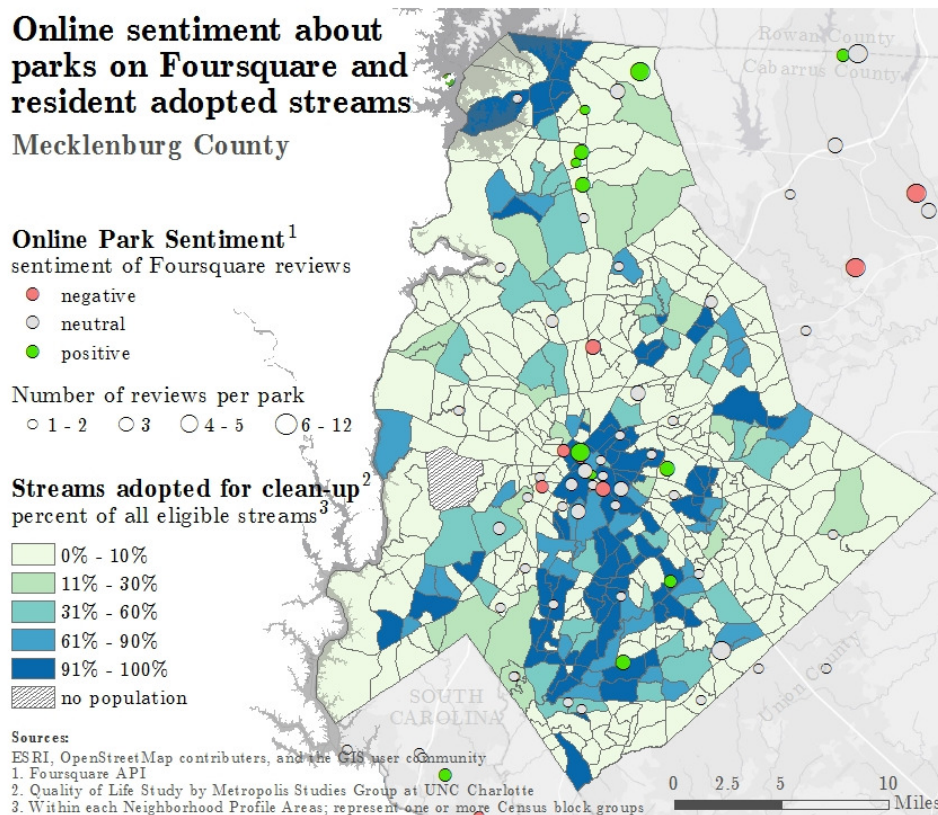


Figure 35: Online sentiment about parks on Foursquare and resident adopted streams

Figure 35 (on page 87) shows all locations in the Foursquare database that are categorized under “park” and at which at least one review was posted. Here again red dots refer to a negative sentiment in the reviews of users, a green dot to a positive sentiment and a grey dot refers to a neutral sentiment. The size of each dot represents the number of reviews left by users at that location. On the background is the percent of all eligible streams that were adopted by residents for clean-up (adopt-a-stream movement) within each neighborhood as reported by the Charlotte-Mecklenburg Quality of Life Explorer. This may indicate commitment and engagement to keep one’s appealing.

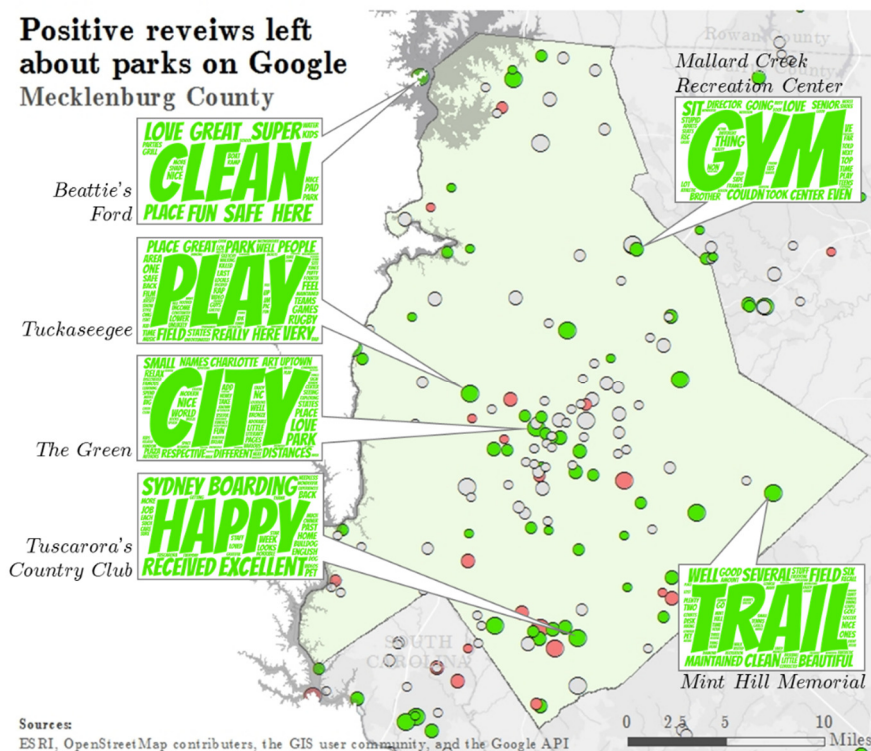


Figure 36: Positive reviews left about parks on Google

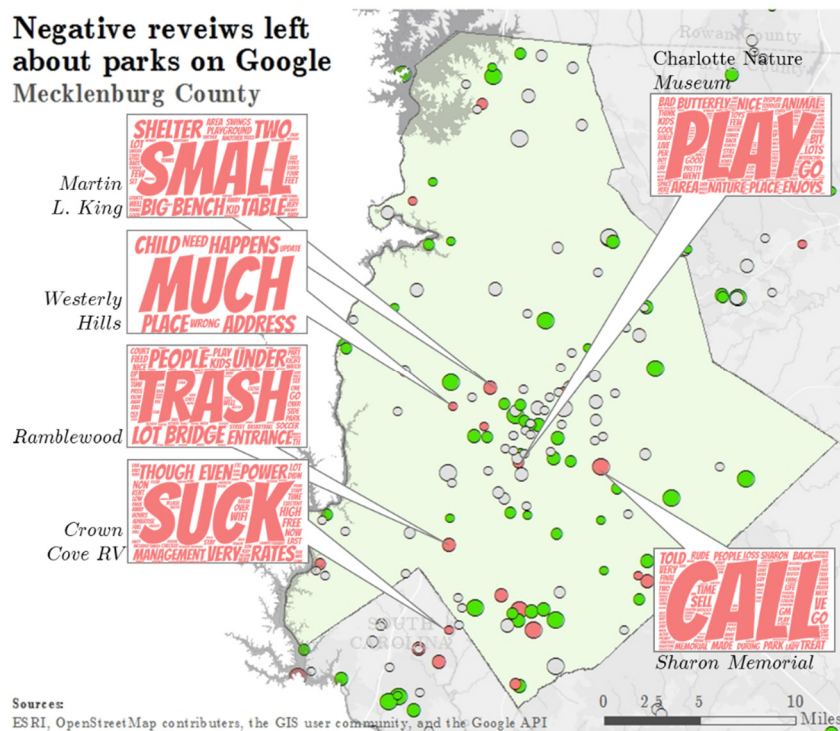


Figure 37: Negative reviews left about parks on Google

To summarize the reviews left by Google users at each location, a word cloud (using tagul³ tools) was generated for 6 parks where the overall sentiment of the reviews was positive (see Figure 36). The most common word that appears in reviews about Beattie's Ford Park is "clean", whereas reviews about The Green – which is a plaza near Charlotte's business district, contained the word "city" most frequently. This figure shows that different park locations generate different topics of discussion based on their available amenities. Figure 37 summarizes the reviews left by Google users using a word cloud for 6 parks where the overall sentiment of the reviews was negative. The most common word that appears in reviews about Ramblewood Park is "trash", whereas reviews about Martin Luther King Park contained the word "small" most frequently. Here again, different park locations generate different topics of discussion. Many people called the Sharon Memorial Park before planning their visit and were not pleased because of the rudeness of the people receiving their call. It is important to mention that Sharon Memorial is a cemetery and Crown Cove is a recreational vehicle (RV) park.

Some locations under the "park" category within the Google database do not fit entirely fit out criteria for parks and recreational facilities within the scope of increasing physical activity.

³ Tagul provides a free and online tool to generate word clouds based on a text: tagul.com

DISCUSSIONS

1. Modeling Spatial Accessibility to Parks Using the VFCA Method

A case study on public parks in Mecklenburg County (NC) illustrated the VFCA method under three different scenarios. In the first scenario, parameters were chosen to replicate those of the 2SFCA method; meaning that catchments sizes around each park were equal and park size was the only characteristic used to weigh accessibility scores. In this scenario (scenario I) with the car as transport mode, block groups in the western part of the county rank high on park access and block groups in the eastern part all result in low ranks to the exception of a handful. Theoretically, this makes sense since 7 of the 8 largest nature preserves are located in the western part of the county (see Figure 11 on page 34). These parks drive the results of the model because size is the weighing factor. For the other three modes of transport, block groups on the western edge of the county rank high on accessibility, here too, matching the location of larger parks. However, because of the bigger travel limitations compared to the car, the patterns show less clustering and tend to follow the respective underlying network of each mode. The second scenario presented also took park size as sole factor dictating the attractiveness of a public park but makes the catchment of each park vary based on the attractiveness. Under that scenario (scenario II) with the car as transport mode, northern and outlying suburban areas of Mecklenburg County enjoy much higher levels of accessibility. Here too, this was expected given the significant proportion of larger parks located in the northern periphery of the county. On the other hand, when size and amenities were weighted equally (scenario III), populations living closer to the center of Mecklenburg County (Charlotte business district) and along the main road arteries experienced higher

accessibility values.

Overall, the different parameters used to evaluate spatial accessibility can generate very contrasting outcomes. Consequently, caution must be adopted (1) when choosing a spatial access model, (2) when interpreting spatial patterns of accessibility and (3) when providing policy advice based on the results of the study. Clearly, there is no “one size fits all” model; the circumstances of the modeling effort should help dictate the parameters being used and ideally, more than one model should be run to ascertain a more complete image of the accessibility landscape of an urban area. The robustness of the results can be supported when several accessibility measures generate similar outcomes. The VFCA model has several strengths. First, the concept of attraction (Delmelle, Li & Murray, 2012; Farhan & Murray, 2006; Roy & Thill, 2004) is explicitly modeled by a weighted objective. The catchment area of each public park contracts or expands based on the level of attraction and the preferred mode of transportation. Second, the methodology is deployable to other commodities such as transportation infrastructure, schools, farmer’s markets and medical centers. It would also be pertinent to implement in cities in other parts of the world where urbanization and transportation issues vary dramatically from the southern U.S. city featured in this study. Finally, since all parameters are designed to be flexible, this approach has the capability to support scenario analysis, a key exercise for sparking critical and strategic thinking in the planning process (Xiang & Clarke, 2003).

2. Accessibility to Parks in Mecklenburg County: Potential vs. Revealed

Residents of Mecklenburg County (18 years or older) surveyed at a public park between September and November of 2015 were satisfied with their access to parks.

From all visitors surveyed, men were more satisfied with their access to parks compared to women. On a scale of 1 out of 5 (1 being very poor and 5 being very good access), the median rating by Mecklenburg residents was 5. This finding is an important argument against the poor ranking Charlotte is given by the Trust for Public Land (TPL) since 2012.

The method used by the TPL gives the highest weight to areas that have a public park within a 10-minute walk (see Figure 6 on page 23). Walking access to parks is one of the main reasons Charlotte ranks low on their *ParkScore*® compared to other U.S. cities. Similar to their conclusion, all accessibility patterns from the VFCA method resulted in very low scores when walking. From the residents of Mecklenburg County that were surveyed at public parks, 29.7% reported their closest park to be at a 10-minute walking distance or less. Ordinal logit regression suggests that residents who have a lower access to parks by foot is associated with a lower perceived access to parks in general. Visitors who reported having a park within a 10-minute walk or less were significantly more satisfied with their access to parks compared to those who had to walk more than 10 minutes. This suggests that Mecklenburg County residents do value being able to walk to a park.

All patterns from the VFCA resulted in higher number of block groups with high access scores when traveling by car. Although Charlotte is known to be a car-centric city, this does not necessarily imply residents prefer traveling by car. In fact, surveys show that park visitors who did not own nor have access to a car were less satisfied with their access to parks compared to those who did. This finding suggests that Mecklenburg residents do travel using other modes of transport than the car, and that when they do,

they experience barriers that significantly lowers their perceived access to parks.

The TPL gives a high weight to areas that have a park at a walking distance and also weigh accessibility by the total acreage of parkland available. Size being one of the driving factors in their model; the areas of high and very high need (see Figure 6 on page 23) match block groups with low park access by car in the results of the VFCA under scenario I. Even though their model is weighted more heavily by walking access to parks, it does seem like their model is driven by the acreage of parkland available in the northwestern part of the county. The Charlotte-Mecklenburg Quality of Life Explorer measures accessibility to parks differently; in their model park access is considered high if the percentage of households in a neighborhood that are within a 0.5-mile distance from a public park is high. The neighborhoods that exhibit a low percentage of households with park access (see Figure 7 on page 25) do not closely coincide with areas of high or very high need estimated by the TPL (see Figure 6 on page 23). However, the pattern by the Charlotte-Mecklenburg Quality of Life Explorer does resemble the patterns obtained from the VFCA method under scenario III (see Figure 22 on page 66). This suggests that their model is not driven by the size of parks. When asked about the importance of park characteristics, visitors report safety to be the most important characteristic of a park. Women however, rate the importance of safety higher than men do. Given that women rate their access to parks significantly lower and rate the importance of safety significantly higher than men, this might suggest that women experience a bigger barrier to access parks because of safety concerns. Although the size of a park was reported to be important to park visitors, it is a less important characteristic compared to aesthetic and design of the park and available amenities.

3. Willingness to Travel to Parks

Visitors indicated that they drive between 10 and 30 minutes to public parks. The median travel time by Mecklenburg residents to neighborhood and community parks was 10 minutes while the median to regional parks and nature preserves was 15 minutes. The median travel time to Romare Bearden park (an urban park) was 17.5 minutes which indicates that residents travel a longer distance to get to an urban park. Romare Bearden park is located in a walk-friendly area of the City of Charlotte, where other activities such as baseball games, bars, restaurants and museums are accessible by foot. Visitors at Romare Bearden park may not travel primarily for the park. Instead, this location provides opportunity to combine multiple activities in one trip, which may explain the longer travel times compared to other park types.

In the VFCA method, catchment areas of 10 minutes for neighborhood parks, and up to 45 minutes for nature preserves were used. Given the reported travel times by Mecklenburg residents, these catchment sizes clearly overestimate their willingness to travel to parks. It is also important to notice that a park type does not show consistent catchment areas (see Figure 27 on page 74). Instead some neighborhood parks may attract visitors that live outside the neighborhood while certain regional parks may have smaller catchment sizes than expected. Therefore, using park type to define the size of catchment areas may not be appropriate. It is important to mention that the travel times reported here reflect those of residents that are already visiting parks. Therefore, they do not provide insights about the willingness to travel to parks for residents that do not visit parks, which may be lower than that of park visitors.

The youth (ages 18-24) seems to be willing to travel slightly longer distance

compared to people ages 60 years or older. Individuals with an income lower than the poverty line (\$24,000 per year, which may also include students) and those with an annual household income higher than \$150,000, seem to be willing to travel longer times to get to a park. This might indicate a bigger time availability for both income categories. Finally, willingness to travel is grossly impacted depending on the availability of a car. People who do not have access to or own a car seem to travel up to 90 minutes to get to a park. This is an important finding because it shows that Mecklenburg residents who do not have a car, still seek to go to parks.

4. The Use of Parks as a Place for Physical Activity

From all visitors surveyed at public parks, 59% reported using parks for physical activity on a regular basis. Additionally, 66% of visitors reported coming to the park they were surveyed at to engage in physical activity. Interestingly, visitors who came to engage in physical activity perceive their access to parks to be lower than those who did not come for that reason. This may indicate a need by individuals who already use parks as a place for physical activity to have easier access to parks. Additionally, these park users may be better acquainted with the park itself and its equipment, which may make them more aware of issues at parks (e.g. defect water fountains). Nature preserves and regional parks have the highest percentage of visitors who report using the park as a place for physical activity. Although a lower percentage of visitors reported engaging in physical activity at neighborhood and community parks, still over 50% do use parks for this purpose. On the other hand, less than 30% of respondents at Romare Bearden park reported coming to the park to engage in physical activity. This is an important finding because it suggests that certain park types are more appealing to visitors as a place for

physical activity, which may be helpful for prevention of heart disease.

5. Access to Parks and Environmental Justice

Using a more complete inventory of parks and recreational facilities, park availability per square mile was illustrated in Figure 30. This pattern is very similar to the one obtained by the Quality of Life Explorer (see Figure 7). More parks per square mile coincide with neighborhoods that have a higher percentage of households with park access within 0.5 mile. Both maps show a large portion of Mecklenburg County with little or no access to parks. However, from the surveys, residents seem to be satisfied with their level of access in Mecklenburg County. This disparity between what studies show and what park visitors express, may be explained by two factors. First, residents of Mecklenburg County that do not visit parks are not represented in these results. Their perceived access to parks may bring down the level of satisfaction reported in this study. Second, one big limitation of previous studies is the omission of non-public parks when evaluating access to parks. For this study, I generated an inventory of non-public parks to address this issue. The majority of parks in this inventory represent recreational amenities that are managed by homeowner's associations such as basketball courts or outdoor pools. This source of data is extremely significant in two ways; (1) it contains two times more parks than the public park inventory and (2) it suggests an important issue of environmental justice. From Figure 31 (on page 82), the number of private parks per square mile is lowest in neighborhoods that show high percentages of blacks. Moreover, the areas of high and very high need highlighted in the study by the TPL (Figure 6 on page 23) correspond to areas with a high number of private parks per square mile. The results of their study is therefore misleading.

The availability of public and non-public parks per square mile in Mecklenburg County (public and non-public) were mapped in Figure 32 (on page 83), which better reflect true availability of parks in the county. Using insight from the surveys, I weighed each park based on safety, size and number of available amenities (see Figure 33 on page 84). Safety accounted two times more than size or amenities. This map is presented with the percent beneficiaries for Medicaid or the statewide medical plan per neighborhood which could be seen as a representation of individuals in need of free and accessible options to reduce their risk for disease. This map, suggests that areas directly North and West of the Charlotte business district should be prioritized by planners of park and recreational facilities.

6. Real Time Monitoring of Park Satisfaction

Google and Foursquare reviews provide useful data about parks, which can be used to assess park satisfaction in the region. Both providers however, seem to receive reviews about parks that are in neighborhoods where residents show more community engagement. Therefore, this data source should be used as a complement rather than a substitute to assess park satisfaction. By extracting online reviews from additional media platforms, such as Twitter or Yelp, more content can be collected and analyzed. Data-mining techniques such as sentiment analysis are freely available and easy to use, which makes this tool feasible for regular monitoring of park reviews.

RECOMMENDATION

First, caution must be adopted (1) when choosing a spatial access model, (2) when interpreting spatial patterns of accessibility and (3) when providing policy advice based on the results of the study. From this study, it is clear that there is no “one size fits all” model; the local context should help dictate the parameters being used and ideally, more than one model should be run to ascertain a more complete image of the accessibility landscape of an urban area. Second, the study carried out by the Trust for Public Land, which ranks cities by their score on park availability, is influential across the U.S. however the credibility of their results should be validated at the local level. Their maps showing levels of need for parks in particular, should not be used for decision-making purposes as they do not reflect the local context. Third, studies on park access need to validate the completeness of their park inventories. Publically available data, which is often managed by the local Department of Park and Recreation, often omit non-publically managed spaces such as homeowner’s association parks and recreational equipment. Fourth, safety is a more important characteristic for park visitors compared to park size. Many studies have used park size to weigh accessibility to parks because this data is usually easily attainable along with public park locations. Safety, is an especially important variable to consider, since this study suggests that women may face important barriers to parks because of safety concerns. Fifth, different park types attract individuals for different purposes. Regional parks and nature preserves attract the most visitors that come to engage in physical exercise. Prevention strategies for heart disease should look into parks as a place for regular physical exercise. Sixth, the Department of Park and Recreation in Mecklenburg County should be aware of the significant disparity in

availability of non-public parks by minorities. In Mecklenburg County, neighborhoods with a high percentage of blacks are associated with low availability of non-public parks, which may be an environmental justice concern. Seventh, it would be interesting to compare these findings with a city that is more walkable than Charlotte, and see if there too, residents without access to a car would perceive their access significantly lower. These results may support the need for an increase in the number of neighborhood parks rather than an emphasis on larger regional parks, which chiefly serve residents with a personal car. Finally, the use of online platforms that allow users to leave reviews about parks and recreational facilities should be considered as an additional source of data to monitor park satisfaction and needs of the residents.

CONCLUSIONS

Modeling the notion of access remains a challenge. In my dissertation, I argue that existing methods to evaluate equitable availability of parks have been somewhat disjoint from the intent for which parks are planned. Additionally, the notion of “access” often depends on the local context and needs of residents. After consultations with park planners of Mecklenburg County, I present a re-conceptualization of a popular spatial accessibility model initially developed for healthcare applications, in which I incorporate normative standards used by park planners. Then, to get a better sense of the local context and needs of residents, park visitors were surveyed and asked about their perceived access to parks, their transportation choices and their use of parks as place to engage in physical activity. Finally, I consult reviews about parks left by users on online platforms such as Google to illustrate the application of data mining techniques to monitor park satisfaction.

The accessibility model introduced in this article, along with the accompanying case study is beneficial for planners and policy makers looking to improve access to parks and recreational facilities in their area. This work has underscored the importance of planning for equity from a holistic perspective; transportation infrastructure, facility locations and associated level of service, safety and local perceptions are all critically important in shaping the accessibility landscape of an urban area. Increasing the total number or acreage of public parks may not always be the best outcome for some neighborhoods. Collecting park user’s information and experience is crucial and can help improve our understanding of park access while enhancing the specification of access models. However, this would require effective communication among different

administrations. Ultimately, access models that are sound have the potential to become an effective planning and policy tool to develop and communicate prevention strategies. As discussed in this study, the assessment and improvement of access to public parks holds great potential in the worldwide battle against heart disease.

FUTURE RESEARCH

For future research, it would be interesting to apply the VFCA method when non-public park locations are included as well. Also, the importance of park characteristics as reported by park visitors should be used to calibrate the parameters of the VFCA. Finally, this dissertation disputes many of the findings of the study by the TPL. However, it would make a better case to replicate the study by the TPL taking into account non-public parks. Also, since safety has been reported to be a more important park characteristic than park acreage, it would be interesting to replicate the TPL study taking into account crime rather than park acreage. Finally, more research needs to be done about the use of social media to monitor park satisfaction. For example, dictionaries that are tailored to interpret comments that relate to parks would be an important improvement to make.

REFERENCES

- Afzalan, N., & Muller, B. (2014). The Role of Social Media in Green Infrastructure Planning: A Case Study of Neighborhood Participation in Park Siting. *Journal of Urban Technology*, 21(3): 67-83.
- Barry, S. J. (2014). Using Social Media to Discover Public Values, Interests, and Perceptions about Cattle Grazing on Park Lands. *Environmental management*, 53(2): 454-464.
- Bauman, A., & Craig, C. L. (2005). The place of physical activity in the WHO Global Strategy on Diet and Physical Activity. *International Journal of Behavioral Nutrition and Physical Activity*, 2(10). doi - 10.1186/1479-5868-2-10
- Bedimo-Rung, A. L., Mowen, A. J., & Cohen, D. A. (2005). The significance of parks to physical activity and public health: a conceptual model. *American journal of preventive medicine*, 28(2), 159-168. doi: 10.1016/j.ampre.2004.10.024
- Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1): 1-8.
- Boone, C. G., Buckley, G. L., Grove, J. M., & Sister, C. (2009). Parks and people: An environmental justice inquiry in Baltimore, Maryland. *Annals of the Association of American Geographers*, 99(4), 767-787. doi - 10.1080/00045600903102949
- Brownson, R. C., Boehmer, T. K., & Luke, D. A. (2005). Declining rates of physical activity in the United States: what are the contributors?. *Annual Review of Public Health*, 26, 421-443.
- Casas, I. (2007). Social Exclusion and the Disabled: An Accessibility Approach. *The Professional Geographer*, 59(4), 463-477. doi - 10.1111/j.1467-9272.2007.00635.x
- CDC, Centers for Disease Control and Prevention. (1999). Achievements in Public Health, 1900-1999: Decline in deaths from heart disease and stroke--United States, 1900-1999. *Morbidity and Mortality Weekly Report*, 48(30), 649-656.
- CDC, Centers for Disease Control and Prevention. (2016). *Heart Disease Facts: Heart Disease in the United States*. Available at: cdc.gov/heartdisease/facts.htm.
- Ciuccarelli, P., Lupi, G., & Simeone, L. (2014). *Visualizing the data city: social media as a source of knowledge for urban planning and management*. Springer Science & Business Media.
- Cromley, E. K., & McLafferty, S. L. (2012). *GIS and Public Health (2nd ed.)*. New York, NY: The Guilford Press. isbn - 978-1-60918-750-7

- Curran, W., & Hamilton, T. (2012). Just green enough: contesting environmental gentrification in Greenpoint, Brooklyn. *Local Environment*, 17(9), 1027-1042.
- Dai, D. (2011). Racial/ethnic and socioeconomic disparities in urban green space accessibility: Where to intervene?. *Landscape and Urban Planning*, 102(4), 234-244. doi - 10.1016/j.landurbplan.2011.05.002
- Delmelle, E. C., & Casas, I. (2012). Evaluating the spatial equity of bus rapid transit-based accessibility patterns in a developing country: The case of Cali, Colombia. *Transport Policy*, 20, 36-46. doi - 10.1016/j.tranpol.2011.12.001
- DHHS, U.S. Department of Health and Human Services. (2010). *Healthy People 2020: Physical Activity*, Retrieved from: healthypeople.gov/2020/TopicsObjectives2020/overview.aspx?topicid=33
- Donabedian, A. (1973) *Aspects of Medical Care Administration: Specifying Requirements for Health Care*. Cambridge MA: Harvard University Press.
- Dony, C.C., Delmelle, E.M., & Delmelle, E.C. (2015) Re-conceptualizing accessibility to parks in multi-modal cities: A Variable-width Floating Catchment Area (VFCA) method. *Landscape and urban Planning*, 143: 90-99.
- Engel, G. L. (1977). The need for a new medical model: a challenge for biomedicine. *Science*, 196(4286), 129-136.
- Frieden, T. R. (2010). A framework for public health action: the health impact pyramid. *American journal of public health*, 100(4), 590-595. doi: 10.2105/AJPH.2009.185652
- Garcia Esparza, S., O'Mahony, M. P., & Smyth, B. (2010, September). On the real-time web as a source of recommendation knowledge. In *Proceedings of the fourth ACM conference on Recommender systems* (pp. 305-308). ACM.
- Guagliardo, M. F. (2004). Spatial accessibility of primary care: concepts, methods and challenges. *International Journal of Health Geographics*, 3, 3
- Gutiérrez, J., & García-Palomares, J. C. (2008). Distance-measure impacts on the calculation of transport service areas using GIS. *Environment and Planning B: Planning and Design*, 35(3), 480-503. doi - 10.1068/b33043
- Haskell, W. L., Lee, I. M., Pate, R. R., Powell, K. E., Blair, S. N., Franklin, B. A., . . . Bauman, A. (2007). Physical activity and public health: updated recommendation for adults from the American College of Sports Medicine and the American Heart Association. *Circulation*, 116(9), 1081- 1093. doi - 10.1161/CIRCULATION.107.185649

- Hill, J. O., Wyatt, H. R., Reed, G. W., & Peters, J. C. (2003). Obesity and the environment: where do we go from here?. *Science*, 299(5608), 853-855. doi: 10.1126/science.1079857
- Huff, D. L. (1964). Defining and estimating trade areas. *The Journal of Marketing*, 28(3), 34-38. doi - 10.2307/1249154
- IAPD, Illinois Association of Park Districts (2006) *Preserving open space in rapidly developing communities – mitigating urban sprawl*. Available at: [www-ilparks.org/resource/resmgr/files/position_paper_open_space_pd.pdf](http://www.ilparks.org/resource/resmgr/files/position_paper_open_space_pd.pdf)
- Khan, A. A., & Bhardwaj, S. M. (1994). Access to Health Care: A Conceptual Framework and its Relevance to Health Care Planning. *Evaluation & the Health Professions*, 17(1), 60–76. doi:10.1177/016327879401700104
- Koohsari, M. J., Kaczynski, A. T., Giles-Corti, B., & Karakiewicz, J. A. (2013). Effects of access to public open spaces on walking: Is proximity enough?. *Landscape and Urban Planning*, 117, 92-99. doi: 10.1016/j.landurbplan.2013.04.020
- Lee, G., & Hong, I. (2013). Measuring spatial accessibility in the context of spatial disparity between demand and supply of urban park service. *Landscape and Urban Planning*, 119, 85-90. doi - 10.1016/j.landurbplan.2013.07.001
- Lopez, R. P., & Hynes, H. P. (2006). Obesity, physical activity, and the urban environment: public health research needs. *Environmental Health*, 5(25). doi - 10.1186/1476-069X-5-25
- Luo, W. & Wang, F. (2003). Measures of spatial accessibility to health care in a GIS environment: synthesis and a case study in the Chicago region. *Environment and Planning B: Planning and Design*, 30(6), 865-884. doi - doi:10.1068/b29120
- Luo, W., & Qi, Y. (2009). An enhanced two-step floating catchment area (E2SFCA) method for measuring spatial accessibility to primary care physicians. *Health & Place*, 15(4), 1100-1107. doi - 10.1016/j.healthplace.2009.06.002
- Luo, W., & Whippo, T. (2012). Variable catchment sizes for the two-step floating catchment area (2SFCA) method. *Health & Place*, 18(4), 789-795. doi - 10.1016/j.healthplace.2012.04.002
- Mao, L., & Nekorchuk, D. (2013). Measuring spatial accessibility to healthcare for populations with multiple transportation modes. *Health & Place*, 24, 115-122. doi - 10.1016/j.healthplace.2013.08.008
- Maroko, A. R., Maantay, J. A., Sohler, N. L., Grady, K. L., & Arno, P. S. (2009). The complexities of measuring access to parks and physical activity sites in New York City: a quantitative and qualitative approach. *International Journal of Health Geographics*, 8(1), 1. doi: 10.1186/1476-072X-8-34

- Matthews, C. E., Cohen, S. S., Fowke, J. H., Han, X., Xiao, Q., Buchowski, M. S., ... & Blot, W. J. (2014). Physical activity, sedentary behavior, and cause-specific mortality in black and white adults in the Southern Community Cohort Study. *American journal of epidemiology*, 180(4), 394-405. doi: 10.1093/aje/kwu142
- Mavoa, S., Witten, K., McCreanor, T., & O'Sullivan, D. (2012). GIS based destination accessibility via public transit and walking in Auckland, New Zealand. *Journal of Transport Geography*, 20(1), 15-22. doi - 10.1016/j.jtrangeo.2011.10.001
- McCormack, G. R., Rock, M., Toohey, A. M., & Hignell, D. (2010). Characteristics of urban parks associated with park use and physical activity: a review of qualitative research. *Health & Place*, 16(4), 712-726. doi - 10.1016/j.healthplace.2010.03.003
- MCDH, Mecklenburg County Department of Health. (2016) *2015 Mecklenburg County State of the Country Health Report, Mecklenburg County Department of Health, Health Statistics and Epidemiology*. Available at: charmack.org/mecklenburg/county/HealthDepartment/HealthStatistics.
- McGinnis, J. M., & Foege, W. H. (1993). Actual causes of death in the United States. *Jama*, 270(18), 2207-2212. doi:10.1001/jama.1993.03510180077038
- McGinnis, J. M., Williams-Russo, P., & Knickman, J. R. (2002). The case for more active policy attention to health promotion. *Health affairs*, 21(2), 78-93. doi: 10.1377/hlthaff.21.2.78
- McGrail, M. R., & Humphreys, J. S. (2009). Measuring spatial accessibility to primary care in rural areas: improving the effectiveness of the two-step floating catchment area method. *Applied Geography*, 29(4), 533-541. doi - 10.1016/j.apgeog.2008.12.003
- McShane C. on behalf of the Urban Institute at UNC Charlotte (June, 2014) *National ranking puts Charlotte near bottom for 'ParkScore'*. UNC Charlotte Urban Institute, Division of Academic Affairs. Available at: ui.uncc.edu/story/parkscore-charlotte-trails-park-land-accessiblity-0. Accessed January 5, 2016.
- Miyake, K. K., Maroko, A. R., Grady, K. L., Maantay, J. A., & Arno, P. S. (2010). Not Just A Walk in The Park: methodological improvements for determining environmental justice implications of park access in New York City for the promotion of physical activity. *Cities and the Environment*, 3(1), Article 8. pmcid - PMC3160641
- Moon, G., & Gillespie, R. (1995). *Society and health: an introduction to social science for health professionals*. Psychology Press. ISBN-13: 978-0415110228
- Moore, L. V., Diez Roux, A. V., Evenson, K. R., McGinn, A. P., & Brines, S. J. (2008). Availability of recreational resources in minority and low socioeconomic status areas. *American Journal of Preventive Medicine*, 34(1), 16-22. doi - 10.1016/j.amepre.2007.09.021

- Mozaffarian, D., Benjamin, E. J., Go, A. S., Arnett, D. K., Blaha, M. J., Cushman, M., ... & Howard, V. J. on behalf of the American Heart Association Statistics Committee and Stroke Statistics Subcommittee (2016a). Heart disease and stroke statistics—2016 update: Summary. *Circulation*, 133(4), e39-e45. doi: 10.1161/CIR.0000000000000350
- Mozaffarian, D., Benjamin, E. J., Go, A. S., Arnett, D. K., Blaha, M. J., Cushman, M., ... & Howard, V. J. on behalf of the American Heart Association Statistics Committee and Stroke Statistics Subcommittee (2016b). Heart disease and stroke statistics—2016 update: Physical Inactivity. *Circulation*, 133(4), e78-e88. doi: 10.1161/CIR.0000000000000350
- Mueller, N., Rojas-Rueda, D., Basagaña, X., Cirach, M., Cole-Hunter, T., Dadvand, P., ... & Tonne, C. (2016). Urban and Transport Planning Related Exposures and Mortality: A Health Impact Assessment for Cities. *Environmental Health Perspectives*. doi: 10.1289/EHP220
- Nicholls, S. (2001). Measuring the accessibility and equity of public parks: a case study using GIS. *Managing Leisure*, 6(4), 201-219. doi - 10.1080/13606710110084651
- NM Incite (2012) *State of the Media: The social media report*. Joint venture between Nielsen and McKinsey.
- Openshaw, S. (1983). *The modifiable areal unit problem*. Norwick: Geo Books. ISBN 0860941345.
- Páez, A., Scott, D. M., & Morency, C. (2012). Measuring accessibility: positive and normative implementations of various accessibility indicators. *Journal of Transport Geography*, 25, 141-153. doi - 10.1016/j.jtrangeo.2012.03.016
- Parks, S. E., Housemann, R. A., & Brownson, R. C. (2003). Differential correlates of physical activity in urban and rural adults of various socioeconomic backgrounds in the United States. *Journal of Epidemiology and Community Health*, 57, 29-35. doi - 10.1136/jech.57.1.29
- Paul, M. J., & Dredze, M. (2011, July). You are what you Tweet: Analyzing Twitter for public health. *In ICWSM* (pp. 265-272).
- Physical Activity Council (2016). *2016 Participation Report*. Available at: physicalactivitycouncil.com. Accessed May 20, 2016.
- Reyes, M., Páez, A., & Morency, C. (2014). Walking accessibility to urban parks by children: A case study of Montreal. *Landscape and Urban Planning*, 125, 38-47. doi - 10.1016/j.landurbplan.2014.02.002
- Rosenberger, R. S., Sneh, Y., Phipps, T. T., & Gurvitch, R. (2005). A spatial analysis of linkages between health care expenditures, physical inactivity, obesity and recreation supply. *Journal of Leisure Research*, 37(2), 216.

- Salathé, M., & Khandelwal, S. (2011). Assessing vaccination sentiments with online social media: implications for infectious disease dynamics and control. *PLoS Computer Biology*, 7(10), e1002199.
- Schwanen, T., & Páez, A. (2010). The mobility of older people – an introduction. *Journal of Transport Geography* 18(5), 591-595. doi - 10.1016/j.jtrangeo.2010.06.001
- Sister, C., Wolch, J., & Wilson, J. (2010). Got green? Addressing environmental justice in park provision. *GeoJournal*, 75(3), 229-248. doi - 10.1007/s10708-009-9303-8
- Talen, E. (2003). Neighborhoods as service providers: a methodology for evaluating pedestrian access. *Environment and Planning B: Planning and Design*, 30(2), 181-200. doi - 10.1068/b12977
- TPL, Trust for Public Land (2010) *The Economic Benefits of the Park and Recreation System of Mecklenburg County, North Carolina*. Available at: tpl.org/charlottesmecklenburg-county-park-value-report. Accessed March 15, 2014.
- TPL, Trust for Public Land. (2016) *ParkScore Index*. Available at: parkscore.tpl.org. Accessed May 16, 2016.
- Vaughan, K. B., Kaczynski, A. T., Stanis, S. A. W., Besenyi, G. M., Bergstrom, R., & Heinrich, K. M. (2013). Exploring the distribution of park availability, features, and quality across Kansas City, Missouri by income and race/ethnicity: an environmental justice investigation. *Annals of Behavioral Medicine*, 45(1), 28-38. doi - 10.1007/s12160-012-9425-y
- Walls, M. (2009). *Parks and Recreation in the United States: Local Park Systems*. Resources for the Future: Backgrounder (June Edition). Washington, DC.
- Wilson, T., Wiebe, J., & Hoffmann, P. (2005, October). Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing* (pp. 347-354). Association for Computational Linguistics.
- Wolch, J. R., Byrne, J., & Newell, J. P. (2014). Urban green space, public health, and environmental justice: The challenge of making cities ‘just green enough’. *Landscape and Urban Planning*, 125, 234-244. doi - 10.1016/j.landurbplan.2014.01.017
- Wong, C. A., Sap, M., Schwartz, A., Town, R., Baker, T., Ungar, L., & Merchant, R. M. (2015). Twitter Sentiment Predicts Affordable Care Act Marketplace Enrollment. *Journal of medical Internet research*, 17(2).
- Xiang, W. -N., & Clarke, K. C. (2003). The Use of Scenarios in Land Use Planning. *Environment and Planning B: Planning and Design*, 30(6), 885-909. doi - 10.1068/b2945

Zhang, X., Lu, H., & Holt, J. B. (2011). Modeling spatial accessibility to parks: a national study. *International Journal of Health Geographics*, 10(31). doi - 10.1186/1476-072X-10-31

APPENDIX A: IRB APPROVALS AND AMENDMENTS

1. Initial Approval



UNC CHARLOTTE

Research and Economic Development

Office of Research Compliance

9201 University City Blvd, Charlotte, NC 28223-0001

t/ 704.687.1876 f/ 704.687.0980 <http://research.uncc.edu/compliance-ethics>**Institutional Review Board (IRB) for Research with Human Subjects***Approval of Exemption***Protocol #** 15-05-07**Title:** Park Utilization in Mecklenburg County for Physical Activity**Date:** 5/13/2015**Responsible Faculty** Dr. Eric Delmelle Geography & Earth Sciences**Investigator** Ms. Coline Dony Geography & Earth Sciences

The Institutional Review Board (IRB) certifies that the protocol listed above is exempt under category 2 (45 CFR 46.101).

Research involving the use of educational tests (cognitive, diagnostic, aptitude, achievement), survey procedures, interview procedures or observation of public behavior, unless:

- a) information obtained is recorded in such a manner that human subjects can be identified, directly or through identifiers linked to the subjects; and
- b) any disclosure of the human subjects' responses outside the research could reasonably place the subjects at risk of criminal or civil liability or be damaging to the subjects' financial standing, employability, or reputation.

This approval will expire one year from the date of this letter. In order to continue conducting research under this protocol after one year, the "Annual Protocol Renewal Form" must be submitted to the IRB. Please note that it is the investigator's responsibility to promptly inform the committee of any changes in the proposed research, as well as any unanticipated problems that may arise involving risks to subjects. Amendment and Event Reporting forms are available on our web site: <http://research.uncc.edu/compliance-ethics/human-subjects/amending-your-protocol> or <http://research.uncc.edu/compliance-ethics/human-subjects/reporting-adverse-events>

Dr. M. Lyn Exum, IRB Chair

Date

2. IRB Amendment Approval (September 2015)



UNC CHARLOTTE

Research and Economic Development

Office of Research Compliance

9201 University City Blvd, Charlotte, NC 28223-0001

t/ 704.687.1876 f/ 704.687.0980 <http://research.uncc.edu/compliance-ethics>

Institutional Review Board (IRB) for Research with Human Subjects

Approval of Exemption

Protocol #	15-05-07		
Title:	Park Utilization in Mecklenburg County for Physical Activity		
Date:	5/13/2015		
Responsible Faculty	Dr. Eric	Delmelle	Geography & Earth Sciences
Investigator	Ms. Coline	Dony	Geography & Earth Sciences

The Institutional Review Board (IRB) certifies that the protocol listed above is exempt under category 2 (45 CFR 46.101).

Research involving the use of educational tests (cognitive, diagnostic, aptitude, achievement), survey procedures, interview procedures or observation of public behavior, unless:

- a) information obtained is recorded in such a manner that human subjects can be identified, directly or through identifiers linked to the subjects; and
- b) any disclosure of the human subjects' responses outside the research could reasonably place the subjects at risk of criminal or civil liability or be damaging to the subjects' financial standing, employability, or reputation.

This approval will expire one year from the date of this letter. In order to continue conducting research under this protocol after one year, the "Annual Protocol Renewal Form" must be submitted to the IRB. Please note that it is the investigator's responsibility to promptly inform the committee of any changes in the proposed research, as well as any unanticipated problems that may arise involving risks to subjects. Amendment and Event Reporting forms are available on our web site: <http://research.uncc.edu/compliance-ethics/human-subjects/amending-your-protocol> or <http://research.uncc.edu/compliance-ethics/human-subjects/reporting-adverse-events>

Dr. M. Lyn Exum, IRB Chair

5/19/15

Date

The UNIVERSITY of NORTH CAROLINA at CHARLOTTE

An Equal Opportunity/Affirmative Action Employer

3. IRB Amendment Approval (October 2015)



UNC CHARLOTTE

Research and Economic Development

Office of Research Compliance

9201 University City Blvd, Charlotte, NC 28223-0001

t/ 704.687.1876 f/ 704.687.0980 <http://research.uncc.edu/compliance-ethics>

**Institutional Review Board (IRB) for Research with Human Subjects
University of North Carolina at Charlotte**

Approval of Amendment

Protocol # 15-05-07

Title: Park Utilization in Mecklenburg County for Physical Activity

Date: 10/21/2015

Investigator: Ms. Coline Dony Geography & Earth Sciences
Co-investigator: Dr. Eric Delmelle Geography & Earth Sciences
Co-investigator: Ms. Ashley Stevens Alzheimer's Association of Western North Carolina
Research Assistants: Collin Wood, Kelly O'Connor, Joshua Leslie, William Sun, Daniel Yonto, Kaitlin Colandrea, Michael Desjardins

The Institutional Review Board (IRB) has approved the amendment of the protocol listed above for Research with Human Subjects per 45 CFR 46.111.

Please note that it is the investigator's responsibility to promptly inform the committee of any changes in the proposed research, as well as any unanticipated problems that may arise involving risks to subjects.

Amendment Details: Update to research team. Add Research Assistants: Collin Wood (training completion 10/16/2015); Kelly O'Connor (training completion 10/19/2015); Joshua Leslie (training completion 10/20/2015); William Sun (training completion 10/16/2015); Daniel Yonto (training completion 3/21/2015); Kaitlin Colandrea (training completion 10/19/2015); and Michael Desjardins (training completion 9/22/2015).

Catherine Runden

Catherine Runden
Office of Research Compliance

10/21/15
Date



4. IRB Renewal Approval



UNC CHARLOTTE

Research and Economic Development

Office of Research Compliance

9201 University City Blvd, Charlotte, NC 28223-0001

t/ 704.687.1876 f/ 704.687.0980 <http://research.uncc.edu/compliance-ethics>

Institutional Review Board (IRB) for Research with Human Subjects
University of North Carolina at Charlotte

Continuing Approval of Exemption

~ for Year 2 of Study ~

Protocol #	15-05-07		
Protocol Type:	Exempt 2		
Title:	Park Utilization in Mecklenburg County for Physical Activity		
Date:	5/12/2016		
Investigator:	Ms. Coline	Dony	Geography & Earth Sciences
Responsible Faculty:	Dr. Eric	Delmelle	Geography & Earth Sciences

The Institutional Review Board (IRB) certifies that the protocol listed above continues to be exempt under category 2 (CFR 46.101.b.2).

Research involving the use of educational tests (cognitive, diagnostic, aptitude, achievement), survey procedures, interview procedures or observation of public behavior, unless:

- a) information obtained is recorded in such a manner that human subjects can be identified, directly or through identifiers linked to the subjects; and
- b) any disclosure of the human subjects' responses outside the research could reasonably place the subjects at risk of criminal or civil liability or be damaging to the subjects' financial standing, employability, or reputation.

The continuing approval of this protocol will expire one year from the date of this letter. In order to continue conducting research under this protocol after one year, the "Annual Renewal" form must be submitted to the IRB. The renewal form can be obtained from the Office of Research Compliance web page (<http://research.uncc.edu/compliance-ethics/human-subjects>).

Please note that it is the investigator's responsibility to promptly inform the committee of any changes in the proposed research, and of any adverse events or unanticipated risks to subjects or others. Amendment and Event Reporting forms are available on our web page.


 Dr. M. Lyn Exum, IRB Chair


 Date



APPENDIX B: SURVEY INSTRUMENT

1. Questionnaire Page 1

YOUR RESPONSES TO THIS SURVEY WILL REMAIN ANONYMOUS AND CONFIDENTIAL!

Questions that follow are about your visit purpose and travel experience today :

1. How did you travel to this park today? (Check one)
 I carpooled by car by bicycle I walked by light rail by bus by motorcycle
2. Where did you travel from to come to this park today? (Check one)
 home work school friend(s) family Other: _____
3. How much time did it take to travel to this park today? _____ hours _____ minutes
4. Is this the first time you visit this park?
 Yes No → a. When was the first time you came to this park? (Check one)
 less than 1 year ago 1 to 3 years ago more than 3 years ago
 b. In the past year, how often did you come to this park? (Check one)
 Daily A few times a week Weekly Monthly Annually
5. Did you come to exercise or engage in physical activity today? Yes No
6. Is this the closest park to your house? Yes No Not sure
7. The following questions are about the park that is the closest to your house:
 - a. Does it ask for an entrance fee? Yes No Not sure
 - b. Is it accessible for anyone (e.g. Public Park)? Yes No Not sure
 - c. Is it easy to find space to park your car? Yes No Not sure
 - d. Would you say it's easy to walk to that park? Yes No Not sure
 - e. Would you say it is safe? Yes No Not sure
 - f. Is the park well-kept / in good shape? Yes No Not sure
 - g. How often did you go there in the past year? Regularly Sometimes Never
 - h. Did you ever go there to exercise or engage in physical activity? Yes No
 - i. How many different amenities do you think it has? None 1-2 3-4 5 or more
 - j. How many minutes would it take to walk there? 1-5 6-10 11-20 21-30 > 31 min.
8. In the past year, did you go to any other parks within Mecklenburg County?
 No Yes → a. How many did you visit in the past year? 1-2 3-5 6 or more
9. Check all transport options you have used to travel to these parks in the past year? (Check all that apply)
 Car Carpooling Bicycle Walking Light rail Bus Motorcycle
10. In the past year, how often did you exercise or engage in physical activity for 30 minutes or more?
 4 times per week or more 2-3 times per week Weekly Monthly Rarely Never
11. In the past year, how often did you exercise or engage in physical activity at parks?
 4 times per week or more 2-3 times per week Weekly Monthly Rarely Never
12. How would you rate your access to parks? Very poor 1 2 3 4 5 Very good
13. When you go to ANY park, : Not important Very important
 - a. How important are park amenities? 1 2 3 4 5
 - b. How important is the park design and aesthetics? 1 2 3 4 5
 - c. How important is the size of the park? 1 2 3 4 5
 - d. How important is the safety of the park? 1 2 3 4 5

2. Questionnaire Page 2

YOUR RESPONSES TO THIS SURVEY WILL REMAIN ANONYMOUS AND CONFIDENTIAL!

Questions that follow are more personal

14. In the past year, did you ...				
a. ... own or have access to a car?	Yes, I own a car	Yes, I have access to a car	No	
b. ... have a dog?	Yes	No		
c. ... have a gym membership?	Yes	No		
15. Are you currently...				
a. ... retired?	Yes	No		
b. ... a student?	Yes	No		
c. ... a homeowner?	Yes	No		
16. Do you currently have a paid job?				
No	Yes →	a. Do you work more than 20h per week?	Yes	No
		b. Does your job require you to engage in physical activity <u>regularly</u> ?	Yes	No
17. Do you have children?				
No	Yes →	a. How many are under age 12?	None	1-2
		b. How many currently live at your house?	None	1-2
		c. Are you currently a stay-at-home parent?	Yes	No
18. What is the highest level of education you have completed? (<i>Check one</i>)				
Some high school	Technical or trade school certificate	Some college		
High school completed	Higher education degree	University degree		
19. In 2014, what was your annual household income, after tax? (<i>Check one</i>)				
less than \$25,000	\$50,000 to \$74,999	\$100,000 to \$149,999		
\$25,000 to \$49,999	\$75,000 to \$99,999	\$150,000 or more		
20. What is your ethnicity / race? (<i>Check all that apply</i>)				
Black	Native American	White	Hispanic	Other: _____
African American	Indian	Asian	Latino	_____
21. Looking back at the past year, how would <u>you</u> rate your health? (<i>Check one</i>)				
Very good	Good	Average	Poor	Very poor
22. What is ...				
a. ... your age?	_____			
b. ... your height?	_____ feet _____ inches	_____ cm		
c. ... your weight?	_____ pounds	_____ kilograms		
d. ... your gender?	Male	Female	Don't want to specify	
23. Do you currently live in Mecklenburg County?				
No	Yes →	a. How many years have you lived here?	< 1	1-2
			3-5	> 5
24. What is your ZIP Code? _____				
25. What is the street name of your home? _____				
WRITE IN ALL CAPITAL LETTERS				
26. What is the nearest cross street to your home? _____				
WRITE IN ALL CAPITAL LETTERS				

APPENDIX C: INVITATION TO FILL-OUT WEB-BASED SURVEY



Scan code with your phone
to Participate!



Or go to:
<http://bit.do/parkaccess>

We are interested to better understand how YOU are accessing parks and using its amenities. This survey takes 5 minutes and would help the Charlotte Community greatly. We truly appreciate your time!

Anonymity and Confidentiality

Personal questions such as your household income, ethnicity, height and weight will be asked as well. To safeguard your identity, the questionnaire is anonymous – we will not ask for your name, date of birth or *full* address.

Contact Details: If you have any questions or concerns regarding this survey, please feel free to contact:



Coline C. Dony, M.A, M.Sc.
cdony@uncc.edu

Dr. Eric M. Delmelle
Eric.Delmelle@uncc.edu

APPROVED BY THE INSTITUTIONAL REVIEW BOARD OF THE UNIVERSITY OF NORTH CAROLINA AT CHARLOTTE ON 13 May 2015 FOR 1 YEAR. PROTOCOL NUMBER 15-05-07



Among 75 major U.S. cities,
Charlotte is the lowest scoring city on *ParkScore*®



We study the inequality of access to parks !

If you ...

- ... are interested in this research topic
- ... want to volunteer
- ... like to be outdoors
- ... want experience in qualitative data

Join us! - undergrad and graduate students

We need your help collecting surveys in Charlotte Parks!
(in September and early October 2015)

Contact me, Coline Dony at cdony@uncc.edu

APPENDIX E: CHARACTERISTICS OF PARKS INCLUDED IN SURVEY

Name	Class	Acres	Playgrounds (2-5yrs.)	Playgrounds (5-12yrs.)	Outdoor Pools / Spraygrounds	Pavilions - Large/Indoor Shelters	Picnic Pavilions - Medium Small	Trails - All Surfaces (miles)	Baseball Fields	Softball Fields	Multipurpose Fields	Basketball Courts	Tennis Courts	Volleyball Courts	Dog Parks	Skate Parks	Aquatic Centers/Indoor Pools	Recreation/Fitness Centers	Region	total_am
Reedy Creek Nature Preserve	Nature Preserve	737.35	1	-	-	-	1	10	-	-	-	-	-	-	-	-	-	-	NORTH	2
McDowell Nature Preserve	Nature Preserve	1,114.84	1	1	-	1	1	7	-	-	-	-	-	-	-	-	-	-	SOUTH	4
Latta Plantation Nature Preserve	Nature Preserve	1,478.27	-	-	-	-	4	19.2	-	-	-	-	-	1	-	-	-	-	NORTH	5
Bradford Park	Regional Park	264	1	-	-	-	2	-	5	-	3	-	-	-	-	-	-	-	NORTH	11
Elon Park	Regional Park	118.47	1	-	-	-	-	-	-	-	4	-	-	-	-	-	-	13,067	SOUTH	5
Freedom Park	Regional Park	104.89	1	2	-	1	5	1.94	4	-	4	1	1	2	-	-	-	-	CENTRAL	32
McAlpine Creek Park	Regional Park	109.39	-	-	-	-	-	18	-	-	5	-	-	-	1	-	-	-	SOUTH	6
Shuffletown Park	Community Park	54.2	1	1	-	-	1	-	2	-	-	-	-	2	1	-	-	-	NORTH	8
Sugaw Creek Park	Community Park	72.79	-	1	-	-	3	0.18	-	2	1	2	6	1	-	-	-	13,756	CENTRAL	16
Independence Park	Community Park	21.08	1	1	-	-	4	1.07	1	-	1	2	2	1	-	-	-	9,688	CENTRAL	13
Huntingtowne Farms	Community Park	22.74	1	1	-	-	1	-	-	-	2	1	2	-	-	-	-	-	SOUTH	8
Ramsey Creek Park	Community Park	43.02	-	1	-	-	3	-	-	-	-	-	-	1	1	-	-	-	NORTH	6
Enderly Park	Neighborhood Park	8.48	1	1	-	-	1	-	-	1	-	1	2	-	-	-	-	-	CENTRAL	7
Theresea Clark Elders Park	Neighborhood Park	15.75	1	1	-	-	1	-	-	-	1	1	-	-	-	-	-	-	NORTH	5
Springfield Park	Neighborhood Park	11.52	-	-	-	-	-	-	-	-	-	2	-	-	-	-	-	-	SOUTH	2
McKee Park	Neighborhood Park	19.63	-	-	-	-	-	-	-	-	3	-	-	-	-	-	-	-	SOUTH	3
Cherry Park	Neighborhood Park	1.89	1	1	-	-	-	0.13	-	1	-	1	-	-	-	-	-	-	CENTRAL	4
Romare Beardon Park	Urban Park	5.09	-	-	1	-	-	-	-	-	-	-	-	-	-	-	-	-	CENTRAL	1

APPENDIX F: SURVEY DATA – CLEANING PROCESS (STATA)

clean_survey_data.do - Printed on 09-Aug-16 21:17:05

```

1  cd "C:\Dissertation_Materials\Data\core_datasets\park_survey"
2
3  import delimited complete_survey_data_clean.csv
4  foreach x of var * {
5      rename `x' `x'_orig
6  }
7
8  //
9  gen survey_id = _n
10 //
11 tabulate s_type_orig
12 encode s_type_orig , generate(s_type)
13 //
14 gen double time_digitized = clock(time_dig_orig , "MD20Yhm")
15 format time_digitized %tc
16 //
17 tabulate park_orig
18 encode park_orig , generate(park_name)
19 //
20 tabulate park_name
21 gen unique_id = ""
22 replace unique_id = "morg" if park_orig == "Cherry Park"
23 replace unique_id = "h521" if park_orig == "Elon Park"
24 replace unique_id = "endp" if park_orig == "Enderly Park"
25 replace unique_id = "free" if park_orig == "Freedom Park"
26 replace unique_id = "hunt" if park_orig == "Huntingtown Farms"
27 replace unique_id = "indp" if park_orig == "Independence Park"
28 replace unique_id = "latn" if park_orig == "Latta Plantation Nature Preserve"
29 replace unique_id = "mccp" if park_orig == "McAlpine Creek Regional Park"
30 replace unique_id = "mcdl" if park_orig == "McDowell Nature Center and Preserve"
31 replace unique_id = "mcke" if park_orig == "McKee Road Park"
32 replace unique_id = "rams" if park_orig == "Ramsey Creek Park"
33 replace unique_id = "renp" if park_orig == "Reedy Creek Nature Preserve"
34 replace unique_id = "cdhp" if park_orig == "Robert Caldwell Bradford Park"
35 replace unique_id = "wepk" if park_orig == "Romare Bearden Park"
36 replace unique_id = "shld" if park_orig == "Sheffield Park"
37 replace unique_id = "clwd" if park_orig == "Shuffletown Park"
38 replace unique_id = "sugp" if park_orig == "Sugaw Creek Park"
39 replace unique_id = "rock" if park_orig == "Thereasea Clark Elder Park"
40 //
41 tabulate park_type_orig
42 gen park_type = .
43 replace park_type = 1 if park_type_orig == "Urban Park"
44 replace park_type = 2 if park_type_orig == "Neighborhood Park"
45 replace park_type = 3 if park_type_orig == "Community Park"
46 replace park_type = 4 if park_type_orig == "Regional Park"
47 replace park_type = 5 if park_type_orig == "Nature Preserve"
48 label define park_type 1 "Urban Park" 2 "Neighborhood Park" 3 "Community Park" 4 "Regional
Park" 5 "Nature Preserve"
49 label values park_type park_type
50 //
51 recode park_type (2 3 = 2) (4 5 = 1) (1 = 3), gen(park_type_simple)
52 label define park_type_simple 2 "Neighborhood or Community Park" 1 "Regional Park or Nature
Preserve" 3 "Urban Park"
53 label values park_type_simple park_type_simple
54 //
55 gen double time_surveyed = clock(time_sur_orig , "MD20Yhm")
56 format time_surveyed %tc
57 //
58 generate day_of_week_temp = string(time_surveyed, "%tcDAYNAME")
59 gen day_of_week = .
60 replace day_of_week = 1 if day_of_week_temp == "Monday"
61 replace day_of_week = 2 if day_of_week_temp == "Tuesday"
62 replace day_of_week = 3 if day_of_week_temp == "Wednesday"
63 replace day_of_week = 4 if day_of_week_temp == "Thursday"
64 replace day_of_week = 5 if day_of_week_temp == "Friday"
65 replace day_of_week = 6 if day_of_week_temp == "Saturday"
66 replace day_of_week = 7 if day_of_week_temp == "Sunday"
67 label define day_of_week 1 "Monday" 2 "Tuesday" 3 "Wednesday" 4 "Thursday" 5 "Friday" 6
"Saturday" 7 "Sunday"
68 label values day_of_week day_of_week

```


clean_survey_data.do - Printed on 09-Aug-16 21:17:05

```

69 //
70 recode day_of_week (1 2 3 4 5 = 1 Weekday) (6 7 = 0 Week-end), gen(weekday)
71 //
72 tabulate transport_orig
73 gen transport = .
74 replace transport = 1 if transport_orig == "by car"
75 replace transport = 2 if transport_orig == "I carpooled"
76 replace transport = 3 if transport_orig == "by bicycle"
77 replace transport = 4 if transport_orig == "I walked"
78 replace transport = 5 if transport_orig == "by bus"
79 replace transport = 6 if transport_orig == "by motorcycle"
80 replace transport = 7 if transport_orig == "I carpooled, by car, by bicycle, I walked"
81 label define transport 1 "Drive" 2 "Drive - passenger" 3 "Bike" 4 "Walk" 5 "Transit" 6
   "Motorbike" 7 "Multiple modes"
82 label values transport transport
83 //
84 recode transport (1 2 6 = 1 Passive) (3 4 = 2 Active) (5 = 3 Transit) (7 = 4 Mixed), gen(
   transport_type)
85 //
86 tabulate origin_orig
87 gen origin = 6
88 replace origin = 1 if origin_orig == "home"
89 replace origin = 2 if origin_orig == "work"
90 replace origin = 3 if origin_orig == "school"
91 replace origin = 4 if origin_orig == "friend(s)"
92 replace origin = 5 if origin_orig == "family"
93 replace origin = . if origin_orig == "No Answer"
94 label define origin 1 "Home" 2 "Work" 3 "School" 4 "Friend(s)" 5 "Family" 6 "Other"
95 label values origin origin
96 //
97 recode origin (1 = 1 Home) (2 3 = 2 Work-School) (4 5 6 = 3 Other), gen(origin_type)
98 //
99 tabulate travel_time_orig
100 gen travel_time = travel_time_orig
101 destring travel_time, replace force
102 //
103 gen travel_time_ordinal = 5
104 replace travel_time_ordinal = 4 if travel_time < 60
105 replace travel_time_ordinal = 3 if travel_time < 40
106 replace travel_time_ordinal = 2 if travel_time < 20
107 replace travel_time_ordinal = 1 if travel_time < 10
108 replace travel_time_ordinal = . if travel_time == .
109 label define travel_time_ordinal 1 "less than 10 min." 2 "10 - 19 min." 3 "20 - 39 min." 4
   "40 - 59 min." 5 "1 hour or more"
110 label values travel_time_ordinal travel_time_ordinal
111 //
112 gen travel_time_ordinal_simple1 = 1
113 replace travel_time_ordinal_simple1 = 0 if travel_time > 11
114 replace travel_time_ordinal_simple1 = . if travel_time == .
115 label define travel_time_ordinal_simple1 1 "10 min. or less" 0 "more than 10 min."
116 label values travel_time_ordinal_simple1 travel_time_ordinal_simple1
117 //
118 gen travel_time_ordinal_simple2 = 1
119 replace travel_time_ordinal_simple2 = 0 if travel_time < 30
120 replace travel_time_ordinal_simple2 = . if travel_time == .
121 label define travel_time_ordinal_simple2 1 "more than 30 min." 0 "less than 30 min."
122 label values travel_time_ordinal_simple2 travel_time_ordinal_simple2
123 //
124 tabulate first_visit_orig
125 gen first_visit = .
126 replace first_visit = 0 if first_visit_orig == "No"
127 replace first_visit = 1 if first_visit_orig == "Yes"
128 label define yesno 0 "No" 1 "Yes"
129 label values first_visit yesno
130 //
131 tabulate time_first_visit_orig
132 gen time_first_visit = .
133 replace time_first_visit = 1 if time_first_visit_orig == "less than 1 year ago"
134 replace time_first_visit = 2 if time_first_visit_orig == "1 to 3 years ago"
135 replace time_first_visit = 3 if time_first_visit_orig == "more than 3 years ago"
136 replace time_first_visit = .n if time_first_visit_orig == "Not Applicable"

```

clean_survey_data.do - Printed on 09-Aug-16 21:17:05

```

137 label define time_first_visit 1 "less than 1 year ago" 2 "1 to 3 years ago" 3 "more than 3
    years ago"
138 label values time_first_visit time_first_visit
139 //
140 tabulate visit_freq_orig
141 gen visit_frequency = .
142 replace visit_frequency = 1 if visit_freq_orig == "Daily"
143 replace visit_frequency = 2 if visit_freq_orig == "A few times a week"
144 replace visit_frequency = 3 if visit_freq_orig == "Weekly"
145 replace visit_frequency = 4 if visit_freq_orig == "Monthly"
146 replace visit_frequency = 5 if visit_freq_orig == "Annually"
147 replace visit_frequency = .n if visit_freq_orig == "Not Applicable"
148 label define visit_frequency 1 "Daily" 2 "A few times a week" 3 "Weekly" 4 "Monthly" 5
    "Annually"
149 label values visit_frequency visit_frequency
150 //
151 tabulate pa_today_orig
152 gen pa_today = .
153 replace pa_today = 0 if pa_today_orig == "No"
154 replace pa_today = 1 if pa_today_orig == "Yes"
155 label values pa_today yesno
156 //
157 tabulate closest_park_orig
158 gen closest_park = .
159 replace closest_park = 1 if closest_park_orig == "Yes"
160 replace closest_park = 2 if closest_park_orig == "Not Sure"
161 replace closest_park = 3 if closest_park_orig == "No"
162 replace closest_park = .n if closest_park_orig == "Not Applicable"
163 label define yesno_ns 1 "Yes" 2 "Not Sure" 3 "No"
164 label values closest_park yesno_ns
165 //
166 tabulate cp_entrance_orig
167 gen cp_entrance = .
168 replace cp_entrance = 1 if cp_entrance_orig == "Yes"
169 replace cp_entrance = 2 if cp_entrance_orig == "Not Sure"
170 replace cp_entrance = 3 if cp_entrance_orig == "No"
171 replace cp_entrance = .n if cp_entrance_orig == "Not Applicable"
172 label values cp_entrance yesno_ns
173 //
174 tabulate cp_public_orig
175 gen cp_public = .
176 replace cp_public = 1 if cp_public_orig == "Yes"
177 replace cp_public = 2 if cp_public_orig == "Not Sure"
178 replace cp_public = 3 if cp_public_orig == "No"
179 replace cp_public = .n if cp_public_orig == "Not Applicable"
180 label values cp_public yesno_ns
181 //
182 tabulate cp_parking_orig
183 gen cp_parking = .
184 replace cp_parking = 1 if cp_parking_orig == "Yes"
185 replace cp_parking = 2 if cp_parking_orig == "Not Sure"
186 replace cp_parking = 3 if cp_parking_orig == "No"
187 replace cp_parking = .n if cp_parking_orig == "Not Applicable"
188 label values cp_parking yesno_ns
189 //
190 tabulate cp_walkable_orig
191 gen cp_walkable = .
192 replace cp_walkable = 1 if cp_walkable_orig == "Yes"
193 replace cp_walkable = 2 if cp_walkable_orig == "Not Sure"
194 replace cp_walkable = 3 if cp_walkable_orig == "No"
195 replace cp_walkable = .n if cp_walkable_orig == "Not Applicable"
196 label values cp_walkable yesno_ns
197 //
198 tabulate cp_safe_orig
199 gen cp_safe = .
200 replace cp_safe = 1 if cp_safe_orig == "Yes"
201 replace cp_safe = 2 if cp_safe_orig == "Not Sure"
202 replace cp_safe = 3 if cp_safe_orig == "No"
203 replace cp_safe = .n if cp_safe_orig == "Not Applicable"
204 label values cp_safe yesno_ns
205 //

```


clean_survey_data.do - Printed on 09-Aug-16 21:17:05

```

206 tabulate cp_maintenance_orig
207 gen cp_maintenance = .
208 replace cp_maintenance = 1 if cp_maintenance_orig == "Yes"
209 replace cp_maintenance = 2 if cp_maintenance_orig == "Not Sure"
210 replace cp_maintenance = 3 if cp_maintenance_orig == "No"
211 replace cp_maintenance = .n if cp_maintenance_orig == "Not Applicable"
212 label values cp_maintenance yesno_ns
213 //
214 tabulate cp_visit_freq_orig
215 gen cp_visit_frequency = .
216 replace cp_visit_frequency = 1 if cp_visit_freq_orig == "Regularly"
217 replace cp_visit_frequency = 2 if cp_visit_freq_orig == "Sometimes"
218 replace cp_visit_frequency = 3 if cp_visit_freq_orig == "Never"
219 label define cp_visit_frequency 1 "Regularly" 2 "Sometimes" 3 "Never"
220 label values cp_visit_frequency cp_visit_frequency
221 //
222 tabulate cp_pa_orig
223 gen cp_pa = .
224 replace cp_pa = 0 if cp_pa_orig == "No"
225 replace cp_pa = 1 if cp_pa_orig == "Yes"
226 label values cp_pa yesno
227 //
228 tabulate cp_amenities_orig
229 gen cp_amenities = .
230 replace cp_amenities = 1 if cp_amenities_orig == "2-Jan"
231 replace cp_amenities = 2 if cp_amenities_orig == "4-Mar"
232 replace cp_amenities = 3 if cp_amenities_orig == "5 or more"
233 replace cp_amenities = 4 if cp_amenities_orig == "None"
234 label define cp_amenities 1 "1 - 2" 2 "3 - 4" 3 "5 or more" 4 "None"
235 label values cp_amenities cp_amenities
236 //
237 tabulate cp_walk_time_orig
238 gen cp_walk_time = .
239 replace cp_walk_time = 1 if cp_walk_time_orig == "1-5 minutes"
240 replace cp_walk_time = 2 if cp_walk_time_orig == "6-10 minutes"
241 replace cp_walk_time = 3 if cp_walk_time_orig == "11-20 minutes"
242 replace cp_walk_time = 4 if cp_walk_time_orig == "21-30 minutes"
243 replace cp_walk_time = 5 if cp_walk_time_orig == "31 minutes or more"
244 replace cp_walk_time = .d if cp_walk_time_orig == "I don't know"
245 label define cp_walk_time 1 "1-5 minutes" 2 "6-10 minutes" 3 "11-20 minutes" 4 "21-30
minutes" 5 "31 minutes or more"
246 label values cp_walk_time cp_walk_time
247 //
248 recode cp_walk_time (1 2 = 1) (3 4 5 = 0), gen(cp_walk_time_simple)
249 label define cp_walk_time_simple 1 "10 minutes or less" 0 "more than 10 minutes"
250 label values cp_walk_time_simple cp_walk_time_simple
251 //
252 tabulate cp_other_parks_orig
253 gen other_parks = .
254 replace other_parks = 0 if cp_other_parks_orig == "No"
255 replace other_parks = 1 if cp_other_parks_orig == "Yes"
256 label values other_parks yesno
257 //
258 tabulate cp_number_op_orig
259 gen number_other_parks = .
260 replace number_other_parks = 1 if cp_number_op_orig == "2-Jan"
261 replace number_other_parks = 2 if cp_number_op_orig == "5-Mar"
262 replace number_other_parks = 3 if cp_number_op_orig == "6 or more"
263 replace number_other_parks = .n if cp_number_op_orig == "Not Applicable"
264 label define number_other_parks 1 "1 - 2" 2 "3 - 5" 3 "6 or more"
265 label values number_other_parks number_other_parks
266 //
267 tabulate transport_used_orig
268 gen transport_used = subinstr(transport_used_orig,"Carpooling","Pooling",.)
269
270 gen transport_used_car = 0
271 replace transport_used_car = 1 if strpos(transport_used, "Car")
272 replace transport_used_car = . if transport_used == "No Answer"
273 label values transport_used_car yesno
274
275 gen transport_used_carpool = 0

```

clean_survey_data.do - Printed on 09-Aug-16 21:17:05

```

276 replace transport_used_carpool = 1 if strpos(transport_used, "Pooling")
277 replace transport_used_carpool = . if transport_used == "No Answer"
278 label values transport_used_carpool yesno
279
280 gen transport_used_bicycle = 0
281 replace transport_used_bicycle = 1 if strpos(transport_used, "Bicycle")
282 replace transport_used_bicycle = . if transport_used == "No Answer"
283 label values transport_used_bicycle yesno
284
285 gen transport_used_walking = 0
286 replace transport_used_walking = 1 if strpos(transport_used, "Walking")
287 replace transport_used_walking = . if transport_used == "No Answer"
288 label values transport_used_walking yesno
289
290 gen transport_used_transit = 0
291 replace transport_used_transit = 1 if strpos(transport_used, "Bus")
292 replace transport_used_transit = 1 if strpos(transport_used, "Light rail")
293 replace transport_used_transit = . if transport_used == "No Answer"
294 label values transport_used_transit yesno
295
296 gen transport_used_motorcycle = 0
297 replace transport_used_motorcycle = 1 if strpos(transport_used, "Motorcycle")
298 replace transport_used_motorcycle = . if transport_used == "No Answer"
299 label values transport_used_motorcycle yesno
300 //
301 tabulate pa_freq_30_orig
302 gen pa_freq_30 = .
303 replace pa_freq_30 = 1 if pa_freq_30_orig == "4 times per week or more"
304 replace pa_freq_30 = 2 if pa_freq_30_orig == "2-3 times per week"
305 replace pa_freq_30 = 3 if pa_freq_30_orig == "Weekly"
306 replace pa_freq_30 = 4 if pa_freq_30_orig == "Monthly"
307 replace pa_freq_30 = 5 if pa_freq_30_orig == "Rarely"
308 replace pa_freq_30 = 6 if pa_freq_30_orig == "Never"
309 label define freq 1 "4 times per week or more" 2 "2-3 times per week" 3 "Weekly" 4 "Monthly"
310 5 "Rarely" 6 "Never"
311 label values pa_freq_30 freq
312 //
313 tabulate pa_freq_parks_orig
314 gen pa_freq_parks = .
315 replace pa_freq_parks = 1 if pa_freq_parks_orig == "4 times per week or more"
316 replace pa_freq_parks = 2 if pa_freq_parks_orig == "2-3 times per week"
317 replace pa_freq_parks = 3 if pa_freq_parks_orig == "Weekly"
318 replace pa_freq_parks = 4 if pa_freq_parks_orig == "Monthly"
319 replace pa_freq_parks = 5 if pa_freq_parks_orig == "Rarely"
320 replace pa_freq_parks = 6 if pa_freq_parks_orig == "Never"
321 label values pa_freq_parks freq
322 //
323 recode pa_freq_parks (1 2 3 = 1) (4 5 = 2) (6 = 3), gen(pa_freq_parks_simple)
324 label define pa_freq_parks_simple 1 "Regularly" 2 "Sometimes" 3 "Never"
325 label values pa_freq_parks_simple pa_freq_parks_simple
326 //
327 tabulate access_score_orig
328 gen access_score = .
329 replace access_score = 1 if access_score_orig == "1"
330 replace access_score = 2 if access_score_orig == "2"
331 replace access_score = 3 if access_score_orig == "3"
332 replace access_score = 4 if access_score_orig == "4"
333 replace access_score = 5 if access_score_orig == "5"
334 replace access_score = .n if access_score_orig == "Not Applicable"
335 //
336 recode access_score (1 2 3 = 1) (4 = 2) (5 = 3), gen(access_score_simple)
337 label define access_score_simple 1 "average or lower" 2 "good access" 3 "excellent access"
338 label values access_score_simple access_score_simple
339 //
340 tabulate amenity_score_orig
341 gen amenity_score = .
342 replace amenity_score = 1 if amenity_score_orig == "1"
343 replace amenity_score = 2 if amenity_score_orig == "2"
344 replace amenity_score = 3 if amenity_score_orig == "3"
345 replace amenity_score = 4 if amenity_score_orig == "4"
346 replace amenity_score = 5 if amenity_score_orig == "5"

```

clean_survey_data.do - Printed on 09-Aug-16 21:17:05

```

346 replace amenity_score = .n if amenity_score_orig == "Not Applicable"
347 //
348 recode amenity_score (1 2 3 = 1) (4 = 2) (5 = 3), gen(amenity_score_simple)
349 label define importance_simple 1 "neutral or not important" 2 "important" 3 "very important"
350 label values amenity_score_simple importance_simple
351 //
352 tabulate aesthetic_score_orig
353 gen aesthetic_score = .
354 replace aesthetic_score = 1 if aesthetic_score_orig == "1"
355 replace aesthetic_score = 2 if aesthetic_score_orig == "2"
356 replace aesthetic_score = 3 if aesthetic_score_orig == "3"
357 replace aesthetic_score = 4 if aesthetic_score_orig == "4"
358 replace aesthetic_score = 5 if aesthetic_score_orig == "5"
359 replace aesthetic_score = .n if aesthetic_score_orig == "Not Applicable"
360 //
361 recode aesthetic_score (1 2 3 = 1) (4 = 2) (5 = 3), gen(aesthetic_score_simple)
362 label values aesthetic_score_simple importance_simple
363 //
364 tabulate size_score_orig
365 gen size_score = .
366 replace size_score = 1 if size_score_orig == "1"
367 replace size_score = 2 if size_score_orig == "2"
368 replace size_score = 3 if size_score_orig == "3"
369 replace size_score = 4 if size_score_orig == "4"
370 replace size_score = 5 if size_score_orig == "5"
371 replace size_score = .n if size_score_orig == "Not Applicable"
372 //
373 recode size_score (1 2 3 = 1) (4 = 2) (5 = 3), gen(size_score_simple)
374 label values size_score_simple importance_simple
375 //
376 tabulate safety_score_orig
377 gen safety_score = .
378 replace safety_score = 1 if safety_score_orig == "1"
379 replace safety_score = 2 if safety_score_orig == "2"
380 replace safety_score = 3 if safety_score_orig == "3"
381 replace safety_score = 4 if safety_score_orig == "4"
382 replace safety_score = 5 if safety_score_orig == "5"
383 replace safety_score = .n if safety_score_orig == "Not Applicable"
384 //
385 recode safety_score (1 2 3 = 1) (4 = 2) (5 = 3), gen(safety_score_simple)
386 label values safety_score_simple importance_simple
387 //
388 tabulate car_availability_orig
389 gen car_availability = .
390 replace car_availability = 2 if car_availability_orig == "Yes, I own a car"
391 replace car_availability = 1 if car_availability_orig == "Yes, I have access to a car"
392 replace car_availability = 0 if car_availability_orig == "No"
393 label define car_availability 2 "Yes, I own a car" 1 "Yes, I have access to a car" 0 "No"
394 label values car_availability car_availability
395 //
396 recode car_availability (1 2 = 1 Yes) (3 = 0 No), gen(car_availability_binary)
397 //
398 tabulate dog_orig
399 gen dog = .
400 replace dog = 0 if dog_orig == "No"
401 replace dog = 1 if dog_orig == "Yes"
402 replace dog = .n if dog_orig == "Not Applicable"
403 label values dog yesno
404 //
405 tabulate gym_orig
406 gen gym = .
407 replace gym = 0 if gym_orig == "No"
408 replace gym = 1 if gym_orig == "Yes"
409 replace gym = . if gym_orig == "Yes, No"
410 label values gym yesno
411 //
412 tabulate retired_orig
413 gen retired = .
414 replace retired = 0 if retired_orig == "No"
415 replace retired = 1 if retired_orig == "Yes"
416 label values retired yesno

```


clean_survey_data.do - Printed on 09-Aug-16 21:17:05

```

417 //
418 tabulate student_orig
419 gen student = .
420 replace student = 0 if student_orig == "No"
421 replace student = 1 if student_orig == "Yes"
422 replace student = . if student_orig == "Yes, No"
423 label values student yesno
424 //
425 tabulate homeowner_orig
426 gen homeowner = .
427 replace homeowner = 0 if homeowner_orig == "No"
428 replace homeowner = 1 if homeowner_orig == "Yes"
429 replace homeowner = . if homeowner_orig == "Yes, No"
430 label values homeowner yesno
431 //
432 tabulate employment_orig
433 gen employment = .
434 replace employment = 0 if employment_orig == "No"
435 replace employment = 1 if employment_orig == "Yes"
436 label values employment yesno
437 //
438 tabulate full_time_orig
439 gen full_time = .
440 replace full_time = 0 if full_time_orig == "No"
441 replace full_time = 1 if full_time_orig == "Yes"
442 replace full_time = .n if full_time_orig == "Not Applicable"
443 label values full_time yesno
444 //
445 tabulate pa_work_orig
446 gen pa_work = .
447 replace pa_work = 0 if pa_work_orig == "No"
448 replace pa_work = 1 if pa_work_orig == "Yes"
449 replace pa_work = .n if pa_work_orig == "Not Applicable"
450 label values pa_work yesno
451 //
452 tabulate children_orig
453 gen children = .
454 replace children = 0 if children_orig == "No"
455 replace children = 1 if children_orig == "Yes"
456 label values children yesno
457 //
458 tabulate children_l2_orig
459 gen children_l2 = .
460 replace children_l2 = 1 if children_l2_orig == "1 to 2"
461 replace children_l2 = 2 if children_l2_orig == "3 or more"
462 replace children_l2 = 3 if children_l2_orig == "None"
463 replace children_l2 = .n if children_l2_orig == "Not Applicable"
464 label define children_l2 1 "1 to 2" 2 "3 or more" 3 "None"
465 label values children_l2 children
466 //
467 tabulate children_home_orig
468 gen children_home = .
469 replace children_home = 1 if children_home_orig == "1 to 2"
470 replace children_home = 2 if children_home_orig == "3 or more"
471 replace children_home = 3 if children_home_orig == "None"
472 replace children_home = .n if children_home_orig == "Not Applicable"
473 label values children_home children
474 //
475 tabulate stay_home_parent_orig
476 gen stay_home_parent = .
477 replace stay_home_parent = 0 if stay_home_parent_orig == "No"
478 replace stay_home_parent = 1 if stay_home_parent_orig == "Yes"
479 replace stay_home_parent = .n if stay_home_parent_orig == "Not Applicable"
480 label values stay_home_parent yesno
481 //
482 tabulate education_orig
483 gen education = .
484 replace education = 1 if education_orig == "Some high school"
485 replace education = 2 if education_orig == "High school completed"
486 replace education = 3 if education_orig == "Some College"
487 replace education = 4 if education_orig == "Technical or trade school certificate"

```

clean_survey_data.do - Printed on 09-Aug-16 21:17:05

```

488 replace education = 5 if education_orig == "Higher education degree"
489 replace education = 6 if education_orig == "University degree"
490 replace education = 6 if education_orig == "Some high school, University degree"
491 label define education 1 "Some high school" 2 "High school completed" 3 "Some College" 4
    "Technical or trade school certificate" 5 "Higher education degree" 6 "University degree"
492 label values education education
493 //
494 recode education (1 = 1 No-Highschool) (2 3 = 2 Highschool) (4 5 6 = 3 Higher-Education),
    gen(education_simple)
495 //
496 tabulate household_income_orig
497 gen income = .
498 replace income = 1 if household_income_orig == "less than $25,000"
499 replace income = 2 if household_income_orig == "$25,000 to $49,999"
500 replace income = 3 if household_income_orig == "$50,000 to $74,999"
501 replace income = 4 if household_income_orig == "$75,000 to $99,999"
502 replace income = 5 if household_income_orig == "$100,000 to $149,999"
503 replace income = 6 if household_income_orig == "$150,000 or more"
504 label define income 1 "less than $25,000" 2 "$25,000 to $49,999" 3 "$50,000 to $74,999" 4
    "$75,000 to $99,999" 5 "$100,000 to $149,999" 6 "$150,000 or more"
505 label values income income
506 //
507 tabulate ethnicity_orig
508
509 gen ethnicity_afr_am_black = 0
510 replace ethnicity_afr_am_black = 1 if strpos(ethnicity_orig, "African American")
511 replace ethnicity_afr_am_black = 1 if strpos(ethnicity_orig, "Black")
512 replace ethnicity_afr_am_black = . if ethnicity_orig == "No Answer"
513 label values ethnicity_afr_am_black yesno
514
515 gen ethnicity_hisp_latino = 0
516 replace ethnicity_hisp_latino = 1 if strpos(ethnicity_orig, "Hispanic")
517 replace ethnicity_hisp_latino = 1 if strpos(ethnicity_orig, "Latino")
518 replace ethnicity_hisp_latino = 1 if strpos(ethnicity_orig, "Mexican")
519 replace ethnicity_hisp_latino = . if ethnicity_orig == "No Answer"
520 label values ethnicity_hisp_latino yesno
521
522 gen ethnicity_white = 0
523 replace ethnicity_white = 1 if strpos(ethnicity_orig, "White")
524 replace ethnicity_white = . if ethnicity_orig == "No Answer"
525 label values ethnicity_white yesno
526
527 gen ethnicity_asian = 0
528 replace ethnicity_asian = 1 if strpos(ethnicity_orig, "Asian")
529 replace ethnicity_asian = 1 if strpos(ethnicity_orig, "Indian")
530 replace ethnicity_asian = . if ethnicity_orig == "No Answer"
531 label values ethnicity_asian yesno
532
533 gen ethnicity_native_am = 0
534 replace ethnicity_native_am = 1 if strpos(ethnicity_orig, "Native American")
535 replace ethnicity_native_am = . if ethnicity_orig == "No Answer"
536 label values ethnicity_native_am yesno
537
538 gen ethnicity_other = 0
539 replace ethnicity_other = 1 if strpos(ethnicity_orig, "Maroccan")
540 replace ethnicity_other = 1 if strpos(ethnicity_orig, "Berber")
541 replace ethnicity_other = 1 if strpos(ethnicity_orig, "Melungeon")
542 replace ethnicity_other = 1 if strpos(ethnicity_orig, "Other")
543 replace ethnicity_other = 1 if strpos(ethnicity_orig, "Palestinian")
544 replace ethnicity_other = 1 if strpos(ethnicity_orig, "West Indian")
545 replace ethnicity_other = 1 if strpos(ethnicity_orig, "Norwegian")
546 replace ethnicity_other = . if ethnicity_orig == "No Answer"
547 label values ethnicity_other yesno
548 //
549 tabulate health_orig
550 gen health = .
551 replace health = 1 if health_orig == "Very poor"
552 replace health = 2 if health_orig == "Poor"
553 replace health = 3 if health_orig == "Average"
554 replace health = 4 if health_orig == "Good"
555 replace health = 5 if health_orig == "Very good"

```

clean_survey_data.do - Printed on 09-Aug-16 21:17:05

```

556 replace health = 3 if health_orig == "Good, Average"
557 label define health 1 "Very poor" 2 "Poor" 3 "Average" 4 "Good" 5 "Very good"
558 label values health health
559 //
560 tabulate age_orig
561 gen age = age_orig
562 destring age, replace force
563 //
564 gen age_nominal = .
565 replace age_nominal = 14 if age > 79
566 replace age_nominal = 13 if age < 80
567 replace age_nominal = 12 if age < 75
568 replace age_nominal = 11 if age < 70
569 replace age_nominal = 10 if age < 65
570 replace age_nominal = 9 if age < 60
571 replace age_nominal = 8 if age < 55
572 replace age_nominal = 7 if age < 50
573 replace age_nominal = 6 if age < 45
574 replace age_nominal = 5 if age < 40
575 replace age_nominal = 4 if age < 35
576 replace age_nominal = 3 if age < 30
577 replace age_nominal = 2 if age < 25
578 replace age_nominal = 1 if age < 20
579 replace age_nominal = . if age == .
580 label define age_nominal 1 "18-19" 2 "20-24" 3 "25-29" 4 "30-34" 5 "35-39" 6 "40-44" 7
    "45-49" 8 "50-54" 9 "55-59" 10 "60-64" 11 "65-69" 12 "70-74" 13 "75-79" 14 "80+"
581 label values age_nominal age_nominal
582 //
583 gen age_nominal_simple = .
584 replace age_nominal_simple = 5 if age > 59
585 replace age_nominal_simple = 4 if age < 60
586 replace age_nominal_simple = 3 if age < 45
587 replace age_nominal_simple = 2 if age < 30
588 replace age_nominal_simple = 1 if age < 25
589 replace age_nominal_simple = . if age == .
590 label define age_nominal_simple 1 "18-24" 2 "25-29" 3 "30-44" 4 "45-59" 5 "60+"
591 label values age_nominal_simple age_nominal_simple
592 //
593 tabulate height_orig
594
595 gen height_feet = height_feet_orig
596 destring height_feet, replace force
597
598 gen height_inches = height_inches_orig
599 destring height_inches, replace force
600
601 gen height_cm = height_cm_orig
602 destring height_cm, replace force
603 replace height_cm = (height_feet * 30.48) + (height_inches * 2.54) if height_orig == "U.S."
604 replace height_cm = .r if height_orig == "Not Readable"
605 replace height_cm = . if height_orig == "No Answer"
606
607 gen height_foot_decimal = .
608 replace height_foot_decimal = height_cm * 0.0328084
609 replace height_foot_decimal = .r if height_orig == "Not Readable"
610 replace height_foot_decimal = . if height_orig == "No Answer"
611 //
612 tabulate weight_orig
613
614 gen weight_lbs = weight_lbs_orig
615 destring weight_lbs, replace force
616
617 gen weight_kg = weight_kg_orig
618 destring weight_kg, replace force
619
620 replace weight_lbs = weight_kg * 2.20462 if weight_orig == "Metric"
621 replace weight_lbs = . if weight_orig == "No Answer"
622
623 replace weight_kg = weight_lbs * 0.453592 if weight_orig == "U.S."
624 replace weight_kg = . if weight_orig == "No Answer"
625

```


clean_survey_data.do - Printed on 09-Aug-16 21:17:05

```

626 gen bmi = (weight_kg / (height_cm / 100)^2)
627 replace bmi = . if height_orig == "No Answer"
628 replace bmi = . if weight_orig == "No Answer"
629
630 gen bmi_nominal = 4
631 replace bmi_nominal = 3 if bmi < 30
632 replace bmi_nominal = 2 if bmi < 25
633 replace bmi_nominal = 1 if bmi < 18.5
634 replace bmi_nominal = . if bmi == .
635 label define bmi_nominal 1 "Underweight" 2 "Normal or Healthy Weight" 3 "Overweight" 4
    "Obese"
636 label values bmi_nominal bmi_nominal
637 //
638 tabulate gender_orig
639 gen gender = .
640 replace gender = 1 if gender_orig == "Female"
641 replace gender = 2 if gender_orig == "Male"
642 replace gender = .s if gender_orig == "Don't want to specify"
643 label define gender 1 "Female" 2 "Male"
644 label values gender gender
645 //
646 tabulate resident_meck_orig
647 gen resident_meck = .
648 replace resident_meck = 0 if resident_meck_orig == "No"
649 replace resident_meck = 1 if resident_meck_orig == "Yes"
650 label values resident_meck yesno
651 //
652 tabulate residence_orig
653 gen residence = .
654 replace residence = 1 if residence_orig == "less than 1 year"
655 replace residence = 2 if residence_orig == "1-2 years"
656 replace residence = 3 if residence_orig == "3-5 years"
657 replace residence = 4 if residence_orig == "more than 5 years"
658 replace residence = .n if residence_orig == "Not Applicable"
659 label define residence 1 "less than 1 year" 2 "1-2 years" 3 "3-5 years" 4 "more than 5 years"
660 label values residence residence
661 //
662 gen zipcode = substr(zipcode_orig,1,5)
663 destring zipcode, replace force
664 replace zipcode = .r if zipcode_orig == "Not readable"
665 replace zipcode = .n if zipcode_orig == "Brazil"
666 replace zipcode = . if zipcode_orig == "No Answer"
667 //
668 gen home_street = home_street_orig
669 gen home_street_provided = 1
670 replace home_street_provided = 0 if home_street_orig == "No Answer"
671 label values home_street_provided yesno
672 //
673 gen inters_street = inters_street_orig
674 gen inters_street_provided = 1
675 replace inters_street_provided = 0 if inters_street_orig == "No Answer"
676 label values inters_street_provided yesno
677 //
678 gen corr_home_street = corr_home_street_orig
679 //
680 gen corr_inters_street = corr_inters_street_orig
681 //
682 gen lat = lat_orig
683 destring lat, replace force
684 //
685 gen lng = lng_orig
686 destring lng, replace force
687 //
688 tabulate inconsistencies_orig
689
690 gen incons_no_location = 0
691 replace incons_no_location = 1 if strpos(inconsistencies_orig, "No location")
692 label values incons_no_location yesno
693
694 gen incons_only_county = 0
695 replace incons_only_county = 1 if strpos(inconsistencies_orig, "Only County")

```

clean_survey_data.do - Printed on 09-Aug-16 21:17:05

```

696 label values incons_only_county yesno
697
698 gen incons_only_zipcode = 0
699 replace incons_only_zipcode = 1 if strpos(inconsistencies_orig, "Only Zipcode")
700 label values incons_only_zipcode yesno
701
702 gen incons_st_not_found = 0
703 replace incons_st_not_found = 1 if strpos(inconsistencies_orig, "Street(s) not found")
704 label values incons_st_not_found yesno
705
706 gen incons_st_not_in_zipcode = 0
707 replace incons_st_not_in_zipcode = 1 if strpos(inconsistencies_orig, "Street(s) not within
Zipcode")
708 label values incons_st_not_in_zipcode yesno
709
710 gen incons_st_not_intersecting = 0
711 replace incons_st_not_intersecting = 1 if strpos(inconsistencies_orig, "Streets not
intersecting")
712 label values incons_st_not_intersecting yesno
713
714 gen incons_main_st_not_provided = 0
715 replace incons_main_st_not_provided = 1 if strpos(inconsistencies_orig, "Street of home not
provided")
716 label values incons_main_st_not_provided yesno
717
718 gen incons_inters_st_not_provided = 0
719 replace incons_inters_st_not_provided = 1 if strpos(inconsistencies_orig, "Intersecting
street not provided")
720 label values incons_inters_st_not_provided yesno
721
722 gen incons_long_st = 0
723 replace incons_long_st = 1 if strpos(inconsistencies_orig, "Long street")
724 label values incons_long_st yesno
725
726 gen incons_none = 0
727 replace incons_none = 1 if strpos(inconsistencies_orig, "None")
728 label values incons_none yesno
729 //
730 tabulate accuracy_level_orig
731 gen accuracy_level = .
732 replace accuracy_level = 1 if strpos(accuracy_level_orig, "Poor accuracy")
733 replace accuracy_level = 2 if strpos(accuracy_level_orig, "Low accuracy")
734 replace accuracy_level = 3 if strpos(accuracy_level_orig, "Good accuracy")
735 replace accuracy_level = 4 if strpos(accuracy_level_orig, "High accuracy")
736 label define accuracy_level 1 "Poor accuracy" 2 "Low accuracy" 3 "Good accuracy" 4 "High
accuracy"
737 label values accuracy_level accuracy_level
738
739 drop *_orig
740
741 save "complete_survey_data_stata_clean.dta", replace
742 export delimited using "complete_survey_data_stata_clean.txt", delimiter(tab) replace
743
744 collapse (first) unique_id park_type (median) median_travel_time=travel_time, by(park_name)
745 label values park_type park_type
746 export delimited using "median_travel_time_per_park.txt", delimiter(tab) replace
747
748 clear
749 use "complete_survey_data_stata_clean"
750 keep survey_id unique_id park_type travel_time lat lng accuracy_level
751 export delimited using "travel_times.txt", delimiter(tab) replace
752
753 clear
754 use "complete_survey_data_stata_clean"
755 keep survey_id lat lng
756 export delimited using "survey_locations.txt", delimiter(tab) replace
757
758 clear
759 import delimited survey_locations_extended.txt
760 save "survey_locations_extended.dta", replace
761 sort survey_id

```


clean_survey_data.do - Printed on 09-Aug-16 21:17:05

```
762
763 clear
764 use "complete_survey_data_stata_clean"
765 sort survey_id
766 merge 1:1 survey_id using "survey_locations_extended"
767 //
768 rename in_meck in_meck_orig
769 gen in_meck = 0
770 replace in_meck = 1 if in_meck_orig == "YES"
771 label values in_meck yesno
772 //
773 drop _merge in_meck_orig
774 save "complete_survey_data_stata_clean.dta", replace
775 drop if in_meck == 0
776
777 collapse (first) unique_id park_type (median) median_travel_time=travel_time, by(park_name)
778
```

APPENDIX G: NORMALITY AND SKEWNESS TEST ON VISITOR'S ACCESS SCORE AND IMPORTANCE OF PARK CHARACTERISTICS

```
. use "complete_survey_data_stata_clean"
```

```
.  
. // Test distribution of access scores and importance scores for skewness  
. sktest *_score
```

Skewness/Kurtosis tests for Normality

Variable	Obs	Pr(Skewness)	Pr(Kurtosis)	joint	
				adj chi2(2)	Prob>chi2
access_score	482	0.0000	0.0000	.	0.0000
amenity_s~re	479	0.0000	0.1106	29.69	0.0000
aesthetic~re	486	0.0000	0.2251	37.08	0.0000
size_score	483	0.0000	0.0687	21.90	0.0000
safety_score	454	0.0000	0.0000	.	0.0000

```
. ladder access_score
```

Transformation	formula	chi2(2)	P(chi2)
cubic	acces~re^3	.	0.000
square	acces~re^2	39.61	0.000
identity	acces~re	.	0.000
square root	sqrt(acces~re)	.	0.000
log	log(acces~re)	.	0.000
1/(square root)	1/sqrt(acces~re)	.	0.000
inverse	1/acces~re	.	0.000
1/square	1/(acces~re^2)	.	0.000
1/cubic	1/(acces~re^3)	.	0.000

```
. ladder size_score
```

Transformation	formula	chi2(2)	P(chi2)
cubic	size_~re^3	.	.
square	size_~re^2	.	0.000
identity	size_~re	21.90	0.000
square root	sqrt(size_~re)	53.76	0.000
log	log(size_~re)	.	0.000
1/(square root)	1/sqrt(size_~re)	.	0.000
inverse	1/size_~re	.	0.000
1/square	1/(size_~re^2)	.	0.000
1/cubic	1/(size_~re^3)	.	0.000

```
. ladder amenity_score
```

Transformation	formula	chi2(2)	P(chi2)
cubic	ameni~re^3	.	.
square	ameni~re^2	.	0.000
identity	ameni~re	29.69	0.000
square root	sqrt(ameni~re)	60.44	0.000
log	log(ameni~re)	.	0.000
1/(square root)	1/sqrt(ameni~re)	.	0.000
inverse	1/ameni~re	.	0.000
1/square	1/(ameni~re^2)	.	0.000
1/cubic	1/(ameni~re^3)	.	0.000

```
. ladder aesthetic_score
```

Transformation	formula	chi2(2)	P(chi2)
cubic	aesth~re^3	.	.
square	aesth~re^2	.	0.000
identity	aesth~re	37.08	0.000
square root	sqrt(aesth~re)	.	0.000
log	log(aesth~re)	.	0.000
1/(square root)	1/sqrt(aesth~re)	.	0.000
inverse	1/aesth~re	.	0.000
1/square	1/(aesth~re^2)	.	0.000
1/cubic	1/(aesth~re^3)	.	0.000

```
. ladder safety_score
```

Transformation	formula	chi2(2)	P(chi2)
cubic	safet~re^3	.	0.000
square	safet~re^2	.	0.000
identity	safet~re	.	0.000
square root	sqrt(safet~re)	.	0.000
log	log(safet~re)	.	0.000
1/(square root)	1/sqrt(safet~re)	.	0.000
inverse	1/safet~re	.	0.000
1/square	1/(safet~re^2)	.	0.000
1/cubic	1/(safet~re^3)	.	0.000

```
.
. // reomove non-residents from the data
. drop if in_meck == 0
(102 observations deleted)
```

```
.
. // Test again for skewness
. sktest *_score
```

Skewness/Kurtosis tests for Normality

Variable	Obs	Pr(Skewness)	Pr(Kurtosis)	adj chi2(2)	joint Prob>chi2
access_score	384	0.0000	0.0001	68.34	0.0000
amenity_s~re	384	0.0000	0.1749	26.26	0.0000
aesthetic~re	388	0.0000	0.3590	30.29	0.0000
size_score	386	0.0000	0.1251	19.91	0.0000
safety_score	359	0.0000	0.0000	.	0.0000

```
. ladder access_score
```

Transformation	formula	chi2(2)	P(chi2)
cubic	acces~re^3	.	0.000
square	acces~re^2	30.03	0.000
identity	acces~re	68.34	0.000
square root	sqrt(acces~re)	.	0.000
log	log(acces~re)	.	0.000
1/(square root)	1/sqrt(acces~re)	.	0.000
inverse	1/acces~re	.	0.000
1/square	1/(acces~re^2)	.	0.000
1/cubic	1/(acces~re^3)	.	0.000

```
. ladder size_score
```

Transformation	formula	chi2(2)	P(chi2)
cubic	size_~re^3	.	.
square	size_~re^2	.	0.000
identity	size_~re	19.91	0.000
square root	sqrt(size_~re)	45.91	0.000
log	log(size_~re)	.	0.000
1/(square root)	1/sqrt(size_~re)	.	0.000
inverse	1/size_~re	.	0.000
1/square	1/(size_~re^2)	.	0.000
1/cubic	1/(size_~re^3)	.	0.000

```
. ladder amenity_score
```

Transformation	formula	chi2 (2)	P (chi2)
cubic	ameni~re^3	.	.
square	ameni~re^2	.	0.000
identity	ameni~re	26.26	0.000
square root	sqrt (ameni~re)	50.56	0.000
log	log (ameni~re)	.	0.000
1/(square root)	1/sqrt (ameni~re)	.	0.000
inverse	1/ameni~re	.	0.000
1/square	1/ (ameni~re^2)	.	0.000
1/cubic	1/ (ameni~re^3)	.	0.000

```
. ladder aesthetic_score
```

Transformation	formula	chi2 (2)	P (chi2)
cubic	aesth~re^3	.	.
square	aesth~re^2	.	0.000
identity	aesth~re	30.29	0.000
square root	sqrt (aesth~re)	70.70	0.000
log	log (aesth~re)	.	0.000
1/(square root)	1/sqrt (aesth~re)	.	0.000
inverse	1/aesth~re	.	0.000
1/square	1/ (aesth~re^2)	.	0.000
1/cubic	1/ (aesth~re^3)	.	0.000

```
. ladder safety_score
```

Transformation	formula	chi2 (2)	P (chi2)
cubic	safet~re^3	66.25	0.000
square	safet~re^2	.	0.000
identity	safet~re	.	0.000
square root	sqrt (safet~re)	.	0.000
log	log (safet~re)	.	0.000
1/(square root)	1/sqrt (safet~re)	.	0.000
inverse	1/safet~re	.	0.000
1/square	1/ (safet~re^2)	.	0.000
1/cubic	1/ (safet~re^3)	.	0.000

```
. clear

. use "complete_survey_data_stata_clean"

.
. // Test distribution of access scores and importance scores for skewness
. sktest travel_time
```

Skewness/Kurtosis tests for Normality

Variable	Obs	Pr (Skewness)	Pr (Kurtosis)	_____ joint _____	
				adj chi2 (2)	Prob>chi2
travel_time	492	0.0000	0.0000	.	0.0000

```
. ladder travel_time
```

Transformation	formula	chi2 (2)	P (chi2)
cubic	travel~e^3	.	0.000
square	travel~e^2	.	0.000
identity	travel~e	.	0.000
square root	sqrt (travel~e)	.	0.000
log	log (travel~e)	9.25	0.010
1/(square root)	1/sqrt (travel~e)	.	0.000
inverse	1/travel~e	.	0.000
1/square	1/(travel~e^2)	.	0.000
1/cubic	1/(travel~e^3)	.	0.000

APPENDIX H: SURVEY DATA – DESCRIPTIVE STATISTICS AND STATISTICAL ANALYSIS (STATA)

interesting_stats_surveys2.do - Printed on 09-Aug-16 21:15:34

```

1  clear
2  use "complete_survey_data_stata_clean"
3
4  // Number of surveys per park (residents vs. non-residents)
5  tab park_type in_meck
6
7  //////////////////////////////////////
8  // Research question 3: //
9  // Perceived access //
10 //////////////////////////////////////
11 clear
12 use "complete_survey_data_stata_clean"
13
14 // Test distribution of access scores and importance scores for skewness
15 sktest *_score
16 ladder access_score
17 ladder size_score
18 ladder amenity_score
19 ladder aesthetic_score
20 ladder safety_score
21
22 // reomove non-residents from the data
23 drop if in_meck == 0
24
25 // Test again for skewness
26 sktest *_score
27 ladder access_score
28 ladder size_score
29 ladder amenity_score
30 ladder aesthetic_score
31 ladder safety_score
32
33 // Distribution is not normal even after tranformations
34 // ordinal logistic regression is advised
35 clear
36 use "complete_survey_data_stata_clean"
37
38 // reomove non-residents from the data
39 drop if in_meck == 0
40
41 // fromat the scores
42 format *_score %12.1f
43 tabstat *_score, stats(mean count) long format c(s)
44
45 // Test association between access scores and socio economic characteristics
46 ologit access_score i.homeowner
47 ologit access_score i.income
48 ologit access_score i.education_simple
49 ologit access_score i.ethnicity_afr_am
50 ologit access_score i.ethnicity_white
51 ologit access_score i.ethnicity_hisp_latino
52 // ALL NON-SIGNIFICANT
53
54 ologit access_score_simple i.gender
55 // SIGNIFICANT within 95% confidence level
56
57 // Test association between access scores and transportation used
58 ologit access_score i.car_availability_binary
59 // SIGNIFICANT within 95% confidence level
60
61 // Test association between access scores and Body Mass Index (BMI)
62 ologit access_score bmi
63 // SIGNIFICANT within 95% confidence level
64
65 // Test association between access scores and availability of park within
66 // a 10-minute walk
67 ologit access_score cp_walk_time
68 // SIGNIFICANT within 95% confidence level
69
70 // Model research question 3
71 ologit access_score i.cp_walk_time_simple bmi i.gender i.car_availability_binary

```

interesting_stats_surveys2.do - Printed on 09-Aug-16 21:15:34

```

72
73 // Re-run logit model on entire survey population
74 clear
75 use "complete_survey_data_stata_clean"
76
77 // Model research question 3
78 ologit access_score i.cp_walk_time_simple bmi i.gender i.car_availability_binary
79
80
81 // Research question 4:
82 // Willingness to travel
83 //
84
85
86 clear
87 use "complete_survey_data_stata_clean"
88
89 // Test distribution of access scores and importance scores for skewness
90 sktest travel_time
91 ladder travel_time
92
93 // reomove non-residents from the data
94 drop if in_meck == 0
95
96 // Test again for skewness
97 sktest travel_time
98 ladder travel_time
99
100 // Tabulate respondents on walk time to closest park
101 tab cp_walk_time_simple
102
103 // Tabulate median and 95th percentile of travel times to park surveyed
104 tabstat travel_time, stats(median p95)
105
106 // Tabulate travel time for different socio-economic characteristics
107 tabstat travel_time, by(age_nominal_simple) stats(median p95 count)
108 tabstat travel_time, by(income) stats(median p95 count)
109 tabstat travel_time, by(education_simple) stats(median p95 count)
110 tabstat travel_time, by(car_availability) stats(median p95 count)
111 tabstat travel_time, by(children_home) stats(median p95 count)
112
113 // change from one category to simpler categories (no impact on median)
114 tabstat travel_time, by(park_type) stats(median p95 count)
115 tabstat travel_time, by(park_type_simple) stats(median p95 count)
116
117 // tabulate the travel time variable that was recoded on an ordinal scale
118 tab travel_time_ordinal
119
120 // Test association between travel time and socio economic characteristics
121 ologit travel_time_ordinal i.homeowner
122 ologit travel_time_ordinal i.income
123 ologit travel_time_ordinal i.education_simple
124 ologit travel_time_ordinal i.ethnicity_white
125 ologit travel_time_ordinal i.gender
126 ologit travel_time_ordinal i.gym
127 ologit travel_time_ordinal age
128 // ALL NON-SIGNIFICANT
129
130 ologit travel_time_ordinal i.ethnicity_afr_am
131 ologit travel_time_ordinal i.ethnicity_hisp_latino
132 // SIGNIFICANT within 90% confidence level
133
134 // Test association between travel time and Body Mass Index (BMI)
135 ologit travel_time_ordinal bmi
136 // NON-SIGNIFICANT
137
138 // Test association between travel time and park type
139 ologit travel_time_ordinal i.park_type
140 // SIGNIFICANT within 95% confidence level
141
142 // Test association between travel time and availability of park within

```


interesting_stats_surveys2.do - Printed on 09-Aug-16 21:15:34

```

143 // a 10-minute walk
144 ologit travel_time_ordinal cp_walk_time
145 // SIGNIFICANT within 95% confidence level
146
147 // Model research question 4
148 ologit travel_time i.park_type_simple i.cp_walk_time_simple i.ethnicity_hisp_latino
149
150 // Re-run logit model on entire survey population
151 clear
152 use "complete_survey_data_stata_clean"
153
154 // Model research question 4
155 ologit travel_time i.park_type_simple i.cp_walk_time_simple i.ethnicity_hisp_latino
156
157
158 // Research question 5:
159 // Physical activity at parks
160 //
161
162 clear
163 use "complete_survey_data_stata_clean"
164
165 drop if in_meck == 0
166
167 // Tabulate responses to question about physical activity
168 tabstat travel_time age bmi access_score, by(pa_today) stats(median p95 count) c(s)
169 tabstat travel_time age bmi access_score, by(pa_freq_parks_simple) stats(median p95 count) c
170 (s)
171
172 // Test association between physical activity and socio economic characteristics
173 logit pa_today i.homeowner
174 logit pa_today i.income
175 logit pa_today i.education_simple
176 logit pa_today i.ethnicity_hisp_latino
177 logit pa_today i.gender
178 logit pa_today i.gym
179 logit pa_today age
180 // ALL NON-SIGNIFICANT
181
182 logit pa_today i.ethnicity_afr_am
183 logit pa_today i.ethnicity_white
184 // SIGNIFICANT within 90% confidence level
185
186 // Test association between physical activity and park type
187 logit pa_today i.park_type
188 logit pa_today i.park_type_simple
189
190
191 // Model research question 5
192 logit pa_today i.park_type_simple i.ethnicity_afr_am
193
194 // Re-run logit model on entire survey population
195 clear
196 use "complete_survey_data_stata_clean"
197
198 // Model research question 5
199 logit pa_today i.park_type_simple i.ethnicity_afr_am
200

```

APPENDIX I: PARKS AND RECREATIONAL FACILITIES DATA – CLEANING PROCESS (PYTHON)

1. Cleaning Framework

clean_all_datasets_framework.py

```

1: # Author:                               Coline C. Dony
2: # Date first written:                   May 8, 2016
3: # Date last updated:                   May 28, 2016
4: # Purpose:                             Data handling for dissertation
5: #
6: # Problem Statement:
7: # For my dissertation, different datasets need to be collected, cleaned and merged
8: # together. To keep track of all collection methods, edits and methods behind
9: # merging data from different sources together, this script executes each step
10: # of the data preparation, by calling each script in which all steps are
11: # documented and commented.
12:
13:
14: """
15:     1. PYTHON LIBRARIES
16:     all libraries necessary to run all scripts
17: """
18:
19: import arcpy
20: import os
21: import urllib2
22: import datetime
23: import time
24: import re
25: import itertools
26:
27:
28: """
29:     2. WORKSPACE DEFINITION
30:     root folder in which all datasets and scripts are located
31: """
32:
33: # Path to root folder
34: workfolder = r'C:\Dissertation_Materials\Data\core_datasets'
35:
36: # Change the working directory to the root folder
37: os.chdir(workfolder)
38:
39: # Change the ArcGIS environment directory to the root folder
40: arcpy.env.workspace = workfolder
41:
42: # Allowing overwriting of files when applying functions from
43: # the arcpy library
44: arcpy.env.overwriteOutput = True
45:
46: arcpy.CheckOutExtension('Spatial')
47: arcpy.CheckOutExtension("Network")
48:
49: # Generating an empty folder (if the folder does not already exist) in which all
50: # temporary datasets that are generated will be stored.
51: if not os.path.exists('temp'): os.makedirs('temp')
52:
53:
54: """
55:     3. COMMON DICTIONARIES
56:     dictionaries that are used in multiple scripts.
57: """
58:
59: # GREENWAY DICTIONARY
60: # dictionary of all developed greenways, which reflect the information in DPR's
61: # inventory (Excel file). This dictionary is used in multiple scripts.
62: greenway_abbreviations = {
63:     'BRIAR CREEK GREENWAY - ARNOLD': 'BRCG_A',
64:     'CAMPBELL CREEK GREENWAY': 'CMCG',
65:     'CLARKS CREEK GREENWAY': 'CLRK',
66:     'FOUR MILE CREEK GREENWAY - MATTHEWS': 'FMCG_M',
67:     'IRWIN CREEK GREENWAY - SYCAMORE': 'IRWG_S',
68:     'LTSCG_C',
69:     'BRIAR CREEK GREENWAY - COLONY': 'BRCG_C',
70:     'CHARLOTTE CITY GREENWAY': 'CCGWY',
71:     'FOUR MILE CREEK GREENWAY - JOHNSTON': 'FMCG_J',
72:     'IRWIN CREEK GREENWAY - CLANTON': 'IRWG_C',
73:     'LITTLE SUGAR CREEK GREENWAY - CUMBERLAND':

```

clean_all_datasets_framework.py

```

68:         'LITTLE SUGAR CREEK GREENWAY - ELIZABETH': 'LTSCG_E',    'LITTLE SUGAR CREEK GREENWAY - RAMBLEWOOD':
        'LTSCG_R',
69:         'MALLARD CREEK GREENWAY - AMARANTHUS': 'MLKG_A',          'MALLARD CREEK GREENWAY - UNIVERSITY': 'MLKG_U',
70:         'MCALPINE CREEK GREENWAY - JOHNSTON': 'MCAG_J',          'MCALPINE CREEK GREENWAY - MONROE': 'MCAG_M',
71:         'MCDOWELL CREEK GREENWAY': 'MCDG',                      'MCMULLEN CREEK GREENWAY': 'MCMG',
72:         'SIX MILE CREEK GREENWAY': 'SMCG',                      'SOUTH PRONG ROCKY RIVER GREENWAY': 'SOPG',
73:         'STEWART CREEK GREENWAY': 'STWC',                      'TOBY CREEK GREENWAY': 'TBYC_P1',
74:         'TORRENCE CREEK GREENWAY': 'TRCK',                      'UNCC': 'UNCC_GWY',
75:         'WALKER BRANCH GREENWAY': 'WABG',                      'WESLEY HEIGHTS GREENWAY': 'WESG',
76:         'WEST BRANCH ROCKY RIVER GREENWAY': 'WBRR'    }
77:
78: # MISSING PARK PROPERTIES
79: # dictionary with all added park entrances
80: missing_parks_abbreviations = {
81:     'Abernathy Park': 'MISS_1',
82:     'Arthur Goodman Memorial Park': 'MISS_2',
83:     'Belle Johnston Park': 'MISS_3',
84:     'Choate Park / Steele Creek Athletic Association': 'MISS_4',
85:     'Hickory Grove United Methodist Athletic Fields': 'MISS_5',
86:     'Jack D. Hughes Memorial Park': 'MISS_6',
87:     'Mint Hill Veterans Memorial Park': 'MISS_7',
88:     'PARK ON WILGROVE / Mint Hill Municipal Park': 'MISS_8',
89:     'Smithville Park': 'MISS_9'
90: }
91:
92:
93: """
94:     4. COMMON SPATIAL REFERENCES
95:     spatial references that are used throughout the data manipulation.
96: """
97:
98: # Spatial Reference for North Carolina State Plane
99: # Full Name: 'NAD_1983_StatePlane_North_Carolina_FIPS_3200_Feet'
100: # ALL shapefiles for this study are projected to this spatial reference
101: sr_nc = arcpy.SpatialReference(2264)
102:
103: # Spatial Reference for World Geodetic System
104: # Full Name: ''
105: # Data scraped from the web is often in this spatial reference
106: sr_wgs = arcpy.SpatialReference(4326)
107:
108:
109: """
110:     5. EXECUTION OF SCRIPTS
111:     all collection, edits and merging of datasources.
112: """
113:
114: execfile(r'clean_greenways.py')
115: print "cleaned greenways"
116:
117: execfile(r'extract_greenway_entrances.py')
118: print "extracted greenway entrances"
119:
120: execfile(r'clean_park_locations.py')
121: print "cleaned park locations"
122:
123: execfile(r'clean_park_properties.py')
124: print "cleaned park properties"
125:
126: execfile(r'extract_additional_park_and_recreation_properties.py')
127: print "additional park and recreation properties added"
128:
129:
130: # Generate an empty dictionary in which the attributes of each feature (polygon)
131: # will be stored using "prk_prp_id" as unique identifier
132: parks_and_recreation = {}
133:
134: execfile(r'link_data_together.py')

```

clean_all_datasets_framework.py

```
135: print "data linked together"
136:
137: execfile(r'clean_attribute_information.py')
138: print "attribute information cleaned"
139:
140: execfile(r'edit_attribute_information.py')
141: print "attribute information edited"
142:
143: execfile(r'add_weblinks.py')
144: print "weblinks added"
145:
146: execfile(r'extract_clean_files.py')
147: print "clean files extracted"
148:
149: execfile(r'crime.py')
150: print "Crime variable done"
151:
152: execfile(r'clean_quality_of_life.py')
153: print "quality of life cleaned!"
154:
155: execfile(r'extend_survey_data.py')
156: print "survey data extended!"
157:
158: execfile(r'calculate_park_densities_and_convert_to_contours.py')
159: print "park densities calculated and converted to contours!"
160:
161: """
162:     6. DELETE TEMPORARY FILES
163:     deletes all files that have been generated during sub-steps in the data manipulation
164:     process.
165: """
166:
167: # Delete the folder that contains all files that have been generated in sub-steps
168: # NOTE: If it is necessary to inspect a file from these substeps, comment this line
169: # of code and the folder will not be deleted.
170: os.remove('temp')
```


2. Clean Greenways

clean_greenways.py

```

1: # Author: Coline C. Dony
2: # Date first written: May 2, 2016
3: # Date Last updated: May 24, 2016
4: # Purpose: Clean some properties of the Greenways
5: #
6: # Problem statement:
7: # Mecklenburg County's Greenways (currently developed) are publically available.
8: # A shapefile of these greenway segments can be downloaded at:
9: # http://maps.co.mecklenburg.nc.us/openmapping/data.html
10: # This shapefile (lines) contains each developed greenway, which is useful for my research.
11: # However, some cleaning and changes are applied to this dataset before I using it:
12: #
13: # Problem - 1: Greenways are usually named after the creek they follow. For example,
14: # Little Sugar Creek (LSC) Greenway follows the course of LSC. In this
15: # shapefile, all segments folloring LSC belong to LSC Greenway. However,
16: # in the inventory provided by the Department of Parks and Recreation
17: # (as an Excel file), three distinct segments of LSC Greenway are identified.
18: # Eventually, we will join the inventory (Excel file) to the features in
19: # this shapefile, we will make the distinctions made in the inventory.
20: # Solution: Based on the inventory (Excel file), shared by the Department of Park and
21: # Recreation, we provide distinct names to greenways with multiple parts.
22: #
23: # Problem - 2: The different greenway entrances are not publicaly available. In this
24: # research, the entrances of parks are used to measure accessibility. Adding
25: # entrances to greenways is an important contribution I would like to make
26: # to this study. Using the shapefile of developed greenways, I can extract all
27: # dangling points, which correspond to the Location (point) where a greenway
28: # can be accessed. However, the topology of the greenway shapefile is
29: # extremely poor (many segments or vertices don't connect to one another),
30: # making it impossible to extract dangles that would correspond to entrances.
31: # Solution: A topology is built on this shapefile to connect all segments and vertices
32: # that should be connected to one another. Once the topology is built, all
33: # dangling points can be extracted and will reflect greenway entrances.
34:
35:
36: """
37: 1. INPUT FILES
38: original, already existng files, used in this script
39: """
40:
41: # Shapefile (lines) that contains all developed greenway segments
42: # Source: http://maps.co.mecklenburg.nc.us/openmapping/data.html
43: greenways_original_shp = r'parks\greenways\Greenways.shp'
44:
45:
46: """
47: 2. TEMPORARY FILES
48: files that are generated in this script but are not the end product / output file.
49: This segment also presents the thought process of the cleaning and editing process.
50: """
51:
52: # Copy of shapefile (lines) that contains all developed greenway segments
53: greenways_shp = 'temp\Greenways.shp'
54:
55: # Shapefile (lines) that contains all developed greenway segments but where the overland
56: # segments have been removed (these are segments that run through neighborhoods).
57: # This shapefile will be edited to make distinctions between different parts of greenways.
58: greenways_non_overland_shp = r'temp\Greenways_non_overland.shp'
59:
60: # An empty geodatabase that is created to built a topology for the line-shapefile
61: geodatabase = r'greenways.gdb'
62:
63: # An empty feature dataset that is added to the geodatabase (this step is also necessary
64: # to built the topology, a topology needs to be built on a feature dataset)
65: feature_dataset = r'greenways'
66:
67: # An empty topology layer that is added in the feature dataset (topology rules will be
68: # added to this layer)

```

clean_greenways.py

```

69: topology = r'greenways_topology'
70:
71: # For certain functions, some shapefiles will be converted to Layer files
72: greenways_lyr = greenways_shp[:-4]
73:
74: """
75: 3. OUTPUT FILE
76: clean greenway shapefile that can link more closely with the inventory.
77: """
78:
79: # The shapefile (lines) that will be extracted once the topology is built and validated.
80: # Distinct segments of the same greenway, as in the inventory (Excel file) will be able
81: # to be extracted and all inconsistencies from the original shapefile are fixed so that
82: # dangles (greenway entrances) can then be extracted.
83: greenways_clean_shp = r'parks\greenways\Greenways_clean.shp'
84:
85: """
86: 4. SOLVE PROBLEM 1
87: distinguish parts of greenway as in the inventory (Excel file).
88: """
89:
90: # Make a copy of the original greenway shapefile (lines)
91: arcpy.CopyFeatures_management(greenways_original_shp, greenways_shp)
92:
93: # Distinguish different greenway segments based on the inventory (Excel file)
94: # received from the Department of Park and Recreation
95: # Using an update cursor, the name of each trail is edited for the 6 greenways
96: # that have different parts
97:
98: attribute_table = arcpy.da.UpdateCursor(greenways_shp, ['memo', 'trail_name', 'FID'])
99: for memo, name, fid in attribute_table:
100:
101:     if name == 'Briar Creek Greenway':
102:         if memo == 'Arnold Dr. to Masonic Dr.': name = 'Briar Creek Greenway - Arnold'
103:         else: name = 'Briar Creek Greenway - Colony'
104:
105:     elif name == 'Four Mile Creek Greenway':
106:         memos = [ 'Bevington Pl', 'Bevington PlElm Ln', 'Johnston Rd to Rea Rd',
107:                   'Johnston Rd Lower McAlpine Connection', 'Radner Ln', 'Rea Rd']
108:         if memo in memos: name = 'Four Mile Creek Greenway - Johnston'
109:         else: name = 'Four Mile Creek Greenway - Matthews'
110:
111:     elif name == 'Irwin Creek Greenway':
112:         memos = ['Greenleaf Av', 'Rays Splash Planet', 'S Sycamore St', 'Sycamore to I-77']
113:         if memo in memos: name = 'Irwin Creek Greenway - Sycamore'
114:         else: name = 'Irwin Creek Greenway - Clanton'
115:
116:     elif name == 'Mallard Creek Greenway':
117:         memos = [ 'UNCC Research Park', 'Mallard Creek Greenway', 'Mallard Crk Ch Rd to I-85',
118:                   'uncc research', 'Kirk Farm Fields', 'UNCC Research Park to Clark's Crk Grnwy']
119:         if memo in memos: name = 'Mallard Creek Greenway - University'
120:         else: name = 'Mallard Creek Greenway - Amaranthus'
121:
122:     elif name in ['McAlpine Creek Greenway', 'Lower McAlpine Creek Greenway']:
123:         memos = [ 'Campbell Crk Grnwy to McAlpine Crk Park', 'cornwallis lane',
124:                   'creekwood apts', 'greyln business park', 'Independence Bv to Sardis Rd',
125:                   'kelford lane', 'Margaret Wallace Rd', 'Margaret Wallace to McAlpine Crk Grnwy',
126:                   'mcAlpine creek park', 'monroe road', 'old bell road parking', 'pineburr road',
127:                   'riverwood drive', 'sardis road', 'sunnywood lane', 'tara st',
128:                   'tower point dr ent', 'WT Harris to Margaret Wallace']
129:         if memo in memos or fid in [250,251,254]: name = 'McAlpine Creek Greenway - Monroe'
130:         else: name = 'McAlpine Creek Greenway - Johnston'
131:
132:     elif name == 'Little Sugar Creek Greenway':
133:         memos_ramblewood = ['Huntingtowne Farms Park', 'Ramblewood Ln']
134:         memos_elizabeth = [ '15th St', '16th @ McDowell St', '17th st', 'Alexander St Park',
135:                              'Alexander St Park to 12th St', 'Belmont Av to Alexander St. Park',
136:                              'Bridge1', 'East 12th St', 'Greenway Crescent Ln', 'N Alexander St',

```

clean_greenways.py

```

137:         'Parkwood to Belmont']
138:     if memo in memos_ramblewood or fid in [93,94,95]:
139:         name = 'Little Sugar Creek Greenway - Ramblewood'
140:     elif memo in memos_elizabeth or fid in [52,55,68,337]:
141:         name = 'Little Sugar Creek Greenway - Elizabeth'
142:     else: name = 'Little Sugar Creek Greenway - Cumberland'
143:     else: continue
144:     attribute_table.updateRow([memo, name, fid])
145:
146: # Delete the update cursor
147: del attribute_table
148:
149:
150: """
151:     5. SOLVE PROBLEM 2
152:     creating a shapefile with clean topology.
153: """
154:
155: # Create an empty geodatabase
156: arcpy.CreateFileGDB_management(r'temp', geodatabase)
157:
158: # Spatial Reference for North Carolina State Plane
159: # Full Name: 'NAD_1983_StatePlane_North_Carolina_FIPS_3200_Feet'
160: # All shapefiles for this study are projected to this spatial reference
161: sr_nc = arcpy.SpatialReference(2264)
162: gdb_path = os.path.join('temp', geodatabase)
163: arcpy.CreateFeatureDataset_management(gdb_path, feature_dataset, sr_nc)
164:
165: # Full path of the feature dataset in the geodatabase
166: feature_dataset_path = os.path.join(gdb_path, feature_dataset)
167:
168: # A shapefile (lines) is generated based on the original greenways shapefile, where
169: # all overland connectors are removed (these are segments that run through neighborhoods).
170: arcpy.MakeFeatureLayer_management(greenways_shp, greenways_lyr)
171: query = "trail_type" <> 'Overland Connector'
172: arcpy.SelectLayerByAttribute_management(greenways_lyr, "NEW_SELECTION", query)
173: arcpy.CopyFeatures_management(greenways_lyr, greenways_non_overland_shp)
174:
175: # The line-shapefile of greenways (no overland connectors) is imported in the feature dataset
176: arcpy.FeatureClassToGeodatabase_conversion(greenways_non_overland_shp, feature_dataset_path)
177:
178: # Name of the feature class that was imported into the geodatabase
179: feature_class = arcpy.ListFeatureClasses('', '', feature_dataset_path)[0]
180:
181: # Create an empty topology layer
182: arcpy.CreateTopology_management(feature_dataset_path, topology)
183:
184: # Full paths of the feature class and the topology layer in the geodatabase
185: feature_class_path = os.path.join(feature_dataset_path, feature_class)
186: topology_path = os.path.join(feature_dataset_path, topology)
187:
188: # Add a feature class to the topology
189: arcpy.AddFeatureClassToTopology_management(topology_path, feature_class_path, 1, 1)
190:
191: # Add a rule to the topology (no dangles)
192: arcpy.AddRuleToTopology_management(topology_path, "Must Not Have Dangles (Line)", feature_class_path)
193:
194: # Set the cluster tolerance (distance within which two points are considered the same point)
195: arcpy.SetClusterTolerance_management(topology_path, 3.28084)
196:
197: # Validate the topology (will change the Feature Class)
198: arcpy.ValidateTopology_management(topology_path)
199:
200: # Extract the topology-fices Feature Class
201: feature_layer = feature_class_path + '_layer'
202: arcpy.MakeFeatureLayer_management(feature_class_path, feature_layer)
203: arcpy.SelectLayerByAttribute_management(feature_layer, "SWITCH_SELECTION")
204: arcpy.CopyFeatures_management(feature_layer, greenways_clean_shp)

```


3. Extract Greenway Entrances

```

                                extract_greenway_entrances.py
1: # Author:                      Coline C. Dony
2: # Date first written:          April 15, 2016
3: # Date last updated:           May 24, 2016
4: # Purpose:                     Extract greenway entrances (points)
5: #
6: # Problem statement:
7: # Mecklenburg County's park entrances are not directly available. However, based on the
8: # shapefile that contains all greenway segments, these entrances can be extracted.
9: # The shapefile of greenway segments is publicly available at:
10: # http://maps.co.mecklenburg.nc.us/openmapping/data.html
11: # This shapefile has been cleaned and edited using the "clean_greenways" python script.
12: #
13: # Solution:                     Based on the cleaned and edited shapefile (lines) that contains all greenway
14: #                               segments that are developed, we extract points of entry by selecting all the
15: #                               dangles.
16:
17: """
18:     1. INPUT FILES
19:     original, already existing files, used in this script
20: """
21:
22: # All greenways (lines) that are currently developed
23: # Source: http://maps.co.mecklenburg.nc.us/openmapping/data.html
24: # Edited: by Coline C. Dony in python script clean_greenways.py
25: greenways_clean_original_shp = r'parks\greenways\Greenways_clean.shp'
26:
27:
28: """
29:     2. TEMPORARY FILES
30:     files that are generated in this script but are not the end product / output file.
31:     This segment also presents the thought process of the cleaning and editing process.
32: """
33:
34: # Copy of shapefile (lines) containing developed greenway segments.
35: # Inconsistencies were fixed with a topology.
36: greenways_clean_shp = r'temp\Greenways_clean.shp'
37:
38: # Shapefile (lines) where all greenways are dissolved based on the trail name
39: # This allows to extract dangles, which will correspond to greenway entry points
40: greenways_dissolve_shp = r'temp\Greenways_clean_dissolve.shp'
41:
42: # Shapefile (points) that contains all dangles from the greenway line shapefile
43: greenways_dangles_shp = r'temp\Greenways_dangles.shp'
44:
45: # Shapefile (points) that contains all greenway entrances (multi-part feature per greenway)
46: greenways_entrances_shp = r'temp\Greenways_entrances.shp'
47:
48: # For certain functions, some shapefiles will be converted to layer files
49: greenways_lyr = greenways_shp[:-4]
50: greenways_entrances_lyr = greenways_entrances_shp[:-4]
51:
52: """
53:     3. OUTPUT FILE
54:     point shapefile containing all greenway entrances.
55: """
56:
57: # Shapefile (points) that contains all greenway entrances (one multi-part feature per greenway)
58: greenways_entrances_clean_shp = r'parks\park_locations\Greenway_entrances.shp'
59:
60: """
61:     4. SOLVE PROBLEM

```

 extract_greenway_entrances.py

```

62:         extract greenway entrances.
63:     """
64:
65:     # Make a copy of the clean greenway file
66:     arcpy.CopyFeatures_management(greenways_clean_original_shp, greenways_clean_shp)
67:
68:     # Dissolve all line features based on the trail name
69:     arcpy.MakeFeatureLayer_management(greenways_clean_shp, greenways_lyr)
70:     arcpy.Dissolve_management(greenways_lyr, greenways_dissolve_shp, 'trail_name')
71:
72:     # Extract all dangles fro the line features
73:     arcpy.FeatureVerticesToPoints_management(greenways_dissolve_shp, greenways_dangles_shp, 'DANGLE')
74:
75:     # Make a duplicate of the dangles
76:     arcpy.CopyFeatures_management(greenways_dangles_shp, greenways_entrances_shp)
77:
78:     # Some points are redundant. For example, there is a dangle at each end of a crossing street.
79:     # It would skew the number of entrances to keep those, therefore, if two points are located
80:     # within 30 meters from one another, only one of both points will be kept.
81:     arcpy.MakeFeatureLayer_management(greenways_entrances_shp, greenways_entrances_lyr)
82:     attribute_table = arcpy.da.SearchCursor(greenways_entrances_shp, "FID")
83:     for fid in attribute_table:
84:         point = arcpy.SelectLayerByAttribute_management(greenways_entrances_lyr, 'NEW_SELECTION',
85:             """FID" = %s""" % fid[0])
86:         points_within_30m = arcpy.SelectLayerByLocation_management(greenways_entrances_lyr,
87:             'WITHIN_A_DISTANCE', greenways_entrances_lyr, '30 Meters')
88:         if int(arcpy.GetCount_management(points_within_30m).getOutput(0)) > 1:
89:             point = arcpy.SelectLayerByAttribute_management(greenways_entrances_lyr, 'NEW_SELECTION',
90:                 """FID" = %s""" % fid[0])
91:             points_within_100m = arcpy.SelectLayerByLocation_management(greenways_entrances_lyr,
92:                 'WITHIN_A_DISTANCE', greenways_entrances_lyr, '100 Meters')
93:             attribute_table2 = arcpy.da.UpdateCursor(points_within_100m, 'FID')
94:             for row in attribute_table2:
95:                 if row != fid: attribute_table2.deleteRow()
96:             del attribute_table2
97:         del attribute_table
98:
99:     # Adding the completion date to each entrance based on the segment they are part of
100:     arcpy.AddField_management(greenways_entrances_shp, 'completion', 'DATE')
101:
102:     arcpy.MakeFeatureLayer_management(greenways_entrances_shp, greenways_entrances_lyr)
103:     attribute_table = arcpy.da.UpdateCursor(greenways_entrances_shp, ['FID', 'completion'])
104:     for fid, date in attribute_table:
105:         point = arcpy.SelectLayerByAttribute_management(greenways_entrances_lyr, 'NEW_SELECTION',
106:             """FID" = %s""" % fid)
107:         greenway_segment = arcpy.SelectLayerByLocation_management(greenways_lyr, 'BOUNDARY_TOUCHES',
108:             greenways_entrances_lyr)
109:         completion_date = [completion for completion in arcpy.da.SearchCursor(greenway_segment,
110:             'completion')]
111:         if len(completion_date) > 0: attribute_table.updateRow([fid, completion_date[0][0]])
112:     del attribute_table
113:
114:     # Make a copy of the clean point entrances
115:     arcpy.CopyFeatures_management(greenways_entrances_shp, greenways_entrances_clean_shp)

```

4. Clean Park Location Points

```

clean_park_locations.py

1: # Author: Coline C. Dony
2: # Date first written: April 12, 2016
3: # Date last updated: May 23, 2016
4: # Purpose: Create a unique identifier for park locations
5:
6: """
7:     1. INPUT FILES
8:     original, already existing files, used in this script
9: """
10:
11: # Shapefile (points) that contains all property entrances whether developed or
12: # undeveloped, and managed by the Department of Park and Recreation (DPR)
13: # Source: http://maps.co.mecklenburg.nc.us/openmapping/data.html
14: park_locations_original_shp = r'parks\park_locations\Park_Locations.shp'
15:
16: # Shapefile (points) that contains all additional park entrances
17: # Source: Digitized by Coline C. Dony on April 19, 2016
18: missing_locations_shp = r'parks\park_locations\Missing_Park_Locations.shp'
19:
20: # Shapefile (points) that contains all greenway entrances
21: # Source: Extracted from greenway shapefile using python script
22: greenways_entrances_shp = r'parks\park_locations\Greenway_entrances.shp'
23:
24:
25: """
26:     2. GENERATED TEMPORARY FILES
27:     files that are generated in this script but are not the end product / output file.
28:     This segment also presents the thought process of the cleaning and editing process.
29: """
30:
31: # Copy of the shapefile containing all property entrances managed by the DPR
32: park_locations_shp = r'temp\Park_Locations.shp'
33:
34: # Shapefile (points) that contains all park entrances and greenway entrances
35: park_locations_and_greenways_shp = r'temp\Park_Locations_and_Greenways.shp'
36:
37:
38: """
39:     3. OUTPUT FILE
40:     clean point shapefile that contains park entrances.
41: """
42:
43: # Shapefile (points) that contains all park and greenway entrances,
44: # in addition to missing parks
45: park_locations_clean_shp = r'parks\park_locations\Park_Locations_clean.shp'
46:
47:
48: """
49:     4. SOLVE PROBLEM 1
50:     make a property entrance identifier that is unique.
51: """
52: # Make a copy of the original property entrances shapefile
53: arcpy.CopyFeatures_management(park_locations_original_shp, park_locations_shp)
54:
55: # Add an empty field to the property entrances shapefile which will contain a unique
56: # identifier for all entrances
57: arcpy.AddField_management(park_locations_shp, 'prk_loc_id', 'TEXT')
58:
59: # Going through the property entrances shapefile row by row and updating the unique
60: # identifier field.
61: attribute_table = arcpy.da.UpdateCursor(park_locations_shp, ['prkid', 'prkname', 'prk_loc_id'])

```

clean_park_locations.py

```

62: for park in attribute_table:
63:
64:     # The park identifier 'FIRST' appears twice in 'prkid', here we make two separate identifiers
65:     if park[0] == 'FIRST' and park[1] == 'BREVARD CALDWELL PROPERTY': park[2] = 'FIRSTB'
66:     else : park[2] = park[0]
67:
68:     attribute_table.updateRow(park)
69: del attribute_table
70:
71:
72: """
73:     5. SOLVE PROBLEM 2
74:     add greenway entrances to the property entrance shapefile.
75: """
76:
77: # Insert the greenway entrance points into the shapefile
78: attribute_table_greenway_entrances = arcpy.da.SearchCursor(greenways_entrances_shp, ['trail_name',
79: 'completion', 'SHAPE@'])
79: attribute_table_park_locations = arcpy.da.InsertCursor(park_locations_shp, ['prkname', 'prk_loc_id',
80: 'prktype', 'prkstatus', 'prkdate', 'SHAPE@'])
80:
81: for trail_name, date, shape in attribute_table_greenway_entrances:
82:     name = trail_name.upper()
83:     prk_loc_id = greenway_abbreviations[name]
84:     attribute_table_park_locations.insertRow([name, prk_loc_id, 'GREENWAY', 'DEVELOPED', date,
85: shape])
85: del attribute_table_greenway_entrances
86: del attribute_table_park_locations
87:
88: """
89:     6. SOLVE PROBLEM 3
90:     add additional park entrances (from public sources).
91: """
92:
93: # Merge the two shapefiles together
94: arcpy.Merge_management([park_locations_shp, missing_locations_shp],
95: park_locations_and_greenways_shp)
95:
96: dissolve = {}
97:
98: # Going through the shapefile row by row to update the unique identifier for the missing parks
99: attribute_table = arcpy.da.SearchCursor(park_locations_and_greenways_shp, '*')
100: for entrance in attribute_table:
101:     name = entrance[3]
102:     prk_loc_id = missing_parks_abbreviations[name] if name in missing_parks_abbreviations else
103: entrance[-1]
103:     date = entrance[-7]
104:     shape = entrance[1]
105:
106:     if prk_loc_id in dissolve:
107:         dates = dissolve[prk_loc_id][-7]
108:         dissolve[prk_loc_id][-7] = dates + [date] if date is not None else dates
109:         dissolve[prk_loc_id][1] += [shape]
110:     else:
111:         dissolve[prk_loc_id] = list(entrance)
112:         dissolve[prk_loc_id][-1] = prk_loc_id
113:         dissolve[prk_loc_id][-7] = [date] if date is not None else []
114:         dissolve[prk_loc_id][1] = [shape]
115: del attribute_table
116:
117: # Generate an empty shapefile (polygons) in which all features associated with potential

```

clean_park_locations.py

```
118: # recreational activities (OSM and impervious surfaces) will be inserted
119: arcpy.CreateFeatureclass_management(workfolder, park_locations_clean_shp, 'MULTIPOINT',
    park_locations_and_greenways_shp, '', '', sr_nc)
120:
121: field_names = [f.name for f in arcpy.ListFields(park_locations_clean_shp)][2:]
122: attribute_table = arcpy.da.InsertCursor(park_locations_clean_shp, field_names + ['SHAPE@'])
123: for prk_loc_id in dissolve:
124:     attributes = dissolve[prk_loc_id][2:]
125:
126:     dates = attributes[-7]
127:     attributes[-7] = min(dates) if len(dates) > 0 else None
128:
129:     points = dissolve[prk_loc_id][1]
130:     multipoint = arcpy.Multipoint(arcpy.Array([arcpy.Point(*point) for point in points]))
131:
132:     attribute_table.insertRow(attributes + [multipoint])
133: del attribute_table
134:
```

5. Clean Park Properties

clean_park_properties.py

```

1: # Author: Coline C. Dony
2: # Date first written: April 12, 2016
3: # Date last updated: May 25, 2016
4: # Purpose: Clean the park properties shapefile to the needs of my research
5:
6: # Problem statement:
7: # Mecklenburg County's Department of Park and Recreation's (DPR) property boundaries are
8: # publically available. A shapefile of these properties can be downloaded at:
9: # http://maps.co.mecklenburg.nc.us/openmapping/data.html
10: # This shapefile contains all properties managed by DPR and is useful for my research.
11: # However it contains certain inconsistencies and does not directly link to the
12: # inventory (Excel file) shared by the DPR. This script will clean and edit this
13: # shapefile for the purposes of my research. To keep track of the source and edits
14: # made on each polygon, two attributes are added to the shapefile ("source" and
15: # "edits" respectively)
16:
17: # Problem - 1: The property identifier ("park_id") is not unique.
18: # Solution: The attribute "prk_prp_id" is generated to uniquely identify all features.
19:
20: # Problem - 2: Although this DPR properties shapefile also contains greenway properties,
21: # no distinction is made between developed and undeveloped greenways.
22: # For our research, making that distinction is important.
23: # Solution: Using the shapefile that contains all developed greenways (lines), greenway
24: # properties that are developed are separated from those that are undeveloped.
25:
26: # Problem - 3: The DPR properties shapefile is missing some properties that have clearly
27: # already been developed or proposed:
28: # A: Some developed greenways do not fall within a property polygon.
29: # B: The proposed outline for Doby Creek Greenway property does not appear
30: # C: Existing park properties are missing from the original data source
31: # D: Private Golf Courses
32: # Solution: Missing properties are added to the properties shapefile:
33: # A: A 10-Meter buffer is drawn around all existing greenways that
34: # do not fall within an existing property polygon.
35: # B: Using the Doby Creek Greenway proposed outline (as a KML), a 10-Meter
36: # buffer is drawn around the proposed outline.
37: # C: A manually digitized shapefile that contains missing park properties
38: # found on Google Maps and Bing Maps will be added to the shapefile.
39: # D: Golf course properties provided by the County
40:
41: # Problem - 4: The property identifier is not consistent with the identifier in the
42: # inventory (Excel file), which I want to link to this shapefile
43: # because the inventory hold additional attributes (e.g. amenities).
44: # The property identifier is not consistent with the identifier in the
45: # park locations shapefile (points) either.
46: # Solution: The attributes "invtry_id" and "prk_loc_id" are generated, matching
47: # identifiers in the inventory (excel file) and park locations shapefile
48: # (points) respectively.
49:
50:
51: """
52: 1. INPUT FILES
53: original, already existng files, used in this script
54: """
55:
56: # Shapefile (polygons) that contains all properties managed by the DPR
57: # Source: http://maps.co.mecklenburg.nc.us/openmapping/data.html
58: properties_original_shp = r'parks\parkproperty\ParkProperty.shp'
59:
60: # Shapefile (lines) that contains developed greenways
61: # Source: http://maps.co.mecklenburg.nc.us/openmapping/data.html

```


clean_park_properties.py

```

62: # Edited: by Coline C. Dony in python script clean_greenways.py
63: greenways_clean_original_shp = r'parks\greenways\Greenways_clean.shp'
64:
65: # Shapefile (polygons) that contains missing park properties that were found on
66: # Google Maps and Bing Maps
67: # Source: Digitized by Coline C. Dony on April 19, 2016
68: properties_missing_shp = r'parks\parkproperty\Missing_ParkProperty.shp'
69:
70: # Shapefile (polygons) that contains developed golfcourses
71: # Source: http://maps.co.mecklenburg.nc.us/openmapping/data.html
72: golfcourses_shp = r'parks\golf_courses\Golf_Courses.shp'
73:
74: # KML-file (Lines) that contains the proposed path for Doby Creek Greenway
75: # Source: https://www.google.com/maps/d/u/0/viewer?mid=zkhhfKmHHcmU.kpdyZDcoMhoU&hl=en\_US
76: creeks_shp = r'parks\creeks_streams\Creeks_Streams.shp'
77: creeks_lyr = creeks_shp[:-4]
78:
79: # Shapefile (Lines) that indicates which greenway properties should be split to separate
80: # out polygons in which a greenway is developed and where it is undeveloped
81: # Source: Digitized by Coline C. Dony on April 12, 2016
82: greenways_split_shp = r'parks\greenways\Greenway_splitter.shp'
83: greenways_split_lyr = greenways_split_shp[:-4]
84:
85:
86: """
87:     2. TEMPORARY FILES
88:     files that are generated in this script but are not the end product / output file.
89:     This segment also presents the thought process of the cleaning and editing process.
90: """
91:
92: # Copy of shapefile (polygons) that contains all properties from the DPR
93: properties_shp = 'temp\ParkProperty.shp'
94:
95: # Copy of shapefile (Lines) that contains developed greenways
96: greenways_clean_shp = 'temp\Greenways_clean.shp'
97:
98: # Shapefile (polygons) that contains greenway properties only
99: properties_gwys_shp = r'temp\ParkProperty_Greenways.shp'
100:
101: # Shapefile (polygons) that contains all properties except for greenway properties
102: properties_non_gwys_shp = r'temp\ParkProperty_Non_Greenways.shp'
103:
104: # Shapefile (polygons) that contains greenway properties only, as single part polygons
105: # The original shapefile is a multi-part polygon (e.g. Little Sugar Creek Greenway
106: # is made of multiple polygons. Since we want to separate developed from undeveloped
107: # parts of each greenway, we convert all polygons to single parts before we overlay
108: # them with the shapefiles (Lines) that contains developed greenways.
109: properties_gwys_as_single_shp = r'temp\ParkProperty_Greenways_as_singlepart.shp'
110:
111: # Shapefile (Lines) that contains developed greenways as single part lines.
112: # The original shapefile is a multi-part Lines (e.g. Little Sugar Creek Greenway
113: # is made of multiple Lines. Since we want to update the name of each greenway
114: # property with the greenway that passes through it, we split all Lines.
115: greenways_as_single_shp = r'temp\Greenways_as_singlepart.shp'
116:
117: # Shapefile (polygons) that contains greenway properties that have a developed greenway
118: # passing through it. These polygons are Labeled "DEVELOPED"
119: properties_gwys_developed_shp = 'temp\ParkProperty_Greenways_as_singlepart_developed.shp'
120:
121: # Shapefile (polygons) that contains greenway properties that do not have a developed
122: # greenway passing through it. These polygons are Labeled "UNDEVELOPED"

```

clean_park_properties.py

```

123: properties_gwys_undeveloped_shp = 'temp\ParkProperty_Greenways_as_singlepart_undeveloped.shp'
124:
125: # Shapefile (polygons) that contains single-part greenway properties that need to
126: # be splitted in two parts (one developed and one undeveloped part).
127: properties_gwys_to_split_1 = r'temp\ParkProperty_Greenways_as_singlepart_to_split_copy1.shp'
128:
129: # Shapefile (polygons) that is a duplicate of the shapefile that contains single-part
130: # greenway properties that need to be splitted (see shapefile above).
131: properties_gwys_to_split_2 = r'temp\ParkProperty_Greenways_as_singlepart_to_split_copy2.shp'
132:
133: # Shapefile (polygons) that is a merge of the shapefile that contains single-part greenway
134: # properties that need to be splitted and its copy. Each polygon appears twice so that the
135: # attributes are copied. Each polygon will be replaced with its respective splitted polygon
136: # (developed or undeveloped part).
137: properties_gwys_to_split_shp = r'temp\ParkProperty_Greenways_as_singlepart_to_split.shp'
138:
139: # Shapefile (polygons) that contains single-part greenway properties that do not need
140: # to be splitted
141: properties_gwys_not_to_split_shp = r'temp\ParkProperty_Greenways_as_singlepart_not_to_split.shp'
142:
143: # Shapefile (polygons) that contains the splitted greenway properties
144: properties_gwys_splitted_shp = r'temp\ParkProperty_Greenways_as_singlepart_to_split_splitted.shp'
145:
146: # Shapefile (lines) that contains greenway segments that are developed but that do not
147: # fall within any greenway property
148: greenways_w_missing_property_shp = 'temp\Greenways_w_missing_property.shp'
149:
150: # Shapefile (polygons) that contains 10-meter buffers around developed greenways that
151: # do not fall within any greenway property
152: properties_gwys_single_developed_missing_shp =
    'temp\ParkProperty_Greenways_as_singlepart_developed_missing.shp'
153:
154: # Shapefile (polygons) that contains contains a 10-meter buffer around the proposed
155: # path for Doby Creek Greenway
156: properties_creeks_shp = r'temp\ParkProperty_Creeks_Streams.shp'
157:
158: # Shapefile (polygons) that contains a more complete set of park and recreational
159: # properties
160: properties_single_merged_shp = 'temp\ParkProperty_as_singlepart.shp'
161:
162: # For certain functions, some shapefiles will be converted to layer files
163: properties_lyr = properties_shp[:-4]
164: properties_gwys_as_single_lyr = properties_gwys_as_single_shp[:-4]
165: properties_gwys_developed_lyr = properties_gwys_developed_shp[:-4]
166: properties_gwys_undeveloped_lyr = properties_gwys_undeveloped_shp[:-4]
167: properties_gwys_to_split_lyr = properties_gwys_to_split_shp[:-4]
168: properties_gwys_splitted_lyr = properties_gwys_splitted_shp[:-4]
169: greenways_clean_lyr = greenways_clean_shp[:-4]
170: greenways_w_missing_property_lyr = greenways_w_missing_property_shp[:-4]
171: greenways_as_single_lyr = greenways_as_single_shp[:-4]
172:
173:
174: """
175:     3. OUTPUT FILE
176:     clean park and recreational properties shapefile that can link more closely with
177:     the inventory (excel file) and park locations shapefile (points)
178: """
179:
180: # Shapefile (polygons) that contains a more complete set of park and recreational
181: # properties and a unique identifier for each feature (prk_prp_id).
182: properties_clean_edits_shp = 'parks\parkproperty\ParkProperty_clean_w_visible_edits.shp'

```

clean_park_properties.py

```

183:
184: # Shapefile (polygons) that contains a more complete set of park and recreational
185: # properties and the edits that were made to each feature (the identifiers are not
186: # unique).
187: properties_clean_shp = 'parks\parkproperty\ParkProperty_clean.shp'
188:
189:
190: """
191:     4. SOLVE PROBLEM 1
192:     generate "prk_prp_id", a unique identifier for properties
193: """
194:
195: # Make a copy of the original DPR's property shapefile (polygons)
196: arcpy.CopyFeatures_management(properties_original_shp, properties_shp)
197:
198: # Add an empty field to the DPR's property shapefile (polygons) which will be updated
199: arcpy.AddField_management(properties_shp, 'prk_prp_id', 'TEXT')
200:
201: # Add two empty fields to the DPR's property shapefile (polygons) to keep track of
202: # the source of each polygon and the edits that have been made to it
203: # These fields will be updated throughout this script
204: arcpy.AddField_management(properties_shp, 'source', 'TEXT')
205: arcpy.AddField_management(properties_shp, 'edits', 'TEXT')
206:
207: # Going through the DPR's property shapefile (polygons) feature-by-feature and update
208: # the fields that were generated (property identifier and source).
209: attribute_table = arcpy.da.UpdateCursor(properties_shp, ['park_id', 'property', 'prk_prp_id',
210: 'source'])
210: for park_id, name, prk_prp_id, source in attribute_table:
211:
212:     # The park_id 'LTSC' appears twice and can not be used as unique key
213:     # park_prp_id is generated to become a unique key for this shapefile
214:     if park_id == 'LTSC' and name == 'LITTLE SUGAR CREEK GREENWAY': prk_prp_id = 'LTSCG'
215:     else : prk_prp_id = park_id
216:
217:     # ALL features are marked with their original source
218:     source = 'park property file: maps.co.mecklenburg.nc.us/openmapping'
219:
220:     # Update the attribute information
221:     attribute_table.updateRow([park_id, name, prk_prp_id, source])
222:
223: # Close the update cursor
224: del attribute_table
225:
226:
227: """
228:     5. SOLVE PROBLEM 2
229:     separating out developed from undeveloped greenway properties.
230: """
231:
232:
233: # First, a shapefile is prepared with only greenway properties
234:
235: # Generating a shapefile (polygons) that contains greenway properties only
236: arcpy.MakeFeatureLayer_management(properties_shp, properties_lyr)
237: query = """park_type" = 'GREENWAY'"""
238: arcpy.SelectLayerByAttribute_management(properties_lyr, "NEW_SELECTION", query)
239: arcpy.CopyFeatures_management(properties_lyr, properties_gwys_shp)
240:
241: # Generating a shapefile (polygons) that contains all properties except greenways
242: arcpy.SelectLayerByAttribute_management(properties_lyr, "SWITCH_SELECTION")

```

clean_park_properties.py

```

243: arcpy.CopyFeatures_management(properties_lyr, properties_non_gwys_shp)
244:
245: # Converting the the polygons in the shapefile that contains greenway properties only
246: # from multi-parts to single-parts. This makes overlay with existng greenways more
247: # appropriate for our purpose (separating developed from undeveloped greenway properties)
248: arcpy.MultipartToSinglepart_management(properties_gwys_shp, properties_gwys_as_single_shp)
249:
250:
251: # Second, Some greenway properties have a developed greenway path running through it. For some
252: # of these polygons, the developed greenway passing through only takes up a minimal
253: # part of the entire polygon. To avoid mis-representation of which properties are already
254: # developed, greenway properties that have a significant part that is not developed
255: # will be separated out from the part that is developed.
256:
257: # Generating a shapefile (polygons) that contains the greenway polygons that need to
258: # be splitted up in a developed and undeveloped part. A duplicate of this shapefile
259: # is generated. Later this shapefile and its copy are merged to have a copy of the
260: # attributes.
261: arcpy.MakeFeatureLayer_management(properties_gwys_as_single_shp, properties_gwys_as_single_lyr)
262: arcpy.MakeFeatureLayer_management(greenways_split_shp, greenways_split_lyr)
263: arcpy.SelectLayerByLocation_management(properties_gwys_as_single_lyr, "CONTAINS",
greenways_split_lyr)
264: arcpy.CopyFeatures_management(properties_gwys_as_single_lyr, properties_gwys_to_split_1)
265: arcpy.CopyFeatures_management(properties_gwys_as_single_lyr, properties_gwys_to_split_2)
266:
267: # Generating a shapefile (polygons) that contains the greenway properties that do not
268: # need to be splitted up
269: arcpy.SelectLayerByAttribute_management(properties_gwys_as_single_lyr, "SWITCH_SELECTION")
270: arcpy.CopyFeatures_management(properties_gwys_as_single_lyr, properties_gwys_not_to_split_shp)
271:
272: # Generating a shapefile (polygons) that contains greenway polygons that need to
273: # be splitted up, copied twice. This way, the attributes get copied. Later the polygon
274: # shape will be replaced with the corresponding splitted polygon.
275: arcpy.Merge_management([properties_gwys_to_split_1, properties_gwys_to_split_2],
properties_gwys_to_split_shp)
276:
277: # Generating a shapefile (polygons) that contains the splitted greenway properties
278: arcpy.FeatureToPolygon_management([properties_gwys_to_split_1, greenways_split_lyr],
properties_gwys_splitted_shp)
279:
280: # Updating the polygons in the shapefile that contains greenway polygons that need to
281: # be splitted up, copied twice. This replaces their original polygon with the splitted
282: # polygon.
283: arcpy.MakeFeatureLayer_management(properties_gwys_splitted_shp, properties_gwys_splitted_lyr)
284: for fid in range(int(arcpy.GetCount_management(properties_gwys_splitted_lyr).getOutput(0))):
285:
286:     # Based on the FID, one splitted part of a greenway property is selected
287:     query = "FID" = %s" % fid
288:     split_polygon = arcpy.SelectLayerByAttribute_management(properties_gwys_splitted_lyr,
"NEW_SELECTION", query)
289:     split_polygon_shape = arcpy.da.SearchCursor(split_polygon, ["SHAPE@", "SHAPE@AREA"]).next()
290:
291:     # Inside the LTSCG feature, there are 4 small polygons that seem to be digitizing
292:     # mistakes. Since these 4 features are smaller than 1 square foot, we skip them
293:     if split_polygon_shape[1] > 1:
294:
295:         # The features in which this splitted polygon falls are selected
296:         # The first of the selected features is then replaced by the splitted polygon
297:         arcpy.MakeFeatureLayer_management(properties_gwys_to_split_shp,
properties_gwys_to_split_lyr)
298:         selected_properties = arcpy.SelectLayerByLocation_management(properties_gwys_to_split_lyr,
"CONTAINS", split_polygon)

```

clean_park_properties.py

```

299:         attribute_table_selection = arcpy.da.UpdateCursor(selected_properties, ["SHAPE@", 'edits',
'property'])
300:         shape, edits, name = attribute_table_selection.next()
301:         if name == 'WALKERS BRANCH GREENWAY': name = 'WALKER BRANCH GREENWAY'
302:         attribute_table_selection.updateRow([split_polygon_shape[0], 'shape\t', name])
303:
304:         # Close the update cursor
305:         del attribute_table_selection
306:
307:         # Close de search cursor
308:         del split_polygon_shape
309:
310: # Merge the greeway properties back together (the properties that did not need editing
311: # with the ones that were splitted into a developed and undeveloped part
312: arcpy.Merge_management([properties_gwys_to_split_shp, properties_gwys_not_to_split_shp],
properties_gwys_as_single_shp)
313:
314:
315: # Third, greenway properties that are developed are distinguished
316: # from the ones that are not developed
317:
318: # Make a copy of the cleaned and edited greenway shapefile (lines)
319: arcpy.CopyFeatures_management(greenways_clean_original_shp, greenways_clean_shp)
320:
321: # Converting the the lines in the shapefile that contains all developed greenways
322: # from multi-parts to single-parts. This makes overlay with greenway properties more
323: # appropriate for our purpose (separating developed from undeveloped greenway properties)
324: arcpy.SplitLine_management(greenways_clean_shp, greenways_as_single_shp)
325:
326: # Generate a shapefile (polygons) that contains developed greenway properties only
327: # This is done by selecting polygons in which a developed greenway runs through
328: arcpy.MakeFeatureLayer_management(properties_gwys_as_single_shp, properties_gwys_as_single_lyr)
329: arcpy.SelectLayerByLocation_management(properties_gwys_as_single_lyr, "INTERSECT",
greenways_clean_shp)
330: arcpy.CopyFeatures_management(properties_gwys_as_single_lyr, properties_gwys_developed_shp)
331:
332: # Generate a shapefile (polygons) that contains undeveloped greenway properties only
333: # This is done by selecting polygons in which no developed greenway runs through
334: arcpy.SelectLayerByAttribute_management(properties_gwys_as_single_lyr, "SWITCH_SELECTION")
335: arcpy.CopyFeatures_management(properties_gwys_as_single_lyr, properties_gwys_undeveloped_shp)
336:
337:
338: # Fourth, all developed greenway names are updated so that distinct parts of a greenway
339: # that can be found in the DPR's inventory (Excel file) can be extracted.
340:
341: # Gong through each developed greenway feature-by-feature
342: attribute_table = arcpy.da.UpdateCursor(properties_gwys_developed_shp, ['FID','property',
'prk_prp_id','park_type','status','edits'])
343: for fid, name, prk_prp_id, park_type, status, edits in attribute_table:
344:
345:     # Based on the FID, one developed greenway property (polygon) is selected
346:     arcpy.MakeFeatureLayer_management(properties_gwys_developed_shp, properties_gwys_developed_lyr)
347:     query = """FID" = %s""" % fid
348:     selected_property = arcpy.SelectLayerByAttribute_management(properties_gwys_developed_lyr ,
"NEW_SELECTION", query)
349:
350:     # ALL developed greenway segments (Lines) that fall within the greenway property
351:     # are selected
352:     arcpy.MakeFeatureLayer_management(greenways_as_single_shp, greenways_as_single_lyr)
353:     greenway_segments = arcpy.SelectLayerByLocation_management(greenways_as_single_lyr, "WITHIN",
selected_property)

```

clean_park_properties.py

```

354:
355:     # Some greenway properties (polygon) contain segments of various greenways (lines)
356:     # The segment with the longest length is selected to represent the name of the
357:     # greenway property
358:
359:     # Going through the selected greenway segments feature-by-feature
360:     attribute_table_gwys = arcpy.da.SearchCursor(greenway_segments, ['SHAPE@LENGTH', 'trail_name'])
361:     longest_segment = 0
362:     for segment_length, segment_name in attribute_table_gwys:
363:         if segment_length > longest_segment:
364:             longest_segment = segment_length
365:             name_longest_segment = segment_name.upper()
366:     prk_prp_id = greenway_abbreviations[name_longest_segment]
367:
368:     # In this process, the name and status of each greenway property (polygons)
369:     # have been edited
370:     edits += 'name\tstatus\t'
371:
372:     # Update the attribute information
373:     attribute_table.updateRow([fid, name_longest_segment, prk_prp_id, 'GREENWAY', 'DEVELOPED', edits])
374:
375:     # Close the search cursor
376:     del attribute_table_gwys
377:
378:     # Close the update cursor
379:     del attribute_table
380:
381:     """
382:
383:     6. SOLVE PROBLEM 3 - A
384:     add missing greenway properties.
385:     """
386:
387:     # First, certain greenway segemnts do not fall within any greenway property. To more
388:     # accurately represent the greenway properties that are developed, a 10 meter buffer
389:     # around each greenway segment that does not fall within a property is geenrated
390:     # and added to the dataset
391:
392:     # Select all greenway segments (lines) that fall within a greenway property
393:     arcpy.MakeFeatureLayer_management(greenways_clean_shp, greenways_clean_lyr)
394:     arcpy.SelectLayerByLocation_management(greenways_clean_lyr, "INTERSECT",
395:     properties_gwys_as_single_shp)
396:
397:     # Add all greenway segments (lines) that or not greenways and not private
398:     query = """trail_type" <> 'Greenway' AND "trail_type" <> 'Private'"""
399:     arcpy.SelectLayerByAttribute_management(greenways_clean_lyr, "ADD_TO_SELECTION", query)
400:
401:     # Although UNCC is labeled as a 'greenway entrance', this trail should be added to the
402:     # dataset
403:     query = """trail_name" = 'UNCC'"""
404:     arcpy.SelectLayerByAttribute_management(greenways_clean_lyr, "REMOVE_FROM_SELECTION", query)
405:
406:     # The selection is now switched (reverted), which represents all segments that
407:     # do not fall within any greenway property
408:     arcpy.SelectLayerByAttribute_management(greenways_clean_lyr, "SWITCH_SELECTION")
409:     arcpy.CopyFeatures_management(greenways_clean_lyr, greenways_w_missing_property_shp)
410:     arcpy.SelectLayerByAttribute_management(greenways_clean_lyr, "CLEAR_SELECTION")
411:
412:     # Generate a shapefile (polygon) representing a 10 Meter buffer around developed
413:     # greenways that do not intersect with any property polygon
414:     arcpy.MakeFeatureLayer_management(greenways_w_missing_property_shp,
415:     greenways_w_missing_property_lyr)

```

clean_park_properties.py

```

414: arcpy.Buffer_analysis(greenways_w_missing_property_lyr,
    properties_gwys_single_developed_missing_shp, '10 Meters', '', 'FLAT')
415:
416: # Insert the generated buffer polygons to the developed greenway properties shapefile
417: properties_gwys = arcpy.da.InsertCursor(properties_gwys_developed_shp, ['SHAPE@', 'park_id',
    'property', 'prk_prp_id', 'park_type', 'status', 'source'])
418: properties_gwys_missing = arcpy.da.SearchCursor(properties_gwys_single_developed_missing_shp,
    ['SHAPE@', 'trail_name'])
419: for shape, name in properties_gwys_missing:
420:     name = name.upper()
421:     park_id = greenway_abbreviations[name][:4]
422:     prk_prp_id = greenway_abbreviations[name]
423:     properties_gwys.insertRow([shape, park_id, name, prk_prp_id, 'GREENWAY', 'DEVELOPED', 'buffer
    around developed greenway'])
424:
425: # Close both search and insert cursors
426: del properties_gwys
427: del properties_gwys_missing
428:
429:
430: """
431:     7. SOLVE PROBLEM 3 - B
432:     add proposed Doby Creek Greenway property.
433: """
434:
435: # The proposed outline for Doby Creek Greenway is available online. Using this
436: # outline, a 10-meter buffer around the outline is generated to represent the greenway
437: # property for Doby Creek Greenway
438:
439: gwys_undeveloped = {
440:     'Doby Creek': 'DBYC',          'Caldwell Station': 'CSTG',
441:     'Cane Creek': 'CCRG',          'Clems Branch': 'CBRG',
442:     'McKee': 'MCKG',              'North Prong Clarke Creek': 'NPCCG',
443:     'Ramah Creek': 'RMCG',        'Rocky River': 'RCKG'}
444:
445: # Select all greenway segments (lines) that fall within a greenway property
446: arcpy.MakeFeatureLayer_management(creeks_shp, creeks_lyr)
447: query = ' OR '.join(["stream" = '%s' % name for name in gwys_undeveloped])
448: arcpy.SelectLayerByAttribute_management(creeks_lyr, "ADD_TO_SELECTION", query)
449:
450: # Generate a shapefile (polygon) representing a 10 Meter buffer around developed
451: # greenways that do not intersect with any property polygon
452: arcpy.MakeFeatureLayer_management(greenways_w_missing_property_shp,
    greenways_w_missing_property_lyr)
453:
454: arcpy.Buffer_analysis(creeks_lyr, properties_creeks_shp, '10 Meters', '', 'FLAT')
455:
456: # Insert the Doby Creek's property (polygon) to the undeveloped greenway properties
457: attribute_table_greenways = arcpy.da.InsertCursor(properties_gwys_undeveloped_shp, ['SHAPE@',
    'property', 'prk_prp_id', 'source'])
458: attribute_table_add_greenways = arcpy.da.SearchCursor(properties_creeks_shp, ['SHAPE@', 'stream'])
459: for shape, creek_name in attribute_table_add_greenways:
460:     name = creek_name + ' Greenway'
461:     prk_prp_id = gwys_undeveloped[creek_name]
462:     source = 'creeks and streams file: maps.co.mecklenburg.nc.us/openmapping'
463:     attribute_table_greenways.insertRow([shape, name, prk_prp_id, source])
464:
465: # Close search and insert cursors
466: del attribute_table_greenways
467: del attribute_table_add_greenways
468:
469: # Update the status field of all undeveloped greenway properties to UNDEVELOPED
470: # (instead of GREENWAY). Since an edit is being made to the original shapefile,

```

clean_park_properties.py

```

469: # the edits attribute is update as well.
470: attribute_table = arcpy.da.UpdateCursor(properties_gwys_undeveloped_shp, ['prk_prp_id','park_type',
    'status', 'edits'])
471: for prk_prp_id, park_type, status, edits in attribute_table:
472:     edits += 'status\t'
473:     attribute_table.updateRow([prk_prp_id + '_UND', 'GREENWAY', 'UNDEVELOPED', edits])
474: del attribute_table
475:
476:
477: """
478:     8. SOLVE PROBLEM 3 - C
479:     adding missing park properties that are clearly developed (Google Maps or Bing Maps).
480: """
481:
482: # Merge all DPR's properties (polygons) back together (greenways and non-greenways)
483: # Additionally, missing properties that are clearly developed (Google Maps or Bing Maps)
484: # and other missing properties that appear in DPR's inventory (Excel file) are added
485: arcpy.Merge_management([properties_non_gwys_shp, properties_gwys_developed_shp,
    properties_gwys_undeveloped_shp, properties_missing_shp], properties_single_merged_shp)
486:
487: # Add an owner attribute to the properties shapefile (polygons)
488: # This attribute provides information on the owner of the property (public/private)
489: arcpy.AddField_management(properties_single_merged_shp, 'owner', 'TEXT')
490:
491: # Update status and owner information for each feature
492: attribute_table = arcpy.da.UpdateCursor(properties_single_merged_shp, ['status','prk_prp_id',
    'source','owner','property','edits'])
493: for status, prk_prp_id, source, owner, name, edits in attribute_table:
494:
495:     # Add owner information for missing properties
496:     if source.isspace():
497:         owner = 'N/A'
498:         source = 'manually added property'
499:     else: owner = 'MECKLENBURG COUNTY'
500:
501:     # Update status
502:     if status == 'MISSING': status = 'DEVELOPED'
503:     if status == 'POSSIBLE SITE': status = 'UNDEVELOPED'
504:     elif status == 'SCHOOL SITE':
505:         status = 'DEVELOPED'
506:         owner = 'SCHOOL'
507:
508:     # Update edits made
509:     if edits.isspace(): edits = 'none'
510:     else: edits = ','.join(edits.strip().split('\t'))
511:
512:     # Add identifiers to missing properties
513:     if name in missing_parks_abbreviations: prk_prp_id = missing_parks_abbreviations[name]
514:
515:     # Update attribute information
516:     attribute_table.updateRow([status, prk_prp_id, source, owner, name, edits])
517:
518: # Close update cursor
519: del attribute_table
520:
521:
522: """
523:     9. SOLVE PROBLEM 3 - D
524:     adding private golf course properties.
525: """
526:

```

clean_park_properties.py

```

527: # Generate a unique list of all golf courses. This list will be used to create unique
528: # identifiers for each golf course.
529: golf_courses = list(set([golf_course for golf_course, in arcpy.da.SearchCursor(golfcourses_shp,
    'course'])))
530:
531: # Insert private gold crouse polygons into the properties shapefile (single-parted)
532: attribute_table_golfcourses = arcpy.da.SearchCursor(golfcourses_shp, ['SHAPE@', 'ownerlastn',
    'course'])
533: attribute_table_parkproperties = arcpy.da.InsertCursor(properties_single_merged_shp, ['SHAPE@',
    'park_type', 'status', 'property', 'prk_prp_id', 'source', 'owner', 'edits'])
534: for shape, owner, name in attribute_table_golfcourses:
535:     if owner != 'MECKLENBURG COUNTY':
536:         park_type = 'GOLF COURSE'
537:         status = 'DEVELOPED'
538:         property_name = name
539:         source = 'golf courses file: maps.co.mecklenburg.nc.us/openmapping'
540:         edits = 'none'
541:         owner = 'PRIVATE'
542:         prk_prp_id = 'GLF_%s' % golf_courses.index(name)
543:         attribute_table_parkproperties.insertRow([shape, park_type, status, property_name,
    prk_prp_id, source, owner, edits])
544:
545: #Close search and insert cursors
546: del attribute_table_golfcourses
547: del attribute_table_parkproperties
548:
549:
550: """
551:     10. SOLVE PROBLEM 4
552:     adding identifiers that link to the park locations shapefile (points) and the
553:     inventory (Excel spreadsheet provided by the DPR).
554: """
555:
556: # Dissolve all polygons back to multi-part polygons instead of single-part polygons
557: # The edits attribute is kept in this shapefile. Thia allows to Lookup which polygons
558: # have been edited.
559: fieldnames = ['park_type', 'status', 'park_id', 'property', 'prk_prp_id', 'source', 'owner', 'edits']
560: arcpy.Dissolve_management(properties_single_merged_shp, properties_clean_edits_shp, fieldnames, '',
    'MULTI_PART')
561:
562: # Add two attributes:
563: # (1) prk_loc_id which represents the unique identifier for the features in the
564: # park locations shapefile (poitns)
565: # (2) invtory_id which represents the unique identifier for each row in DPR's
566: # inventory (Excel file)
567: arcpy.AddField_management(properties_clean_edits_shp, 'prk_loc_id', 'TEXT')
568: arcpy.AddField_management(properties_clean_edits_shp, 'invtory_id', 'TEXT')
569:
570: # Going through each property (polygons) feature-by-feature and updating the
571: # attribute information for both identifiers
572: attribute_table = arcpy.da.UpdateCursor(properties_clean_edits_shp, ['prk_prp_id', 'prk_loc_id',
    'invtory_id'])
573: for prk_prp_id, prk_loc_id, invtory_id in attribute_table:
574:
575:     # The property identifier corresponds to the park location and inventory identifier
576:     # in most cases. Therefore both identifiers are set to the property identifier
577:     prk_loc_id = prk_prp_id
578:     invtory_id = prk_prp_id
579:
580:     # Based on manual inspection of the properties and park locations point shapefile
581:     # some identifiers did not match. Therefore, their matching identifier is

```

clean_park_properties.py

```

582: # specified here.
583: if prk_prp_id == 'AGRC': prk_loc_id = 'ARGC'
584: elif prk_prp_id == 'ELIZ': prk_loc_id = 'ELZA'
585: elif prk_prp_id == 'BREV': prk_loc_id = 'FIRSTB'
586: elif prk_prp_id == 'GRCR': prk_loc_id = 'GRNC'
587:
588: elif prk_prp_id == 'HRNP': prk_loc_id, invtory_id = 'HARP', 'HARP'
589: elif prk_prp_id == 'LTSC': prk_loc_id, invtory_id = 'HIDD', 'HIDD'
590: elif prk_prp_id == 'FBDP': prk_loc_id, invtory_id = 'FLTb', 'FLTb'
591: elif prk_prp_id == 'EVER': prk_loc_id, invtory_id = 'ESTW', 'ESTW'
592: elif prk_prp_id == 'LHTP': prk_loc_id, invtory_id = 'LNCH', 'LNCH'
593: elif prk_prp_id == 'FBNP': prk_loc_id, invtory_id = 'FLNP', 'FLNP'
594: elif prk_prp_id == 'LATH': prk_loc_id, invtory_id = 'LATT', 'LATT'
595:
596: # Based on manual inspection of the properties and DPR's inventory (Excel file)
597: # some identifiers did not match. Therefore, their matching identifier is
598: # specified here.
599: elif prk_prp_id == "CDHP": invtory_id = "INV_3"
600: elif prk_prp_id == "CADI": invtory_id = "INV_4"
601: elif prk_prp_id == "WEPK": invtory_id = "INV_7"
602: elif prk_prp_id == "TMNT": invtory_id = "INV_8"
603: elif prk_prp_id == "WBNP": invtory_id = "INV_9"
604: elif prk_prp_id == "CATW": invtory_id = "USNWC"
605: elif prk_prp_id == "DAVS": invtory_id = "CCNP"
606: elif prk_prp_id == "DAVP": invtory_id = "DAVS"
607: elif prk_prp_id == "WEAC": invtory_id = "WCAP"
608: elif prk_prp_id == "MARS": invtory_id = "MAPK"
609: elif prk_prp_id == "PVLY": invtory_id = "LONG"
610: elif prk_prp_id == "MATM": invtory_id = "MINT"
611: elif prk_prp_id == "WPRC": invtory_id = "POBC"
612: elif prk_prp_id == "PALI": invtory_id = "PAPK"
613: elif prk_prp_id == "PAWC_UND": invtory_id = "PAWCNW"
614: elif prk_prp_id == "TBYC_UND": invtory_id = "TBYC_P2"
615: elif prk_prp_id == "MLKG_A": invtory_id = "MLKGN2"
616: elif prk_prp_id == "MLKG_U": invtory_id = "MLKGN3"
617: elif prk_prp_id == "MLKG_UND": invtory_id = "MLKGN1"
618: elif prk_prp_id == "BRCG_A": invtory_id = "BRCGC1"
619: elif prk_prp_id == "BRCG_C": invtory_id = "BRCGC2"
620: elif prk_prp_id == "BRCG_UND": invtory_id = "BRCGS"
621: elif prk_prp_id == "MCAG_M": invtory_id = "MCAGEA"
622: elif prk_prp_id == "MCAG_J": invtory_id = "MCAGS1"
623: elif prk_prp_id == "MCAG_UND": invtory_id = "MCAGS2"
624: elif prk_prp_id == "IRWG_S": invtory_id = "IRWGC1"
625: elif prk_prp_id == "IRWG_C": invtory_id = "IRWGC2"
626: elif prk_prp_id == "IRWG_UND": invtory_id = "IRWGC3"
627: elif prk_prp_id == "MCMG": invtory_id = "MCMGS1"
628: elif prk_prp_id == "MCMG_UND": invtory_id = "MCMGS2"
629: elif prk_prp_id == "FMCg_M": invtory_id = "FMCgS1"
630: elif prk_prp_id == "FMCg_J": invtory_id = "FMCgS2"
631: elif prk_prp_id == "FMCg_UND": invtory_id = "FMCgS3"
632: elif prk_prp_id == "MCDG": invtory_id = "MCDGNO"
633: elif prk_prp_id == "MCDG_UND": invtory_id = "MCDGNW"
634: elif prk_prp_id == "LTSCG_E": invtory_id = "LTSCC2"
635: elif prk_prp_id == "LTSCG_C": invtory_id = "LTSCC3"
636: elif prk_prp_id == "LTSCG_R": invtory_id = "LTSCS1"
637: elif prk_prp_id == "LTSCG_UND": invtory_id = "LTSCS2"
638: elif prk_prp_id == "LGCG_UND": invtory_id = "LGCGN1"
639: elif prk_prp_id == "POLK_UND": invtory_id = "PODG"
640: elif prk_prp_id == "CLRK_UND": invtory_id = "CLRKN1"
641: elif prk_prp_id == "CLTR_UND": invtory_id = "CLRKN2"
642: elif prk_prp_id == "WABG": invtory_id = "WLKGS1"

```

clean_park_properties.py

```

643:     elif prk_prp_id == "WABG_UND": invtory_id = "WLKGS2"
644:     elif prk_prp_id == "RECG_UND": invtory_id = "RECGEA"
645:     elif prk_prp_id == "DIBG_UND": invtory_id = "DXBR"
646:     elif prk_prp_id == "STWC": invtory_id = "STWCC1"
647:     elif prk_prp_id == "STWC_UND": invtory_id = "STWCC2"
648:     elif prk_prp_id == "TRCK_UND": invtory_id = "TTCG2"
649:     elif prk_prp_id == "SCCG_UND": invtory_id = "SPCG"
650:     elif prk_prp_id == "WBRR": invtory_id = "WBRRG"
651:     elif prk_prp_id == "EDBG_UND": invtory_id = "EDBGC"
652:     elif prk_prp_id == "SMCG_UND": invtory_id = "SMCGS1"
653:     elif prk_prp_id == "SMCG": invtory_id = "SMCGS2"
654:     elif prk_prp_id == "SOPG": invtory_id = "SPGN"
655:     elif prk_prp_id == "MCDP": invtory_id = "MCDB"
656:     elif prk_prp_id == "STNY_UND": invtory_id = "STNY"
657:     elif prk_prp_id == "BKCG_UND": invtory_id = "BKCG"
658:     elif prk_prp_id == "BRCG_UND": invtory_id = "BRCGS"
659:     elif prk_prp_id == "CFCG_UND": invtory_id = "CFCG"
660:     elif prk_prp_id == "CMGC_UND": invtory_id = "CMGC"
661:     elif prk_prp_id == "DBYC_UND": invtory_id = "DBYC"
662:     elif prk_prp_id == "FBRG_UND": invtory_id = "FBRG"
663:     elif prk_prp_id == "GUMC_UND": invtory_id = "GUMC"
664:     elif prk_prp_id == "IRVN_UND": invtory_id = "IRVN"
665:     elif prk_prp_id == "KGSW_UND": invtory_id = "KGSW"
666:     elif prk_prp_id == "MCCG_UND": invtory_id = "MCCG"
667:     elif prk_prp_id == "STCG_UND": invtory_id = "STCG"
668:     elif prk_prp_id == "MISS_4": invtory_id = "SCES"
669:     elif prk_prp_id == "MISS_7": invtory_id = "MINT"
670:     elif prk_prp_id == "SUCG_UND": invtory_id = "BSGC"
671:     elif prk_prp_id == "DBYC_UND": invtory_id = "DBYC"
672:     elif prk_prp_id == "CSTG_UND": invtory_id = "CSTG"
673:     elif prk_prp_id == "CCRG_UND": invtory_id = "CCRG"
674:     elif prk_prp_id == "CBRG_UND": invtory_id = "CBRG"
675:     elif prk_prp_id == "MCKG_UND": invtory_id = "MCKG"
676:     elif prk_prp_id == "NPCCG_UND": invtory_id = "NPCCG"
677:     elif prk_prp_id == "RMCG_UND": invtory_id = "RMCG"
678:     elif prk_prp_id == "RCKG_UND": invtory_id = "RCKG"
679:
680:     # Update the attribute information
681:     attribute_table.updateRow([prk_prp_id, prk_loc_id, invtory_id])
682:
683: # Close the update cursor
684: del attribute_table
685:
686: # Dissolve all polygons without the "edits" attribute. The "prk_prp_id" is now unique
687: fieldnames = ['park_type', 'status', 'park_id', 'property', 'prk_prp_id', 'source', 'owner',
688:               'prk_loc_id', 'invtory_id']
689: arcpy.Dissolve_management(properties_clean_edits_shp, properties_clean_shp, fieldnames, '',
690:                            "MULTI_PART")

```

6. Extract Additional Parks and Recreational Facilities

extract_additional_park_and_recreation_properties.py

```

1: # Author:                               Coline C. Dony
2: # Date first written:                   May 8, 2016
3: # Date last updated:                   May 28, 2016
4: # Purpose:                             Extract additional park and recreational properties
5: #
6: # Problem statement:
7: # Mecklenburg County's Department of Park and Recreation's (DPR) property boundaries are
8: # publically available. A shapefile of these properties can be downloaded at:
9: # http://maps.co.mecklenburg.nc.us/openmapping/data.html
10: # This shapefile contains all properties managed by DPR and is useful for my research.
11: #
12: # However it does not represent a full dataset of park and recreational properties
13: # in Mecklenburg County. This script uses commercial impervious surfaces, tax parcel data
14: # and Open Street Map (OSM) data to complement the DPR's inventory of properties.
15: #
16: # A shapefile of tax parcels can be downloaded at:
17: # http://maps.co.mecklenburg.nc.us/openmapping/data.html
18: # OpenStreetMap data was downloaded at:
19: # https://mapzen.com/data/metro-extracts/#charlotte-north-carolina (OSM2PGSQL SHP)
20: #
21: # Problem - 1: Find park and recreational locations of the OpenStreetMap data source.
22: # Solution:    Using the description of each polygon in the OSM data, features that
23: #              are associated with Leisure are extracted.
24: #
25: # Problem - 2: Find impervious surfaces that coincide with Leisure activities.
26: # Solution:    Using a shapefile (polygons) that contains all commercial impervious
27: #              surfaces and residential tax parcels, these surfaces are distinguished
28: #              from non-Leisure impervious surfaces.
29: #
30: # Problem - 3: Merge both datasets that coincide with Leisure together. Group features
31: #              that belong together (e.g. an HOA park with a tennis court and pool)
32: #              and also remove properties that do not coincide with Leisure activity
33: # Solution:    Group features that are within the same tax parcel together. Based
34: #              on the tax parcel information, features are labeled (e.g. Residential
35: #              Commercial, Treatment plant, etc.) and properties that do not coincide
36: #              with Leisure activities are removed from the dataset
37: #
38: # Problem - 4: Remove features that fall within properties that are in the DPR's inventory,
39: #              to remove duplicate recreational properties.
40: # Solution:    Features (polygons) that fall within DPR's properties (polygons)
41: #              are removed from the dataset.
42:
43:
44: """
45:     1. INPUT FILES
46:     original, already existng files, used in this script
47: """
48:
49: # ALL cadastral tax parcel boundaries (polygons) from the planning department
50: # Source: http://maps.co.mecklenburg.nc.us/openmapping/data.html
51: parcel_taxdata_original_shp = r'planning\parcel_taxdata\Parcel_TaxData.shp'
52:
53: # ALL Open Street Map boundaries (polygons) from Mapzen
54: # Source: https://mapzen.com/data/metro-extracts/#charlotte-north-carolina
55: # Download: OSM2PGSQL SHP
56: osm_polygons_original_shp = r'OpenStreetMap\charlotte_north-carolina.osm2pgsql-
57: shapefiles\charlotte_north-carolina_osm_polygon.shp'
58:
59: # Mecklenburg County boundary
60: # Source: http://maps.co.mecklenburg.nc.us/openmapping/data.html
61: mecklenburg_boundary_shp =
62: r'census\counties\Mecklenburg_county\mecklenburgcounty_boundary\MecklenburgCounty_Boundary.shp'

```

 extract_additional_park_and_recreation_properties.py

```

61:
62: # All impervious commercial surfaces in Mecklenburg County
63: # Source: http://maps.co.mecklenburg.nc.us/openmapping/data.html
64: impervious_commercial_original_shp = r'planning\commercial_impervious\Commercial_Impervious.shp'
65:
66: # All park properties (cleaned)
67: # Source: http://maps.co.mecklenburg.nc.us/openmapping/data.html
68: # Edits: with the python script clean_park_properties.py
69: park_properties_clean_original_shp = r'parks\parkproperty\ParkProperty_clean.shp'
70:
71:
72: """
73:     2. TEMPORARY FILES
74:     files that are generated in this script but are not the end product / output file.
75:     This segment also presents the thought process of the cleaning and editing process.
76: """
77:
78: # Copy of shapefile (polygons) that contains all Open Street Map (OSM) boundaries
79: osm_polygons_shp = r'temp\charlotte_north_carolina_osm_polygon.shp'
80:
81: # Shapefile (polygons) that contains OSM polygons associated with Leisure only
82: osm_polygons_leisure_shp = r'temp\osm_polygon_leisure.shp'
83:
84: # Copy of shapefile (polygons) that contains commercial impervious surfaces
85: impervious_commercial_shp = r'temp\Commercial_Impervious.shp'
86:
87: # Shapefile (polygons) that contains commercial impervious surfaces without a Label
88: impervious_other_shp = r'temp\Commercial_Impervious_other.shp'
89:
90: # Copy of shapefile (polygons) that contains park properties (cleaned dataset)
91: park_properties_clean_shp = r'temp\ParkProperty_clean.shp'
92:
93: # Shapefile (polygons) that contains commercial impervious surfaces without a Label
94: # and that do not fall within a feature in the OSM dataset (to get rid of duplicates)
95: # NOTE: Some features in the OSM dataset are labeled with the amenity they represent. Therefore,
96: # the OSM data is favored over the impervious surfaces dataset.
97: impervious_other_non_OSM_leisure_shp = r'temp\Commercial_Impervious_other_non_osm_leisure.shp'
98:
99: # Shapefile (polygons) that contains commercial impervious surfaces without a Label
100: # and OSM features that coincide with Leisure activities
101: merged_OSM_and_impervious_shp = r'temp\merged_impervious_and_osm_properties.shp'
102:
103: # Shapefile (polygons) that contains commercial impervious surfaces without a Label
104: # and OSM features that coincide with Leisure activities, to which all attributes of
105: # the tax parcel dataset are joined
106: merged_OSM_and_impervious_w_parcel_info_shp =
    r'temp\merged_impervious_and_osm_properties_w_parcel_info.shp'
107:
108: # Shapefile (polygons) that contains commercial impervious surfaces without a Label
109: # and OSM features that coincide with Leisure activities, where features are grouped
110: # if they fall within the same tax parcel
111: # NOTE: The "owner" attribute is generated to distinguish different types of parcel owners.
112: # Based on the tax parcel attributes, groups of features are labeled "Commercial",
113: # "Residential", "Church", "School", etc. This way, a group of features that falls under
114: # recreational activities can be distinguished from those that do not (e.g. treatment plant,
115: # Hotel pool or tennis court, etc.)
116: merged_OSM_and_impervious_dissolved_shp = r'temp\merged_impervious_and_osm_properties_dissolved.shp'
117:
118: # Shapefile (polygons) that contains commercial impervious surfaces without a Label
119: # and OSM features that coincide with Leisure activities, where features are grouped
120: # if they fall within the same tax parcel and that do not fall within a feature of

```

 extract_additional_park_and_recreation_properties.py

```

121: # the DPR's park property dataset (to get rid of duplicates)
122: additional_properies_non_DPR_shp = r'temp\merged_impervious_and_osm_properties_non_DPR.shp'
123:
124: # For certain functions, some shapefiles will be converted to layer files
125: osm_polygons_lyr = osm_polygons_shp[:-4]
126: osm_polygons_leisure_lyr = osm_polygons_leisure_shp[:-4]
127: impervious_commercial_lyr = impervious_commercial_shp[:-4]
128: impervious_other_lyr = impervious_other_shp[:-4]
129: merged_OSM_and_impervious_lyr = merged_OSM_and_impervious_shp[:-4]
130: parcel_taxdata_original_lyr = parcel_taxdata_original_shp[:-4]
131: merged_OSM_and_impervious_w_parcel_info_lyr = merged_OSM_and_impervious_w_parcel_info_shp[:-4]
132: merged_OSM_and_impervious_dissolved_lyr = merged_OSM_and_impervious_dissolved_shp[:-4]
133: additional_properies_non_DPR_lyr = additional_properies_non_DPR_shp[:-4]
134: park_properties_clean_lyr = park_properties_clean_shp[:-4]
135:
136:
137: """
138:     3. OUTPUT FILE
139:     additional park and recreational properties from tax parcel data and OSM data
140: """
141:
142: # Shapefile (polygons) that contains commercial impervious surfaces without a label
143: # and OSM features that coincide with leisure activities, where features are grouped
144: # if they fall within the same tax parcel and that do not fall within a feature of
145: # the DPR's park property dataset (to get rid of duplicates) and that have an "owner"
146: # label that coincides with accessible recreational activities.
147: additional_properies_shp = r'parks\parkproperty\Additional_ParkProperty.shp'
148:
149:
150: """
151:     4. SOLVE PROBLEM 1
152:     extract OSM leisure polygons
153: """
154:
155: # Project the shapefile (polygons) that contains all Open Street Map (OSM) boundaries
156: # NOTE: OSM data is usually in WGS84 whereas all shapefiles for this study are projected
157: # to the North Carolina State Plane Projected Coordinate System
158: arcpy.Project_management(osm_polygons_original_shp, osm_polygons_shp, sr_nc)
159:
160: # Extract features (polygons) from the OSM data that are associated with Leisure
161: arcpy.MakeFeatureLayer_management(osm_polygons_shp, osm_polygons_lyr)
162: query = """"leisure" <> ' ' AND "leisure" <> 'stadium'""""
163: arcpy.SelectLayerByAttribute_management(osm_polygons_lyr, 'NEW_SELECTION', query)
164:
165: # Extract only those features (polygons) that fall within the boundary of
166: # Mecklenburg County (polygon)
167: arcpy.SelectLayerByLocation_management(osm_polygons_lyr, 'HAVE_THEIR_CENTER_IN',
    mecklenburg_boundary_shp, '', 'SUBSET_SELECTION')
168: arcpy.CopyFeatures_management(osm_polygons_lyr, osm_polygons_leisure_shp)
169:
170:
171: """
172:     5. SOLVE PROBLEM 2
173:     extract impervious surfaces that are not labeled and do not fall within OSM features
174: """
175:
176: # Make a copy of the original shapefile (polygons) that contains impervious
177: # surfaces that are commercial
178: arcpy.CopyFeatures_management(impervious_commercial_original_shp, impervious_commercial_shp)
179:
180: # Extracting features (polygons) that are labeled "other". These surfaces coioncide

```

extract_additional_park_and_recreation_properties.py

```

181: # with paved recreational activities (e.g. playgrounds, basketball courts, etc.) but also
182: # with other paved areas (e.g. water sanitation)
183: # NOTE: The features that do not coincide with recreational surfaces (e.g. treatment plants)
184: # are be discarded from the dataset when joining tax parcel information (see solve problem 3)
185: arcpy.MakeFeatureLayer_management(impervious_commercial_shp, impervious_commercial_lyr)
186: query = """"subtheme" = 'Other'""""
187: arcpy.SelectLayerByAttribute_management(impervious_commercial_lyr, 'NEW_SELECTION', query)
188: arcpy.CopyFeatures_management(impervious_commercial_lyr, impervious_other_shp)
189:
190: # Extract Shapefile (polygons) that contains commercial impervious surfaces without a label
191: # and that do not fall within a feature in the OSM dataset (to get rid of duplicates)
192: # NOTE: Features in the OSM dataset are labeled with the amenity they represent.
193: # (e.g. pitch, playground, etc.) Therefore, the OSM data is favored over the
194: # impervious surfaces dataset.
195: arcpy.MakeFeatureLayer_management(impervious_other_shp, impervious_other_lyr)
196: arcpy.SelectLayerByLocation_management(impervious_other_lyr, 'INTERSECT', osm_polygons_leisure_shp)
197: arcpy.SelectLayerByAttribute_management(impervious_other_lyr, 'SWITCH_SELECTION')
198: arcpy.CopyFeatures_management(impervious_other_lyr, impervious_other_non_OSM_leisure_shp)
199:
200:
201: """
202:     6. SOLVE PROBLEM 3
203:     merge both datasets together and group features if they fall within the same tax parcel
204:     while labeling the owner type of each group of features (e.g. Hotel/Motel, Church, etc.)
205: """
206:
207: # First, both datasets are merged together
208:
209: # Generate an empty shapefile (polygons) in which all features associated with potential
210: # recreational activities (OSM and impervious surfaces) will be inserted
211: arcpy.CreateFeatureclass_management(workfolder, merged_OSM_and_impervious_shp, 'POLYGON', '', '',
    '', sr_nc)
212:
213: # Add three empty attributes to the empty shapefile (polygons)
214: # NOTE: the source of each feature (OSM or impervious) appears in the source attribute
215: # and the Leisure Label that OSM gives to each feature is in the OSM_Label attribute
216: osm_fields = ['name', 'source', 'OSM_Label']
217: for fieldname in osm_fields:
218:     arcpy.AddField_management(merged_OSM_and_impervious_shp, fieldname, 'TEXT')
219:
220: # Create a cursor to insert all features in the empty shapefile
221: attribute_table_new_shapefile = arcpy.da.InsertCursor(merged_OSM_and_impervious_shp, ['SHAPE@'] +
    osm_fields)
222:
223: # Going through each feature of the shapefile (polygons) that contains OSM features
224: # associated with leisure activity and add each feature to the new shapefile
225: attribute_table = arcpy.da.SearchCursor(osm_polygons_leisure_shp, ['SHAPE@', 'name', 'leisure'])
226: for shape, name, leisure in attribute_table:
227:     attribute_table_new_shapefile.insertRow([shape, name, 'OpenStreetMap', leisure])
228: del attribute_table
229:
230: # Going through each feature of the shapefile (polygons) that contains impervious
231: # surfaces potentially associated with recreational activity and add each feature
232: # to the new shapefile
233: attribute_table = arcpy.da.SearchCursor(impervious_other_non_OSM_leisure_shp, ['SHAPE@',
    'ldplanname'])
234: for shape, name in attribute_table:
235:     attribute_table_new_shapefile.insertRow([shape, name, 'commercial impervious file:
    maps.co.mecklenburg.nc.us/openmapping', ''])
236: del attribute_table
237:

```

extract_additional_park_and_recreation_properties.py

```

238: # Close insert cursor
239: del attribute_table_new_shapefile
240:
241:
242: # Second, tax parcel attributes are joined to each feature in the new shapefile
243:
244: # Join attributes of the tax parcel each feature falls into
245: arcpy.MakeFeatureLayer_management(merged_OSM_and_impervious_shp, merged_OSM_and_impervious_lyr)
246: arcpy.MakeFeatureLayer_management(parcel_taxdata_original_shp, parcel_taxdata_original_lyr)
247: arcpy.SpatialJoin_analysis(merged_OSM_and_impervious_lyr, parcel_taxdata_original_lyr,
    merged_OSM_and_impervious_w_parcel_info_shp, 'JOIN_ONE_TO_ONE')
248:
249:
250: # Third, using the information from the tax parcels to generate a field that can distinguish
251: # different owner types
252:
253: # Copy attribute information from all features into a list
254: # NOTE: It is faster to clean the dataset in python rather than to use cursors
255: parcel_fields = ['FID', 'pid', 'name', 'cdebuildin', 'descbuildi', 'descproper', 'ownerfirst',
    'ownerlastn']
256: parcel_info = [parcel_info for parcel_info in
    arcpy.da.SearchCursor(merged_OSM_and_impervious_w_parcel_info_shp, parcel_fields)]
257:
258: # Create an empty dictionary in which each record will contain the list of features that
259: # need to be grouped together. This dictionary will dictate how to dissolve all the
260: # features together.
261: dissolve = {}
262: dict_id = 0
263:
264: # Create an empty dictionary in which each record will contain a unique tax parcel
265: # name (or owner). This will make it possible to look up features that fall within
266: # different parcels, but that have the same name (these parcels are mostly adjacent
267: # to one another but have been built in different years)
268: unique_names = {}
269:
270: # Create an empty dictionary in which each record will contain a unique parcel id
271: # This will make it possible to look up parcel features that fall within different parcels,
272: # that have the same parcel identifier (and thus belong together)
273: unique_pids = {}
274:
275: residential_codes = ['%02d' % code for code in [1,2,4,6,9,60,61,62,63,64,71,72,74,83,84]]
276: words_residential = ['apartments', 'apartment homes', 'condo', 'townhomes', 'villas', 'hoa',
    'townes', 'town homes', 'townhome', 'homeowners']
277:
278: words_other = ['daycare', 'treatment plant', 'club', 'mhp', 'nursery', 'greenhouse', 'maintenance',
    'library', 'fountains', 'cpcc', 'ymca', 'development center']
279: words_religious = ['church', 'presby', 'islamic', 'evangelistic', 'baptist']
280: words_school = ['school', 'college', 'education', 'academy', 'c univ', 'queens university', 'unc-
    charlotte']
281: words_hotel = ['hotel', 'inn', 'hilton']
282: words_commercial = ['concrete', 'corporation', 'service', 'automotive', 'energy', 'refinery',
    'utiliy', 'drilling', 'brewery', 'corp', 'construction', 'plantation utility', 'commerce',
    'atherton', 'reventure', 'innovation', 'sardis road park', 'wcf']
283:
284: authorities = ['nc dot', 'city of charlotte', 'mecklenburg county', 'board of education']
285: companies = ['coca-cola', 'wal-mart', 'microsoft', 'citgo', 'econolodge', 'red ventures', 'horizons
    computer learning']
286:
287: for fid, pid, name, blding_code, blding_desc, property_desc, firstname, lastname in parcel_info:
288:
289:     # If a name is empty, replace it with the name of owners

```

 extract_additional_park_and_recreation_properties.py

```

290:     if name.isspace(): name = ' '.join([lastname, firstname])
291:     # Clean name and make all letters lowercase (makes it easier to match words)
292:     name = name.lower()
293:     name = name.replace(' inc', '')
294:     name = name.replace(' llc', '')
295:     if name == 'charlotte latin school': name += 's'
296:     name = name.strip()
297:
298:     # Fill property type with the property description
299:     prperty_type = prperty_desc
300:
301:     # Since some tax parcels do not have a property description, this part of the
302:     # script labels the property type based on words that appear in the name of
303:     # each feature
304:     if blding_code in residential_codes: prperty_type = 'Residential'
305:     if any(word in name for word in words_other + authorities): prperty_type = 'Other'
306:     if any(word in name for word in words_religious): prperty_type = 'Religious Center'
307:     if any(word in prperty_type for word in words_school): prperty_type = 'School'
308:     if any(word in name for word in words_school): prperty_type = 'School'
309:     if any(word in name for word in words_residential): prperty_type = 'Residential'
310:     if name.endswith(' park'): prperty_type = 'Park'
311:     if 'memorial' in name: prperty_type = 'Memorial Park'
312:     if any(word in name for word in words_hotel): prperty_type = 'Hotel/Motel'
313:     if any(word in name for word in words_commercial + companies): prperty_type = 'Commercial'
314:     if name.endswith('farm'): prperty_type = 'Commercial'
315:
316:     # Arrange all features in the dissolve dictionary based on their parcel identifier
317:     # and their parcel name
318:
319:     # If the parcel id and name do not appear in their respective dictionary
320:     # add them to it and to the dissolve dictionary
321:     if pid not in unique_pids and name not in unique_names:
322:         unique_pids[pid] = dict_id
323:         if not any(word in name for word in authorities): unique_names[name] = dict_id
324:         dissolve[dict_id] = [[pid], [fid], [prperty_type], [name]]
325:         dict_id += 1
326:
327:     # Otherwise, if the pid and/or name already appears in one of the respective
328:     # dictionaries, that means this group of features need to be added to an already
329:     # existng record in the dissolve dictionary
330:     else:
331:
332:         # If both, the parcel id and name have a record
333:         if pid in unique_pids and name in unique_names:
334:             did = unique_pids[pid]
335:             did_name = unique_names[name]
336:
337:             # In case they do not have the same identifier in the dissolve dictionary
338:             # both records are merged to one
339:             if did != did_name:
340:                 pids, fids, owners, names = dissolve[did_name]
341:                 dissolve[did][0] += pids
342:                 dissolve[did][1] += fids
343:                 dissolve[did][2] += owners
344:                 dissolve[did][3] += names
345:                 unique_names[name] = did
346:                 for pid_to_change in pids: unique_pids[pid_to_change] = did
347:                 del dissolve[did_name]
348:
349:             # Else, if the parcel id already exists but not the name
350:             elif pid in unique_pids:

```


extract_additional_park_and_recreation_properties.py

```

351:         did = unique_pids[pid]
352:         if not any(word in name for word in authorities): unique_names[name] = did
353:
354:         # Finally, if the name already exists but not the parcel id
355:         elif name in unique_names:
356:             did = unique_names[name]
357:             unique_pids[pid] = did
358:             dissolve[did][0] += [pid]
359:             dissolve[did][1] += [fid]
360:             dissolve[did][2] += [property_type]
361:             dissolve[did][3] += [name]
362:
363: # Clean the dissolve dictionary so that the information can easily be added to the
364: # shapefile in the next step
365: for did in dissolve:
366:     pids, fids, owners, names = dissolve[did]
367:
368:     # Convert the list of parcel identifiers to a string
369:     dissolve[did][0] = ','.join(pids)
370:
371:     # Pick the property type that is most common in the list
372:     if all(property_type.isspace() for property_type in owners): property_type = 'N/A'
373:     else: property_type = max(((item, owners.count(item)) for item in set(owners) if not
item.isspace()), key=lambda a: a[1])[0]
374:     dissolve[did][2] = property_type
375:
376:     # Choose best name for the group of features based on the names in the list
377:     best_name = ''
378:     for name in names:
379:         if not any(word in name for word in authorities): best_name = name
380:     dissolve[did][3] = names[0] if best_name == '' else best_name
381:
382:
383: # Fourth, Based on the dictionary and the shapefile (polygons) that contains OSM and
384: # potential impervious surfaces associated with recreational activities, a new shapefile
385: # is generated that also contains the property type of each group of features
386:
387: # Generate an empty shapefile (polygons) in which all features associated with potential
388: # recreational activities (OSM and impervious surfaces) will be inserted and grouped
389: # together based on their tax parcel information (dissolve dictionary)
390: arcpy.CreateFeatureclass_management(workfolder, merged_OSM_and_impervious_dissolved_shp, "POLYGON",
"", "", "", sr_nc)
391:
392: # Add five empty fields.
393: fields = ['name', 'source', 'pid', 'yearbuilt', 'prpty_type']
394: for fieldname in fields:
395:     arcpy.AddField_management(merged_OSM_and_impervious_dissolved_shp, fieldname, 'TEXT')
396:
397: # Add six empty fields that represent the different Leisure activities OSM keeps track of
398: # NOTE: The third field from the OSM dataset, the "OSM_Label" attribute,
399: # contains the Leisure type of the feature. When merging features that fall within
400: # the same tax parcel together, this attribute is used to fill the count of each
401: # Leisure type within the parcel (similar to DPR's inventory (Excel file)
402: leisure_fields = ['pitch', 'playground', 'multipurp', 'rec_fitnes', 'pool', 'tennis']
403: for fieldname in leisure_fields:
404:     arcpy.AddField_management(merged_OSM_and_impervious_dissolved_shp, fieldname, 'SHORT')
405:
406: # Creating a cursor to insert group of features in the empty shapefile
407: attribute_table = arcpy.da.InsertCursor(merged_OSM_and_impervious_dissolved_shp, fields +
leisure_fields + ['SHAPE@'])
408:

```

extract_additional_park_and_recreation_properties.py

```

409: # Going through each record in the dissolve dictionary
410: for did in dissolve:
411:
412:     # Unpack all values in the dictionary record
413:     pids, fids, prperty_type, unique_name = dissolve[did]
414:
415:     # Select all features that share the same parcel id or parcel name
416:     arcpy.MakeFeatureLayer_management(merged_OSM_and_impervious_w_parcel_info_shp,
merged_OSM_and_impervious_w_parcel_info_lyr)
417:     query = ' OR '.join(["FID" = %s" % fid for fid in fids])
418:     features = arcpy.SelectLayerByAttribute_management(merged_OSM_and_impervious_w_parcel_info_lyr,
'NEW_SELECTION', query)
419:
420:     # Initialize the attribute information for the leisure fields
421:     pitch, playground, multipurp, rec_fitness, pool, tennis = 0, 0, 0, 0, 0, 0
422:
423:     # Inititalize lists of sources, years and pollygon geometries
424:     sources = []
425:     years = []
426:     shapes = []
427:
428:     # Going through all features (polygons) that were selected and track the amenities
429:     # they contain, their source (OSM ro impervious) and their geometry
430:     attribute_table_selection = arcpy.da.SearchCursor(features, fields[:-1] + [osm_fields[-1],
'SHAPE@'])
431:     for name, source, pid, year, leisure, shape in attribute_table_selection:
432:
433:         # Clean and make all Letters Lowercase (easier for matching words)
434:         if name.isspace(): name = unique_name
435:         name = name.lower()
436:
437:         # Keep track of the sources (OSM and impervious) the features come from
438:         if source not in sources: sources += [source]
439:
440:         # Keep track of the parcel years
441:         if year not in years: years += [year]
442:
443:         # Fill Leisure field information if available from the OSM_Label field
444:         if leisure == "pitch": pitch += 1
445:         elif leisure == "playground": playground += 1
446:         elif leisure in ["recreation_ground", "track"]: multipurp += 1
447:         elif leisure == "sports_centre": rec_fitness += 1
448:         elif leisure == "swimming_pool": pool += 1
449:         # Fill Leisure field information if available from the tax parcel name
450:         if any(word in name for word in ['pool', 'swim']): pool = 1 if pool == 0 else pool
451:         if any(word in name for word in ['tennis', 'racquet']): tennis = 1 if tennis == 0 else
tennis
452:
453:         # Add geometry to list
454:         shapes += [shape]
455:
456:     # Close search cursor
457:     del attribute_table_selection
458:
459:     # Clean name before inserting into shapefile
460:     name = unique_name.strip().title()
461:
462:     # Create a multi-part polygon
463:     multipolygon = arcpy.Polygon(arcpy.Array([shape.getPart(0) for shape in shapes]))
464:
465:     # Insert polygon and attribute information in the shapefile

```

extract_additional_park_and_recreation_properties.py

```

466:     attribute_table.insertRow([name, ' '.join(sorted(sources)), pids, min(years), prperty_type,
467: pitch, playground, multipurp, rec_fitness, pool, tennis, multipolygon])
468: # Close insert cursor
469: del attribute_table
470:
471:
472: """
473:     7. SOLVE PROBLEM 4
474:     remove features that fall within DPR's porperties
475: """
476:
477: # Make a copy of the cleaned shapefile (polygons) that contains DPR's properties
478: arcpy.CopyFeatures_management(park_properties_clean_original_shp, park_properties_clean_shp)
479:
480: # Generate shapefile (polygons) that contains all features potentially associated with
481: # recreational activities and that do not fall within polygons of the DPR's properties
482: arcpy.MakeFeatureLayer_management(merged_OSM_and_impervious_dissolved_shp,
merged_OSM_and_impervious_dissolved_lyr)
483: arcpy.SelectLayerByLocation_management(merged_OSM_and_impervious_dissolved_lyr, 'INTERSECT',
park_properties_clean_shp)
484: arcpy.SelectLayerByAttribute_management(merged_OSM_and_impervious_dissolved_lyr, 'SWITCH_SELECTION')
485: arcpy.CopyFeatures_management(merged_OSM_and_impervious_dissolved_lyr,
additional_properies_non_DPR_shp)
486:
487:
488: """
489:     8. SOLVE PROBLEM 1
490:     extract only features associated with recreational activities
491: """
492:
493: arcpy.MakeFeatureLayer_management(additional_properies_non_DPR_shp,
additional_properies_non_DPR_lyr)
494: recreational_properties = ['Residential', 'Church', 'Park']
495: query = ' OR '.join(["prpty_type" = '%s' % recreational for recreational in
recreational_properties])
496: arcpy.SelectLayerByAttribute_management(additional_properies_non_DPR_lyr, 'NEW_SELECTION', query)
497: arcpy.CopyFeatures_management(additional_properies_non_DPR_lyr, additional_properies_shp)

```


7. Link Data Together

link_data_together.py

```

1: # Author:                               Coline C. Dony
2: # Date first written:                   June 3, 2016
3: # Date last updated:                   June 16, 2016
4: # Purpose:                             Link all files together
5:
6: # Problem statement:
7: # Mecklenburg County's Department of Park and Recreation's (DPR) property boundaries
8: # are publically available. Additionally, locations of park entrances are publically
9: # available as well. These shapefiles contain data useful for my research and can be
10: # downloaded at: http://maps.co.mecklenburg.nc.us/openmapping/data.html
11: # Michael Kirschman, Deputy Director of Mecklenburg County's DPR shared with me
12: # an Excel spreadsheet containing all parks in their inventory. In this inventory,
13: # the amenities available at each park are provided as well, which is useful for
14: # my research.
15:
16: # The information in these two shapefiles and spreadsheet can not be
17: # easily matched. Although some unique identifiers in those three files do match,
18: # some identifiers are completely different in all three files even though they
19: # correspond to the same park or recreational service. For my dissertation, I
20: # want to generate a dataset of parks and recreational facilities that is as
21: # complete as possible. For that, I start from the cleaned park properties shapefile
22: # (in which I added private golf courses and greenway properties that are missing
23: # (see clean_park_properties.py). Additionally, I generated a unique identifier
24: # for each park property and added two additional identifiers (foreign keys) that
25: # correspond to the unique identifiers in the park locations shapefile (points) and
26: # the inventory (Excel file) respectively. Using these primary and foreign keys,
27: # I generate a dictionary with all the information from these three source files
28: # structured in a way useful to my dissertation.
29:
30:
31: """
32:     1. INPUT FILES
33:     original, already existng files, used in this script
34: """
35:
36: # All properties (polygons) of the Department of Park and Recreation (DPR) - whether
37: # developed or undeveloped. A python script is required to run before this shapefile
38: # is available
39: park_properties_clean_original_shp = r'parks\parkproperty\ParkProperty_clean.shp'
40: park_properties_clean_shp = r'temp\ParkProperty_clean.shp'
41:
42: # All park and greenway entrances (points)
43: # A python script is required to run before this shapefile is available
44: park_locations_clean_original_shp = r'parks\park_locations\Park_Locations_clean.shp'
45:
46: # All parks listed in the inventory of the Department of Park and Recreation
47: # Source: Michael Kirschman, Deputy Director of Mecklenburg County's DPR shared it
48: # with me
49: park_inventory_txt = "parks\Parks and Amenities updated June2014_Inventory.txt"
50:
51:
52: """
53:     2. OUTPUT DICTIONARY
54:     dictionary containing organized park data from the property shapefile (polygons),
55:     the locations shapefile (points) and the inventory (excel file)
56: """
57:
58: # The output of this script is the populated dictionary in which the park and
59: # recreational facilities that appear in the park property shapefile (polygons),
60: # park locations shapefile (points) and inventory (excel file) are organized
61: # (see python script "Link_data_together.py")

```

```

62:
63: # The dictionary is called "parks_and_recreation"
64:
65: # NOTE: The empty dictionary in which all this information is organized is
66: # generated in the main python script and is called "parks_and_recreation"
67:
68:
69: """
70:     3. GENERATE LOCATIONS DICTIONARY (POINT DATA)
71:     make a dictionary in which each key is the prk_loc_id and each value is a
72:     dictionary in which each key is an attribute (as it appears in the cleaned
73:     locations shapefile) and each value is the corresponding attribute value
74: """
75:
76: # Store the name of each field of the shapefile's attribute table to a list
77: fields_locations = [field.name for field in arcpy.ListFields(park_locations_clean_original_shp)]
78: fields_locations += ['SHAPE@']
79:
80: # Generate an empty dictionary in which the attributes of each feature (point)
81: # will be stored using "prk_Loc_id" as unique identifier
82: park_locations = {}
83:
84: # Go through each feature (point) of the shapefile and add corresponding attribute
85: # information to the dictionary
86: attribute_table = arcpy.da.SearchCursor(park_locations_clean_original_shp, fields_locations)
87: for location_data in attribute_table:
88:
89:     # For each feature (point), generate a sub-dictionary in which the keys represent the
90:     # attributes as they appears in the shapefile and the values correspond to the
91:     # park's attribute values.
92:     location_data = dict(zip(fields_locations, location_data))
93:
94:     # Clean the attribute values that are string or unicode types by stripping blank spaces
95:     # and converting them to all lowercase letters (this will make it easier
96:     # to match features with the inventory (Excel file) and the property shapefile (polygons)
97:     location_data.update((k, v.lower().strip()) for k, v in location_data.items() if type(v) in
98: [str, unicode])
99:
100:     # Each key in the dictionary represent the unique identifier of each park ("prk_Loc_id")
101:     # The sub-dictionary generated above is added to the park_Locations dictionary
102:     prk_loc_id = location_data['prk_loc_id']
103:     park_locations[prk_loc_id] = location_data
104:
105: # Close the search cursor
106: del attribute_table
107:
108: """
109:     4. GENERATE INVENTORY DICTIONARY (EXCEL DATA)
110:     make a dictionary in which each key is the inventory_id and each value is a
111:     dictionary in which each key is an attribute (as it appears in the original
112:     excel file) and each value is the corresponding attribute value
113: """
114:
115: # Store the information of the inventory (Excel file) to a list, in which each
116: # line is splitted at each tab (\t) character
117: inventory_data = [row.split('\t') for row in open(park_inventory_txt)]
118:
119: # The first line in the inventory (Excel file) contains the attribute names.
120: # The attribute names are stroed in a list and cleaned by stripping all blank spaces
121: # The attribute names of the amenities are stored in a separate list (will be used further)

```

link_data_together.py

```

122: fields_inventory = map(str.strip, inventory_data[0])
123: amenities_from_inventory = fields_inventory[4:-5]
124:
125: # The rest of the inventory (Excel file) contains the data for each park. The last
126: # 6 rows contain information like "totals" and empty rows that do not contain any
127: # park information.
128: inventory_data = inventory_data[1:-6]
129:
130: # Generate an empty dictionary in which the attributes of each park
131: # will be stored using "inventory_id" as unique identifier
132: park_inventory = {}
133:
134: # Go through each park of the inventory (Excel file) and add corresponding attribute
135: # information to the dictionary. The variable "j" will generate a unique identifier
136: # for parks that do not have an identifier yet
137: j = 1
138: for park_data in inventory_data:
139:
140:     # For each park, generate a sub-dictionary in which the keys represent the
141:     # attributes as they appear in the Excel file and the values correspond to the
142:     # park's attribute values.
143:     park_data = dict(zip(fields_inventory, park_data))
144:
145:     # Clean the attribute values by stripping blank spaces and converting them to
146:     # all lowercase letters (this will make it easier to match features with the
147:     # locations shapefile (points) and the property shapefile (polygons). Also,
148:     # certain attributes have '-' or '_' characters to indicate no amenity is
149:     # available at that park. Additionally, characters like '"' and ',' are sometimes
150:     # used for numbers in the thousands (e.g., "1,000"). These characters are removed.
151:     park_data.update((k, v.lower().strip()) for k, v in park_data.items())
152:     park_data.update((k, re.sub('[-_,]', '', park_data[k])) for k in amenities_from_inventory)
153:
154:     # Each key in the dictionary represents the unique identifier of each park ("inventory_id")
155:     inventory_id = park_data['Abbv']
156:
157:     # The identifier 'TBYC' appears twice. Here, a separate identifier is generated
158:     # for both entries. Also, some parks do not have an identifier. Here an identifier
159:     # is generated for those.
160:     if inventory_id == 'tbyc': inventory_id = 'tbyc_p' + park_data['Name'][-1]
161:     elif inventory_id == '': inventory_id, j = 'inv_%s' % j, j + 1
162:
163:     # The sub-dictionary generated above is added to the park_locations dictionary
164:     park_inventory[inventory_id] = park_data
165:
166:
167: """
168: 5. POPULATE OUTPUT DICTIONARY
169: make a dictionary in which each key is the unique_id, which uniquely identifies
170: every park or recreational facility in the shapefiles and inventory. After this
171: step of the script, each record will contain 3 dictionaries, (1) a polygon_data
172: dictionary, (2) a point_data dictionary and (3) an exceld_data dictionary.
173: These three dictionaries contain the attribute information of the park property
174: shapefile, park locations shapefile and excel file respectively. When a park
175: or recreational facility appears in one or two data sources but not all three,
176: the attributes of the file it does not appear in are all blank.
177: """
178:
179: # First, make a dictionary with all park properties in which park locations and
180: # parks from the inventory are linked to each property
181:
182: arcpy.CopyFeatures_management (park_properties_clean_original_shp, park_properties_clean_shp)

```

link_data_together.py

```

183: arcpy.AddGeometryAttributes_management(park_properties_clean_shp, 'AREA', '', 'ACRES')
184:
185: # Store the name of each field of the shapefile's attribute table to a list
186: fields_properties = [f.name for f in arcpy.ListFields(park_properties_clean_shp)]
187: fields_properties += ['SHAPE@']
188:
189: # Generate a list with all unique identifiers from the location shapefile (points)
190: # from the inventory (Excel file). When a location or park of the inventory can be
191: # matched to a property, the identifier will be removed from this list. Once all
192: # properties have been processed, the identifiers that are left in this list will
193: # correspond to locations (points) and parks of the inventory (Excel file) that
194: # do not appear in the dictionary yet. These parks are added to the dictionary
195: # in the next step of this script.
196: identifiers_not_linked_to_property = park_locations.keys() + park_inventory.keys()
197:
198: # Go through each park property (polygon) and add corresponding attribute
199: # information to the dictionary.
200: attribute_table = arcpy.da.SearchCursor(park_properties_clean_shp, fields_properties)
201: for property_data in attribute_table:
202:
203:     # For each property (polygon), generate a sub-dictionary in which the keys represent
204:     # the attributes as they appear in the shapefile and the values correspond to the
205:     # polygon's attribute values.
206:     property_data = dict(zip(fields_properties, property_data))
207:
208:     # Clean the attribute values that are string or unicode types by stripping blank spaces
209:     # and converting them to all lowercase letters (this will make it easier
210:     # to match features with the inventory (Excel file) and the location shapefile (points)
211:     property_data.update((k, v.lower().strip()) for k, v in property_data.items() if type(v) in
212: [str, unicode])
213:
214:     # Each key in the dictionary represents the unique identifier of each polygon ("prk_prp_id")
215:     # The sub-dictionary generated above is added to the park_locations dictionary
216:     prk_prp_id = property_data['prk_prp_id']
217:     parks_and_recreation[prk_prp_id] = {'polygon data' : property_data}
218:
219:     # To link/join the attribute information to a park corresponding to the polygon in
220:     # the inventory (Excel file) or the locations shapefile (points), the respective
221:     # unique identifiers (foreign keys) will be used
222:     prk_loc_id = property_data['prk_loc_id']
223:     invtory_id = property_data['invtory_id']
224:
225:     # Link park property (polygon) to corresponding park location (point)
226:     try:
227:         # If there is a corresponding park location (point) for this property (polygon),
228:         # this line of code will not send a "KeyError" and the corresponding park
229:         # location attributes (in the form of a dictionary) are stored in the variable
230:         # 'location_data'. The unique identifier is removed from the list of all park
231:         # location identifiers generated above.
232:         location_data = park_locations[prk_loc_id]
233:         if prk_loc_id in identifiers_not_linked_to_property:
234:             identifiers_not_linked_to_property.remove(prk_loc_id)
235:     except:
236:         # If there is no corresponding park location (point) for this property (polygon),
237:         # the lines in the 'try' statement above will send a 'KeyError' and redirect
238:         # to these lines of code, where empty values are generated for each park
239:         # location attribute.
240:         location_data = dict(zip(fields_locations, itertools.cycle([''])))
241:
242:     # Add additional attributes from the corresponding park location to the already
243:     # existing attributes from the park property (polygon)

```

link_data_together.py

```

243: parks_and_recreation[prk_prp_id].update({'point data' : location_data})
244:
245: # Link park property (polygon) to corresponding park in inventory (Excel file)
246: try:
247:     # If there is a corresponding inventory park (Excel file) for this property (polygon),
248:     # this line of code will not send a "KeyError" and the corresponding park
249:     # location attributes (in the form of a dictionary) are stored in the variable
250:     # 'inventory_data'. The unique identifier is removed from the list of all park
251:     # identifiers of the inventory, that is generated above.
252:     inventory_data = park_inventory[invtory_id]
253:     if invtory_id in identifiers_not_linked_to_property:
254:         identifiers_not_linked_to_property.remove(invtory_id)
255: except:
256:     # If there is no corresponding inventory park (Excel file) for this property (polygon),
257:     # the lines in the 'try' statement above will send a 'KeyError' and redirect
258:     # to these lines of code, where empty values are generated for each attribute
259:     # in the inventory.
260:     inventory_data = dict(zip(fields_inventory, itertools.cycle([''])))
261:
262: # Add additional attributes from the corresponding park in the invenotry to the
263: # already existng attributes from the park property (polygon) and park location (point)
264: parks_and_recreation[prk_prp_id].update({'excel data' : inventory_data})
265:
266: # Close the search cursor
267: del attribute_table
268:
269: # Second, for each park Location (point) and park of the inventory (Excel file) that
270: # was not linked to a park property (polygon), an additional dictionary entry is
271: # generated. Also, if a park Location (point) corresponds to a park in the inventory
272: # that had no corresponding propoerty either, these parks are inserted as one entry
273: # (under the smaen identifier) in the dictionary
274:
275: # Go through each identifier of the park Location and/or park of the inventory that
276: # is left in the identifiers_not_linked_to_property list
277: for identifier in identifiers_not_linked_to_property:
278:
279:     # Link identifier to corresponding park Location (point)
280:     try:
281:         # If there is park Location (point) with this identifier, this line of code
282:         # will not send a "KeyError" and the corresponding park location attributes
283:         # (in the form of a dictionary) are stored in the variable 'location_data'.
284:         location_data = park_locations[identifier]
285:     except:
286:         # If there is no park Location (point) with this identifier, the lines in
287:         # the 'try' statement above will send a 'KeyError' and redirect to these
288:         # lines of code, where empty values are generated for each park location attribute.
289:         location_data = dict(zip(fields_locations, itertools.cycle([''])))
290:
291: # Add attributes from the park Location to the dictionary
292: parks_and_recreation[identifier] = {'point data' : location_data}
293:
294: # Link identifier to corresponding park in inventory (Excel file)
295: try:
296:     # If there is an inventory park (Excel file) with this identifier, this
297:     # line of code will not send a "KeyError" and the corresponding park inenvtory
298:     # attributes (in the form of a dictionary) are stored in the variable 'inventory_data'.
299:     inventory_data = park_inventory[identifier]
300: except:
301:     # If there is no inventory park (Excel file) with this identifier, the lines
302:     # in the 'try' statement above will send a 'KeyError' and redirect to
303:     # these lines of code, where empty values are generated for each attribute

```

link_data_together.py

```
304:         # in the inventory.
305:         inventory_data = dict(zip(fields_inventory, itertools.cycle([''])))
306:
307:         # Add attributes from the park in the invenotry to the dictionary
308:         parks_and_recreation[identifier].update({'excel data' : inventory_data})
309:
310:         # Add empty park property (polygon) attributes
311:         property_data = dict(zip(fields_properties, itertools.cycle([''])))
312:         parks_and_recreation[identifier].update({'polygon data' : property_data})
```

8. Clean Attribute Information

clean_attribute_information.py

```

1: # Author:                               Coline C. Dony
2: # Date first written:                   June 10, 2016
3: # Date last updated:                   June 16, 2016
4: # Purpose:                             Clean attribute information
5:
6: # Problem statement:
7: # Mecklenburg County's Department of Park and Recreation's (DPR) property boundaries
8: # are publically available. Additionally, locations of park entrances are publically
9: # available as well. These shapefiles contain data useful for my research and can be
10: # downloaded at: http://maps.co.mecklenburg.nc.us/openmapping/data.html
11: # Michael Kirschman, Deputy Director of Mecklenburg County's DPR shared with me
12: # an Excel spreadsheet containing all parks in their inventory. In this inventory,
13: # the amenities available at each park are provided as well, which is useful for
14: # my research.
15:
16: # To get a more complete set of park and recreational facilities, these three files
17: # are merged together (i.e. each park or recreational property (polygon) is linked
18: # to the corresponding location (point) and information in the inventory (excel file).
19: # See the python script "Link_data_together.py". However, when merged, it is
20: # apparent that the attribute information in these three files is not consistent.
21: # For example, one park or facility is labeled as a Community Park in one data source
22: # and as a Neighborhood Park in the other data source. Inconitencies are found
23: # throughout the attributes. In this script, The attribute information is cleaned
24: # by favoring the information that appears in the inventory (Excel file) provided
25: # by Michael Kirschman, Deputy Director of Mecklenburg County's DPR. If the
26: # park or facility does not appear in the inventory, the park property shapefile
27: # (polygons) is favored and if the park or facility only appears in the locations
28: # shapefile, that attribute value is chosen to represent the final attribute value
29: # of that park or recreational facility.
30:
31:
32: """
33:     1. INPUT FILES
34:     original, already existng files, used in this script
35: """
36:
37: # The populated dictionary in which the park and recreational facilities that
38: # appear in the park property shapefile (polygons), park locations shapefile (points)
39: # and inventory (excel file) are organized )see python script "Link_data_together.py"
40:
41: # The dictionary is called "parks_and_recreation"
42:
43:
44: """
45:     2. OUTPUT SUB-DICTIONARY
46:     make a sub-dictionary to the "parks_and_recreation" dictionary called "clean_data"
47:     in which each key is a final attribute useful to my dissertation and each value
48:     corresponds to the most reliable attribute value out of the inventory (excel
49:     file), park property shapefile (polygons) or park locations shapefile (points)
50: """
51:
52: # The dictionary is called "clean_data"
53:
54:
55: """
56:     3. CHOOSING THE MOST RELIABLE ATTRIBUTE VALUE
57:     make a dictionary for each park or recreational facility in which each key
58:     is a final attribute useful to my dissertation and each value corresponds to
59:     the most reliable attribute
60: """
61: max_size = 0

```

```

62: # Going through the dictionary with all park and recreational facilities
63: for unique_id, attributes in parks_and_recreation.items():
64:
65:     # separating out the attributes of each data source for this particular
66:     # park or recreational facility
67:     polygon_data = attributes['polygon data']
68:     point_data = attributes['point data']
69:     excel_data = attributes['excel data']
70:
71:     # Generate an empty dictionary in which the final attributes, useful to my
72:     # dissertation will be stored
73:     clean_attributes = {}
74:
75:     # PARK NAME: Choose the park name by favoring the name appearing in the
76:     # park property shapefile (polygons). If not available, favor the name appearing
77:     # in the park locations shapefile (points). If not available either, favor the
78:     # name in the inventory (excel file). Add the chosen name to the
79:     # clean_attributes dictionary
80:     names = [polygon_data['property'], point_data['prkname'], excel_data['Name']]
81:     clean_attributes['name'] = [name for name in names if not name == ''][0]
82:
83:     # PARK TYPE: Choose the park type by favoring the type appearing in the
84:     # inventory (excel file). If not available, favor the type appearing
85:     # in the park properties shapefile (polygons). If not available either, favor
86:     # the type in the park locations shapefile (points). Add the chosen type to the
87:     # clean_attributes dictionary
88:     types = [excel_data['Class'], polygon_data['park_type'], point_data['prktype']]
89:     clean_attributes['type'] = [park_type for park_type in types if not park_type == ''][0]
90:
91:     # PARK STATUS: Choose the park status by favoring the status appearing in the
92:     # inventory (excel file). If not available, favor the status appearing
93:     # in the park properties shapefile (polygons). If not available either, favor
94:     # the status in the park locations shapefile (points). Add the chosen status to
95:     # the clean_attributes dictionary
96:     statuses = [excel_data['Status'], polygon_data['status'], point_data['prkstatus']]
97:     status = [status for status in statuses if not status == ''][0]
98:     clean_attributes['status'] = status.title()
99:
100:    # PARK ADDRESS: Choose the park address by favoring the address appearing in the
101:    # inventory (excel file). If not available, favor the address appearing
102:    # in the park locations shapefile (points). Add the chosen address to
103:    # the clean_attributes dictionary
104:    addresses = [excel_data['Address'], point_data['prkaddr']]
105:    addresses = [address for address in addresses if not address == '']
106:    address = addresses[0] if len(addresses) else ''
107:    clean_attributes['address'] = address.title()
108:
109:    # PARK ENTRANCE GEOMETRY: Choose the park x/y-coordinated by extracting the
110:    # point geometry in the park locations shapefile (points). Add the entrance's
111:    # geometry to the clean_attributes dictionary
112:    clean_attributes['entrance'] = point_data['SHAPE@']
113:    clean_attributes['property'] = polygon_data['SHAPE@']
114:
115:    # PARK MANAGEMENT: Choose the park management by extracting the information
116:    # from the the park properties shapefile (polygons). Add the management
117:    # to the clean_attributes dictionary
118:    clean_attributes['managed_by'] = polygon_data['owner']
119:
120:    # PARK SIZE: Choose the park size by favoring the size appearing in the
121:    # inventory (excel file). If not available, favor the size appearing
122:    # in the park locations shapefile (points). Add the chosen size to

```

```

123: # the clean_attributes dictionary
124: sizes = [excel_data['Acres'], point_data['prksize'], polygon_data['POLY_AREA']]
125: sizes = [park_size for park_size in sizes if not park_size == '']
126: park_size = sizes[0] if len(sizes) else 0
127: clean_attributes['size'] = float(park_size)
128:
129: # PARK AGE: Calculate the park age by extracting the park development date
130: # from the the park locations shapefile (points) and taking the difference in
131: # years from 2015. Add the age to the clean_attributes dictionary
132: park_date = 2015 - point_data['prkdate'].year if point_data['prkdate'] else None
133: clean_attributes['age'] = park_date
134:
135: # PARK AMENITIES: Calculate the number of each amenity by extracting this
136: # information from the the inventory (excel file). For my dissertation, certain
137: # original amenities are merged together (e.g. playground for 2-5 year old's
138: # and playground for 5-12 year old's are merged to one amenity called playgournd)
139:
140: # List of amenities useful for my research
141: clean_amenity_fields = [
142:     'playground', 'basketball', 'pitch', 'tennis', 'multipurpose', 'volleyball',
143:     'trails', 'pool', 'skate park', 'dog park', 'pavilion', 'recreation center',
144:     'nature center', 'museum center']
145:
146: clean_amenity_fields_10char = [
147:     'playground', 'basketball', 'pitch', 'tennis', 'multiprpse', 'volleyball',
148:     'trails', 'pool', 'skate_park', 'dog_park', 'pavilion', 'recrea_ctr',
149:     'nature_ctr', 'museum_ctr', 'recrea_id', 'nature_id', 'museum_id']
150:
151: # Generate an empty dictionary in which each key is one amenity and each value
152: # is the corresponding numebr or size of each amenity
153: amenities = {}
154:
155: # Going through each amenity in the list above and calculating how many of each
156: # amenity there is (merging certain original amenities together)
157: total_number_amenities = 0
158: for index, clean_amenity in enumerate(clean_amenity_fields):
159:     clean_amenity_10char = clean_amenity_fields_10char[index]
160:     amenities[clean_amenity_10char] = 0
161:     for amenity_from_inventory in amenities_from_inventory:
162:         amenity_from_inventory_temp = amenity_from_inventory.lower()
163:         amenity_from_inventory_temp = amenity_from_inventory_temp.replace('softball', 'pitch')
164:         amenity_from_inventory_temp = amenity_from_inventory_temp.replace('baseball', 'pitch')
165:         if clean_amenity.split()[0] in amenity_from_inventory_temp:
166:             amenity_value = excel_data[amenity_from_inventory]
167:             amenity_value = 0 if amenity_value == '' else float(amenity_value)
168:             amenities[clean_amenity_10char] += amenity_value
169:
170:             if clean_amenity.endswith('center') or clean_amenity == 'pool':
171:                 if amenity_value > 0: total_number_amenities += 1
172:             else: total_number_amenities += amenity_value
173:
174:             if clean_amenity.endswith('center'): amenities[clean_amenity_fields_10char[index + 3]] = ''
175:
176: amenities['am_total'] = total_number_amenities
177:
178: # Add the amenities dictionary to the clean_attributes dictionary
179: clean_attributes['amenities'] = amenities
180:
181: # PARK NOTES: Extract the park notes from the the inventory (excel file).
182: # Add the park notes to the clean_attributes dictionary
183: park_notes = excel_data['Notes']

```

```

184:     clean_attributes['notes'] = park_notes
185:
186:     clean_attributes['weblink'] = ''
187:
188:     # Add clean attributes to the park or recreational facility in the
189:     # park_and_recreation dictionary
190:     parks_and_recreation[unique_id]['clean data'] = clean_attributes
191:
192:
193: """
194:     4. FURTHER CLEAN ATTRIBUTE INFORMATION
195:     further clean certain attributes so that naming of attributes is consistent
196: """
197:
198: # Going through the dictionary with all park and recreational facilities
199: for unique_id, attributes in parks_and_recreation.items():
200:
201:     # Select the sub-dictionary that contains the cleaned attribute information
202:     # for that park or recreational facility
203:     clean_data = attributes['clean data']
204:
205:     # 1. PARK NAME:
206:     name = clean_data['name']
207:
208:     # Remove the park type from their name
209:     for words in ('neighborhood park', 'community park', 'regional park'):
210:         name = name.replace(words, 'park')
211:         name = name.replace('nature preserve', 'preserve')
212:         name = name.replace('recreation center', 'center')
213:
214:     # Make golf course names more consistent
215:     for words in ('country club', 'golf course', 'golf resort', 'golf club', 'golf and country
club', 'golf learning ctr'):
216:         name = name.replace(words, 'golf')
217:
218:     # Fix additional inconsistencies
219:     name = name.replace(' street ', ' st ')
220:     name = name.title()
221:     name = name.replace('&', 'and')
222:     name = name.replace('"S"', 's')
223:
224:     # Update name
225:     clean_data['name'] = name
226:
227:     # 2. PARK TYPE:
228:     park_type = clean_data['type']
229:
230:     # The inventory uses greenways (with 's') and the park property shapefile (polygons)
231:     # uses greenway (without 's'). Making this type consistent
232:     park_type = park_type.replace('greenways', 'greenway')
233:
234:     # Update type
235:     clean_data['type'] = park_type.title()
236:
237:     # 2. MANAGEMENT:
238:     managed_by = clean_data['managed_by']
239:
240:     # Use CMS instead of School to make it consistent with other parks and recreational
241:     # facilities that are managed by CMS
242:     managed_by = managed_by.replace('school', 'CMS')
243:

```

clean_attribute_information.py

```
244:     # Fix additional inconsistencies
245:     managed_by = managed_by.title()
246:     managed_by = managed_by.replace('Cms', 'CMS')
247:     managed_by = managed_by.replace('Of', 'of')
248:
249:     # Update management
250:     clean_data['managed_by'] = managed_by
251:
252:     # Update clean data dictionary for that park or recreational facility
253:     parks_and_recreation[unique_id]['clean data'].update(clean_data)
```


9. Edit Attribute Information

edit_attribute_information.py

```

1: # Author:                               Coline C. Dony
2: # Date first written:                   June 13, 2016
3: # Date last updated:                   June 16, 2016
4: # Purpose:                             Clean attribute information
5:
6: # Problem statement:
7: # Mecklenburg County's Department of Park and Recreation's (DPR) property boundaries
8: # are publically available. Additionally, locations of park entrances are publically
9: # available as well. These shapefiles contain data useful for my research and can be
10: # downloaded at: http://maps.co.mecklenburg.nc.us/openmapping/data.html
11: # Michael Kirschman, Deputy Director of Mecklenburg County's DPR shared with me
12: # an Excel spreadsheet containing all parks in their inventory. In this inventory,
13: # the amenities available at each park are provided as well, which is useful for
14: # my research.
15:
16: # To get a more complete set of park and recreational facilities, these three files
17: # are merged together (i.e. each park or recreational property (polygon) is linked
18: # to the corresponding location (point) and information in the inventory (excel file).
19: # See the python script "Link_data_together.py". However, when merged, it is
20: # apparent that the attribute information in these three files is not consistent.
21: # After making the attribute information more consistent (see python script
22: # clean_attribute_information.py), there are still some attributes that need
23: # editing after manual inspection of these parks on the DPR's webpage and other
24: # online sources.
25:
26: """
27:     1. INPUT FILES
28:     original, already existing files, used in this script
29: """
30:
31: # The populated dictionary in which the park and recreational facilities that
32: # appear in the park property shapefile (polygons), park locations shapefile (points)
33: # and inventory (excel file) are organized (see python script "Link_data_together.py")
34: # The dictionary is called "parks_and_recreation"
35:
36: # The sub-dictionary "clean data" in which the cleaned attribute information is
37: # stored.
38:
39: """
40:
41:     2. OUTPUT: EDITED SUB-DICTIONARY
42:     edit the sub-dictionary to the "parks_and_recreation" dictionary called "clean data"
43: """
44:
45:
46: """
47:     3. EDIT PARK NAMES
48:     edit names of certain park or recreational facilities
49: """
50:
51: # List of parks or recreational facilities of which the name needs editing
52: name_edits = [ ['miss_4', 'Choate / Steele Creek Park'],
53:                ['alex', 'Alexander Street Park'],
54:                ['aqua', 'Mecklenburg County Aquatic Center'],
55:                ['bytr', 'Back Yard Trails - Tyvola/WWTP'],
56:                ['catw', 'U.S. National Whitewater Center'],
57:                ['cdhp', 'Robert Caldwell Bradford Park'],
58:                ['cedr', 'Cedarwood Park'],
59:                ['cmgp', 'Camp Green Park'],
60:                ['drta', 'Derita Creek Park'],
61:                ['drud', 'Druid Hills Elementary School'],

```

edit_attribute_information.py

```

62:         ['ebmp', 'E.B. Moore Park'],
63:         ['h521', 'Elon Park'],
64:         ['horn', 'Hornet's Nest Park'],
65:         ['hrnp', 'Harrisburg Road Neighborhood Park'],
66:         ['hysd', 'Matthews-Sardis Park'],
67:         ['irwn', 'Irwin Center and Ray's Splash Planet'],
68:         ['jatc', 'Jeff Adams Tennis Center'],
69:         ['jcsu', 'JCS University Track'],
70:         ['math', 'Mecklenburg County Sportsplex'],
71:         ['mcke', 'McKee Road Park'],
72:         ['moep', 'Merry Oaks Park'],
73:         ['natm', 'Freedom Park Museum'],
74:         ['pali', 'Palisades Park Elementary School'],
75:         ['penn', 'Charlie and Ida Graham / Pennington Park'],
76:         ['perl', 'Pearl Street Park'],
77:         ['pvly', 'Pine Valley / Longleaf Park'],
78:         ['rock', 'Theresa Clark Elder Park'],
79:         ['scs', 'Steele Creek Elementary School'],
80:         ['uncc_gwy', 'UNCC Greenway'],
81:         ['weac', 'Robbins / Westmoreland Park']]
82:
83: # Edit each name in the "clean data" sub-dictionary
84: for unique_id, edit_name in name_edits:
85:     parks_and_recreation[unique_id]['clean data']['name'] = edit_name
86:
87: """
88:
89:     4. EDIT PARK TYPES
90:     edit types of certain park or recreational facilities
91: """
92:
93: # List of parks or recreational facilities of which the type needs editing
94: type_edits = [ ['Community Park', ['hysd', 'uncc']],
95:                ['Neighborhood Park', ['pali', 'inv_2']],
96:                ['Nature Preserve', ['shrm']],
97:                ['Regional Park', ['math']]
98:
99: # Edit each type in the "clean data" sub-dictionary
100: for edit_type, unique_ids in type_edits:
101:     for unique_id in unique_ids:
102:         parks_and_recreation[unique_id]['clean data']['type'] = edit_type
103:
104: """
105:
106:     5. EDIT PARK STATUS
107:     edit the status of certain park or recreational facilities
108: """
109:
110: # List of parks or recreational facilities of which the status needs editing
111: status_edits = [['Developed',
112:                  ['miss_7', 'afrm', 'canp', 'coww', 'davs',
113:                   'drud', 'ffpk', 'hysd', 'pali', 'penc',
114:                   'rurl', 'shrm', 'stph', 'wbnp']],
115:                 ['No Public Access',
116:                  ['autn', 'berr', 'brnh', 'shfp', 'gcnp',
117:                   'hayp', 'mtld', 'stvn']],
118:                 ['Limited Public Access',
119:                  ['bigr', 'gate', 'oehp', 'prnp', 'tmnt']],
120:                 ['Proposed',
121:                  ['dblp', 'fbnp', 'univ']],

```

edit_attribute_information.py

```

122:                                     'tbyc_und']],
123:                                     ['Closed',      ['cadi']],
124:                                     ['No longer occupied', ['epdo']]]
125:
126: # Edit each status in the "clean data" sub-dictionary
127: for edit_status, unique_ids in status_edits:
128:     for unique_id in unique_ids:
129:         parks_and_recreation[unique_id]['clean data']['status'] = edit_status
130:
131:
132: """
133:     6. EDIT PARK ENTRANCE GEOMETRY
134:     edit the entrance geometry of certain park or recreational facilities
135: """
136:
137: # Edit each geometry in the "clean data" sub-dictionary
138: for unique_id, entrance in [(('clar', arcpy.Point(1471280.900, 581452.519))):
139:     parks_and_recreation[unique_id]['clean data']['entrance'] = entrance
140:
141:
142: """
143:     7. EDIT PARK MANAGEMENT
144:     edit the management of certain park or recreational facilities
145: """
146:
147: # List of parks or recreational facilities of which the management needs editing
148: managed_by_edits = [['Mint Hill',      ['miss_7']],
149:                     ['Town of Huntersville', ['canp', 'dbwp', 'nmpk', 'cdhp',
150:                                     'miss_1']],
151:                     ['Town of Davidson',   ['ffpk', 'inv_1']],
152:                     ['Town of Matthews',   ['hysd', 'sqr1']],
153:                     ['CMS',               ['pali', 'penc', 'mlkr', 'tmrc',
154:                                     'tsrc', 'wprc', 'bere']],
155:                     ['Tarheel Trailblazers', ['bytr']],
156:                     ['City of Charlotte',   ['grnp']],
157:                     ['Town of Cornelius',   ['weac']],
158:                     ['Mecklenburg County',  ['glf_15']],
159:                     ['Real Estate Services (RES) Dept.', ['amjp', 'indp', 'latp']]]
160:
161: # Edit each management in the "clean data" sub-dictionary
162: for edit_managed_by, unique_ids in managed_by_edits:
163:     for unique_id in unique_ids:
164:         parks_and_recreation[unique_id]['clean data']['managed_by'] = edit_managed_by
165:
166:
167: """
168:     8. EDIT PARK SIZE
169:     edit the size of certain park or recreational facilities
170: """
171:
172: # Edit each size in the "clean data" sub-dictionary
173: for unique_id, edit_size in [(('prnp', 49)]:
174:     parks_and_recreation[unique_id]['clean data']['size'] = edit_size
175:
176:
177: """
178:     9. LINK RECREATIONAL CENTERS TO THEIR PARK
179:     certain recreational centers are on a park property. To avoid two
180:     different entrances to the same park property, these centers are
181:     added as an amenity
182: """

```

```

183: # Keep track of all centers that are attached to a park
184: centers_within_a_park = []
185:
186: # List of parks and their corresponding recreational center
187: rec_center_edits = [['abrp', 'abrc'],
188:                     ['amjp', 'amjc'],
189:                     ['upst', 'bere'],
190:                     ['clan', 'agrc'],
191:                     ['h521', 'elnc'],
192:                     ['hapk', 'hffc'],
193:                     ['gray', 'naom'],
194:                     ['grnp', 'grcr'],
195:                     ['penc', 'hgrc'],
196:                     ['indp', 'hawc'],
197:                     ['latp', 'latc'],
198:                     ['mlcp', 'mldc'],
199:                     ['mdlp', 'mdlc'],
200:                     ['mens', 'grdy'],
201:                     ['moep', 'morc'],
202:                     ['mthp', 'mthc'],
203:                     ['npdr', 'nmrc'],
204:                     ['pkrd', 'pktn'],
205:                     ['revp', 'revc'],
206:                     ['rspk', 'jadc'],
207:                     ['sthw', 'svct'],
208:                     ['sugp', 'sugc'],
209:                     ['tmcw', 'wgrc'],
210:                     ['tucp', 'tucc'],
211:                     ['wcpk', 'wclt']]
212:
213: # Merge both amenities and add the unique identifier of the recreational center
214: # to the park in the 'recreation center' amenity.
215: for unique_id, rec_center_id in rec_center_edits:
216:     unique_id = unique_id
217:     rec_center_id = rec_center_id
218:     park_amenities = parks_and_recreation[unique_id]['clean data']['amenities']
219:     rec_center_amenities = parks_and_recreation[rec_center_id]['clean data']['amenities']
220:     added_amenities = 0
221:     for clean_amenity_10char in clean_amenity_fields_10char:
222:         park_amenity = park_amenities[clean_amenity_10char] +
223:         rec_center_amenities[clean_amenity_10char]
224:         parks_and_recreation[unique_id]['clean data']['amenities'][clean_amenity_10char] =
225:         park_amenity
226:         if clean_amenity_10char.endswith('id'): continue
227:         elif clean_amenity_10char.endswith('ctr') or clean_amenity_10char == 'pool':
228:             if rec_center_amenities[clean_amenity_10char] > 0: added_amenities += 1
229:             else: added_amenities += rec_center_amenities[clean_amenity_10char]
230:         parks_and_recreation[unique_id]['clean data']['amenities']['am_total'] += added_amenities
231:         total_number_amenities = parks_and_recreation[unique_id]['clean data']['amenities']['am_total']
232:         parks_and_recreation[unique_id]['clean data']['amenities']['recrea_id'] = rec_center_id
233:         centers_within_a_park += [rec_center_id]
234:
235: """
236: 10. LINK NATURE CENTERS TO THEIR PARK
237: nature centers are on park properties. To avoid two different entrances to the
238: same park property, these centers are added as an amenity
239: """
240: # List of parks and their corresponding nature center
241: nature_center_edits = [ ['latn', ['lpeq', 'lath', 'crap']],

```

edit_attribute_information.py

```

242:             ['mcdl', ['']],
243:             ['renp', ['']]]
244:
245: # Add the unique identifier of the recreational center to the park in the
246: # 'nature center' amenity.
247: for unique_id, nature_center_ids in nature_center_edits:
248:     parks_and_recreation[unique_id]['clean data']['amenities']['nature_ctr'] = 1
249:     parks_and_recreation[unique_id]['clean data']['amenities']['nature_id'] = ',
'.join(nature_center_ids)
250:     centers_within_a_park += nature_center_ids
251:
252: """
253:     11. LINK MUSEUM TO THEIR PARK
254:     certain museums are on park properties. To avoid two different entrances to the
255:     same park property, these centers are added as an amenity
256: """
257:
258: # Add the unique identifier of the recreational center to the park in the
259: # 'museum' amenity.
260: for unique_id, museum_id in [['free', 'natm']]:
261:     parks_and_recreation[unique_id]['clean data']['amenities']['museum_ctr'] = 1
262:     parks_and_recreation[unique_id]['clean data']['amenities']['museum_id'] = museum_id
263:     centers_within_a_park += [museum_id]

```

10. Add Park Weblinks

```

add_weblinks.py

1: """
2:     12. ADD WEBPAGE TO EACH PARK
3:     certain parks have a webpage on the DPR's webpage with detailed information.
4: """
5:
6: # Generate a reverse dictionary in which each key is the name of the park and
7: # each value is the unique_id. This will be useful to link the webpage to the
8: # dictionary with all park and recreational facilities.
9: reverse_dictionary = {}
10: for unique_id, attributes in parks_and_recreation.items():
11:     name = attributes['clean data']['name'].lower().replace('.', '')
12:     if unique_id.endswith('_und'): name += ' undeveloped'
13:     reverse_dictionary[name] = unique_id
14:
15:
16: # PARKS
17:
18: # The webpage of the DPR organizes their parks and webpages in the three County regions.
19: regions = ['North', 'South', 'Central']
20:
21: # Go through the parks listed in each county region
22: for region in regions:
23:
24:     # Main page
25:     url =
26:     r'http://charmeck.org/mecklenburg/county/ParkandRec/Parks/ParksByRegion/%sRegion/Pages/default.aspx'
27:     % region
28:
29:     # String of page content that contains links to each park in that region
30:     page_contents = urllib2.urlopen(url).read().split('mc-publishing-page-content')[1]
31:
32:     # Convert string to list by splitting the string at each HTML list tag
33:     link_contents = [content for content in page_contents.split('li>') if content.startswith('<a
34: href="/mecklenburg/county/ParkandRec')]
35:
36:     # Go through each item in the list, which contains the park name, its address and webpage link
37:     for link_content in link_contents:
38:
39:         # Park's webpage
40:         link, = [r'http://charmeck.org' + part for part in link_content.split('') if
41: part.startswith('/mecklenburg')]
42:
43:         # Park's address
44:         replace = re.compile('|'.join(map(re.escape, ['&#160;', '</', 'field', '<span>', ',', ' '
45: ])))
46:         adres = [replace.sub('', part) for part in link_content.split('</a>')][-1].strip()
47:
48:         # Park's name
49:         park_name, = [part[:-3] for part in link_content.split('>') if part.endswith('</a>')]
50:         park_name = park_name.lower().strip().strip(',').replace('.', '')
51:
52:         # Link each park to the dictionary using the reverse dictionary
53:         for option in range(5):
54:             if option == 0: name = park_name
55:             elif option == 1: name = park_name + ' park'
56:             elif option == 2: name = park_name.replace(' street ', ' st ')
57:             elif option == 3:
58:                 replace = re.compile('|'.join(map(re.escape, ['neighborhood park', 'regional park',
59: 'community park', 'district park', 'field'])))
60:                 name = replace.sub("park", park_name)
61:             else:

```

add_weblinks.py

```

56:         if park_name == 'david b waymer flyer field': name = 'david b waymer flying field'
57:         elif park_name == 'colonel francis beatty park': name = 'colonel francis j beatty
    park'
58:         elif park_name == 'steele creek park': name = 'steele creek elementary school'
59:         elif park_name == 'druid hills': name = 'druid hills elementary school'
60:         elif park_name == '': name = 'fourth ward park'
61:         else: print park_name
62:         try:
63:             unique_id = reverse_dictionary[name]
64:             parks_and_recreation[unique_id]['clean data']['weblink'] = link
65:             break
66:         except: continue
67:
68:
69: # NATURE PRESERVES
70:
71: # Main page
72: url =
    r'http://charmeck.org/mecklenburg/county/ParkandRec/StewardshipServices/NaturePreserves/Pages/Preser
73:
74: # String of page content that contains links to each nature preserve
75: page_contents = urllib2.urlopen(url).read().split('mc-publishing-page-content')[1]
76:
77: # Convert string to list by splitting the string at each HTML list tag
78: link_contents = [content for content in page_contents.split('li>') if
    'href="/mecklenburg/county/ParkandRec/StewardshipServices/NaturePreserves' in content]
79:
80: # Go through each item in the list, which contains the preserve name and webpage link
81: for link_content in link_contents:
82:
83:     # Preserve's webpage
84:     link, = [r'http://charmeck.org' + part for part in link_content.split('') if
    part.startswith('/mecklenburg')]
85:
86:     # Preserve's Name
87:     preserve = [part.split('<')[0] for part in link_content.split('>') if part.endswith('</a>') or
    part.endswith('</font>')][0]
88:     preserve = preserve.replace('&#160;', ' ').strip().lower()
89:
90:     # Link each preserve to the dictionary using the reverse dictionary
91:     for option in range(2):
92:         if option == 0: name = preserve + ' preserve'
93:         else:
94:             if preserve == "cowan's ford": name = 'cowans ford wildlife refuge'
95:             elif preserve == 'shuffleton prairie': name = 'shuffletown prairie spreserve'
96:             elif preserve == 'winget': name = 'thomas m winget preserve'
97:             else: print preserve
98:         try:
99:             unique_id = reverse_dictionary[name]
100:             parks_and_recreation[unique_id]['clean data']['weblink'] = link
101:             break
102:         except: continue
103:
104:
105: # ADDITIONAL NATURE PRESERVES
106:
107: # McDowell and Sherman Branch Nature Preserves
108: link =
    'http://charmeck.org/mecklenburg/county/ParkandRec/StewardshipServices/NaturePreserves/Pages/Preserv
109: for unique_id in ['mtld', 'shrm']:
110:     parks_and_recreation[unique_id]['clean data']['weblink'] = link

```

add_weblinks.py

```

111:
112: # Latta Plantation Nature Preserve
113: link =
114:     'http://charmeck.org/mecklenburg/county/ParkandRec/StewardshipServices/NaturePreserves/Pages/Latta%2
115: for unique_id in ['crap', 'lath', 'lpeq']:
116:     parks_and_recreation[unique_id]['clean data']['weblink'] = link
117:
118: # RECREATION CENTERS
119:
120: # Main page
121: url =
122:     r'http://charmeck.org/mecklenburg/county/ParkandRec/Facilities/RecreationCenters/Pages/Default.aspx'
123: # String of page content that contains links to each recreation center
124: page_contents = urllib2.urlopen(url).read().split('mc-publishing-page-content')[1]
125:
126: # Convert string to List by splitting the string at each HTML list tag
127: link_contents = [content for content in page_contents.split('td') if
128:     '/mecklenburg/county/ParkandRec/Facilities/RecreationCenters/Pages' in content]
129: # Go through each item in the list, which contains the preserve name and webpage link
130: for link_content in link_contents:
131:
132:     # Recreation center's webpage
133:     link, = [r'http://charmeck.org' + part for part in link_content.split('') if
134:         part.startswith('/mecklenburg')]
135:
136:     # Recreation center's name
137:     rec_center = [part.split('<')[0] for part in link_content.split('>') if part.endswith('</a') or
138:         part.endswith('</font')][0]
139:     rec_center = rec_center.replace('&#160;', ' ').strip().lower()
140:
141:     # Link each recreation center to the dictionary using the reverse dictionary
142:     for option in range(3):
143:         if option == 0: name = rec_center.replace('recreation center', 'center')
144:         elif option == 1: name = rec_center.replace('recreation center', ' center')
145:         else:
146:             if rec_center == 'mlk middle schoolrecreation site': name = 'martin luther king jr
147:                 center'
148:             elif rec_center == 'revolution park sports academy': name = 'revolution sports and
149:                 learning academy'
150:             elif rec_center == 'winget schoolrecreation site': name = 'winget center'
151:             else: print rec_center
152:         try:
153:             unique_id = reverse_dictionary[name]
154:             parks_and_recreation[unique_id]['clean data']['weblink'] = link
155:             break
156:         except: continue
157:
158: # DEVELOPED GREENWAYS
159:
160: # Main page
161: url =
162:     r'http://charmeck.org/mecklenburg/county/ParkandRec/Greenways/OpenGreenways/Pages/default.aspx'
163: # String of page content that contains links to each greenway
164: page_contents = urllib2.urlopen(url).read().split('ms-core-sideNavBox-removeLeftMargin')[1]
165:
166: # Convert string to List by splitting the string at each HTML list tag

```

 add_weblinks.py

```

164: link_contents = [content for content in page_contents.split('li>') if
    '/mecklenburg/county/ParkandRec/Greenways/OpenGreenways' in content]
165:
166: # Go through each item in the List, which contains the greenway name and webpage link
167: for link_content in link_contents:
168:
169:     # Greenway webpage
170:     link, = [r'http://charmeck.org' + part for part in link_content.split('') if
    part.startswith('/mecklenburg')]
171:
172:     # Greenway name
173:     greenway = [part.split('<')[0] for part in link_content.split('menu-item-text">')][1]
174:     greenway = greenway.replace('&#39;', '').strip().lower()
175:
176:     # Link each greenway to the dictionary using the reverse dictionary
177:     for option in range(2):
178:         if option == 0: names = [greenway]
179:         else:
180:             if greenway == 'briar creek greenway - arnold drive to masonic drive': names = ['briar
    creek greenway - arnold']
181:             elif greenway == 'briar creek greenway - myers park high school': names = ['briar creek
    greenway - colony']
182:             elif greenway == 'little sugar creek greenway':
183:                 names = ['little sugar creek greenway - cumberland', 'little sugar creek greenway -
    elizabeth', 'little sugar creek greenway - ramblewood']
184:             elif greenway == 'irwin creek and stewart creek greenways':
185:                 names = ['irwin creek greenway - clanton', 'irwin creek greenway - sycamore',
    'stewart creek greenway']
186:             elif greenway == 'lower mcalpine creek, mcmullen creek, and four mile creek greenways':
187:                 names = ['mcalpine creek greenway - johnston', 'mcmullen creek greenway', 'four
    mile creek greenway - johnston']
188:             elif greenway == 'mallard creek and clark's creek':
189:                 names = ['mallard creek greenway - amaranthus', 'mallard creek greenway -
    university', 'four mile creek greenway - johnston', 'clarks creek greenway']
190:             elif greenway == "upper mcalpine creek greenway":
191:                 names = ['mcalpine creek greenway - monroe', 'mallard creek greenway - university',
    'four mile creek greenway - johnston', 'clarks creek greenway']
192:             else: print greenway
193:         try:
194:             for name in names:
195:                 unique_id = reverse_dictionary[name]
196:                 parks_and_recreation[unique_id]['clean data']['weblink'] = link
197:             break
198:         except: continue
199:
200:
201: # UNDEVELOPED GREENWAYS
202:
203: # Main page
204: url =
    r'http://charmeck.org/mecklenburg/county/ParkandRec/Greenways/PlannedGreenways/Pages/default.aspx'
205:
206: # String of page content that contains links to each greenway
207: page_contents = urllib2.urlopen(url).read().split('mc-publishing-page-content')[1]
208:
209: # Convert string to List by splitting the string at each HTML list tag
210: link_contents = [content for content in page_contents.split('li>') if
    '/mecklenburg/county/ParkandRec/Greenways/PlannedGreenways' in content]
211:
212: # Go through each item in the List, which contains the greenway name and webpage link
213: for link_content in link_contents:

```

```

214:
215:     # Greenway webpage
216:     link, = [r'http://charmeck.org' + part for part in link_content.split('') if
part.startswith('/mecklenburg')]
217:
218:     # Greenway name
219:     greenway, = [part.split('<')[0] for part in link_content.split('>') if part.endswith('</a')]
220:     greenway = greenway.lower()
221:
222:     # Link each greenway to the dictionary using the reverse dictionary
223:     for option in range(3):
224:         if option == 0: names = [greenway]
225:         elif option == 1: names = [greenway + ' undeveloped']
226:         else:
227:             if greenway == 'mcalpine creek greenway extension': names = ['mcalpine creek greenway
undeveloped']
228:             elif greenway == 'torrence, lower mcdowell creek greenway':
229:                 names = ['torrence creek greenway undeveloped', 'mcdowell creek greenway
undeveloped']
230:             else: print greenway
231:         try:
232:             for name in names:
233:                 unique_id = reverse_dictionary[name]
234:                 parks_and_recreation[unique_id]['clean data']['weblink'] = link
235:                 status = parks_and_recreation[unique_id]['clean data']['status']
236:                 parks_and_recreation[unique_id]['clean data']['status'] = 'Proposed' if status ==
'Undeveloped' else status
237:                 break
238:         except: continue
239:
240:     # ADDITIONAL UNDEVELOPED GREENWAYS
241:
242:     # Main page
243:     link =
'http://charmeck.org/mecklenburg/county/ParkandRec/Greenways/PlannedGreenways/Pages/default.aspx'
244:
245:     # Link each greenway to the dictionary
246:     for unique_id, attributes in parks_and_recreation.items():
247:         if unique_id.endswith('_und') and attributes['clean data']['managed_by'] == '':
248:             parks_and_recreation[unique_id]['clean data']['weblink'] = link
249:
250:     # GOLF COURSES
251:
252:     # Link each greenway to the dictionary
253:     for unique_id in ['rglf', 'revg', 'ctmy', 'sunh', 'glf_15']:
254:         parks_and_recreation[unique_id]['clean data']['weblink'] = 'http://www.charlottepublicgolf.com/'
255:
256:     # The traditional golf course's webpage
257:     parks_and_recreation['trgl']['clean data']['weblink'] = 'http://www.thetraditiongolfclub.com/'

```

11. Extract Clean Datasets

extract_clean_files.py

```

1: # Author:                               Coline C. Dony
2: # Date first written:                   June 13, 2016
3: # Date last updated:                   June 16, 2016
4: # Purpose:                             Extract clean files
5:
6: # Problem statement:
7: # Mecklenburg County's Department of Park and Recreation's (DPR) property boundaries
8: # are publically available. Additionally, locations of park entrances are publically
9: # available as well. These shapefiles contain data useful for my research and can be
10: # downloaded at: http://maps.co.mecklenburg.nc.us/openmapping/data.html
11: # Michael Kirschman, Deputy Director of Mecklenburg County's DPR shared with me
12: # an Excel spreadsheet containing all parks in their inventory. In this inventory,
13: # the amenities available at each park are provided as well, which is useful for
14: # my research.
15:
16: # To get a more complete set of park and recreational facilities, these three files
17: # are merged together (i.e. each park or recreational property (polygon) is linked
18: # to the corresponding location (point) and information in the inventory (excel file).
19: # See the python script "Link_data_together.py". However, when merged, it is
20: # apparent that the attribute information in these three files is not consistent.
21: # After making the attribute information more consistent (see python script
22: # clean_attribute_information.py), and editing additional attribute information
23: # after manual inspection of these parks on the DPR's webpage and other
24: # online sources, the dataset is now most complete and consistent across datasources.
25: # In this script, the three original files are extracted (polygons shapefile,
26: # points shapefile and excel file) cut with cleaned data and where the three
27: # sources can be linked to one-another easily.
28:
29:
30: """
31:     1. INPUT FILES
32:     original, already existing files, used in this script
33: """
34:
35: # The populated dictionary in which the park and recreational facilities that
36: # appear in the park property shapefile (polygons), park locations shapefile (points)
37: # and inventory (excel file) are organized (see python script "Link_data_together.py")
38: # The dictionary is called "parks_and_recreation"
39:
40: # The sub-dictionary "clean data" in which the cleaned attribute information is
41: # stored.
42:
43:
44: """
45:     2. OUTPUT FILES
46:     edit the sub-dictionary to the "parks_and_recreation" dictionary called "clean data"
47: """
48:
49: # Shapefile (polygons) that contains a more complete set of park and recreational
50: # properties and the edits that were made to each feature (the identifiers are not
51: # unique).
52: properties_final_shp = r'parks\parkproperty\ParkProperty_final.shp'
53:
54: # Shapefile (polygons) that contains a more complete set of park and recreational
55: # properties and the edits that were made to each feature (the identifiers are not
56: # unique).
57: locations_final_shp = r'parks\park_locations\Park_Locations_final.shp'
58: locations_final_lyr = locations_final_shp[:-4]
59:
60: locations_final_developed_shp = r'parks\park_locations\Park_Locations_final_developed.shp'
61: locations_final_developed_lyr = locations_final_developed_shp[:-4]

```

extract_clean_files.py

```

62: locations_final_private_shp = r'parks\park_locations\Park_Locations_final_private.shp'
63: locations_final_public_shp = r'parks\park_locations\Park_Locations_final_public.shp'
64:
65: # ALL parks listed in the inventory of the Department of Park and Recreation
66: # Source: Michael Kirschman, Deputy Director of Mecklenburg County's DPR shared it
67: # with me
68: park_inventory_final_txt = r'parks\Parks and Amenities_Inventory_final.txt'
69:
70: # Shapefile (polygons) that contains commercial impervious surfaces without a label
71: # and OSM features that coincide with leisure activities, where features are grouped
72: # if they fall within the same tax parcel and that do not fall within a feature of
73: # the DPR's park property dataset (to get rid of duplicates) and that have an "owner"
74: # label that coincides with accessible recreational activities.
75: additional_properties_original_shp = r'parks\parkproperty\Additional_ParkProperty.shp'
76: additional_properties_shp = r'temp\Additional_ParkProperty.shp'
77:
78: """
79:     13. EXPORT EXCEL
80:     certain parks have a webpage on the DPR's webpage with detailed information.
81: """
82:
83: # Certain park or recreational facilities should be removed from the final dataset
84: # because they are duplicates of other records or because they cannot be located
85: # on a map.
86:
87: park_skips = [ 'catw', 'edbg', 'hcr', 'irwgn', 'ltscn1', 'lgcgn2',
88:               'recgne', 'tbyc_p3', 'ttcg1', 'palc', 'mtdl']
89: # The record in the inventory with the Abbv "CATW" is a historic site on the white water center
90: # The record in the inventory with the Abbv "EDBG" is a duplicate
91: # The record in the inventory with the Abbv "HCRG" is a creek that cannot be found
92: # The record in the inventory with the Abbv "IRWGN" is a duplicate
93: # The record in the inventory with the Abbv "LTSCN1" is a duplicate
94: # The record in the inventory with the Abbv "LGCN2" is a duplicate
95: # The record in the inventory with the Abbv "RECGNE" is a duplicate
96: # The record in the inventory with the Abbv "TBYC_P3" is a duplicate
97: # The record in the inventory with the Abbv "TTCG1" is a duplicate
98: # The record in the inventory with the Abbv "PALT" is a duplicate of the Police Athletic Property
99: # The record in the property and location file with id "MLTD" seems to be a trailor park
100: # which we do not need for this research
101:
102: arcpy.CreateFeatureclass_management (workfolder, properties_final_shp, 'POLYGON', '', '', '', sr_nc)
103: arcpy.CreateFeatureclass_management (workfolder, locations_final_shp, 'MULTIPOINT', '', '', '',
104:                                     sr_nc)
105: clean_attributes = ['unique_id', 'name', 'type', 'status', 'address', 'managed_by', 'size', 'age',
106:                   'amenities', 'notes', 'weblink']
107: for clean_attribute in clean_attributes:
108:     if clean_attribute == 'amenities':
109:         attribute_specs = []
110:         for clean_amenity_10char in clean_amenity_fields_10char:
111:             if clean_amenity_10char.endswith('id'): attribute_specs += [[clean_amenity_10char,
112: "TEXT"]]
113:             else: attribute_specs += [[clean_amenity_10char, "DOUBLE"]]
114:         attribute_specs += [['am_total', 'DOUBLE']]
115:     elif clean_attribute in ['size', 'age']:
116:         attribute_specs = [[clean_attribute, "DOUBLE"]]
117:     else: attribute_specs = [[clean_attribute, "TEXT"]]
118: for attribute_name, attribute_type in attribute_specs:
119:     arcpy.AddField_management(properties_final_shp, attribute_name, attribute_type)
120:     arcpy.AddField_management(locations_final_shp, attribute_name, attribute_type)

```

```

120: # Create an empty text file
121: text = open(park_inventory_final_txt, 'w')
122:
123: # Write the headers to the textfile
124: text.write('\t'.join([f.name for f in arcpy.ListFields(properties_final_shp)][3:]))
125:
126: attribute_table_polygons = arcpy.InsertCursor(properties_final_shp)
127: attribute_table_points = arcpy.InsertCursor(locations_final_shp)
128:
129: # Write each park or recreational facility and its attributes
130: for unique_id, attributes in parks_and_recreation.items():
131:     clean_data = attributes['clean_data']
132:     if unique_id not in park_skips + centers_within_a_park:
133:         feature_polygon = attribute_table_polygons.newRow()
134:         feature_point = attribute_table_points.newRow()
135:         values = [unique_id]
136:         feature_polygon.setValue('unique_id', unique_id)
137:         feature_point.setValue('unique_id', unique_id)
138:         for clean_attribute in clean_attributes[1:]:
139:             if clean_attribute == 'amenities':
140:                 for clean_amenity_10char in clean_amenity_fields_10char + ['am_total']:
141:                     value = clean_data[clean_attribute][clean_amenity_10char]
142:                     feature_polygon.setValue(clean_amenity_10char, value)
143:                     feature_point.setValue(clean_amenity_10char, value)
144:                 values += [value]
145:             else:
146:                 value = clean_data[clean_attribute]
147:                 feature_polygon.setValue(clean_attribute, value)
148:                 feature_point.setValue(clean_attribute, value)
149:                 values += [value]
150:             polygon = clean_data['property']
151:             if polygon:
152:                 feature_polygon.shape = polygon
153:                 attribute_table_polygons.insertRow(feature_polygon)
154:             point = clean_data['entrance']
155:             if point:
156:                 feature_point.shape = point
157:                 attribute_table_points.insertRow(feature_point)
158:             text.write('\n' + '\t'.join(map(str, values)))
159:
160: # Close the text file
161: text.close()
162: del attribute_table_polygons
163: del attribute_table_points
164:
165: arcpy.CopyFeatures_management (additional_properies_original_shp, additional_properies_shp)
166: arcpy.AddGeometryAttributes_management(additional_properies_shp, 'AREA_GEODESIC', '', 'ACRES')
167:
168: additional_property_fields = [f.name for f in arcpy.ListFields(additional_properies_shp)][3:]
169: attribute_table = arcpy.da.SearchCursor(additional_properies_shp, additional_property_fields +
    ['SHAPE@', 'SHAPE@XY'])
170: attribute_table_polygons = arcpy.InsertCursor(properties_final_shp)
171: attribute_table_points = arcpy.InsertCursor(locations_final_shp)
172:
173: fid = 1
174: for feature in attribute_table:
175:     feature_polygon = attribute_table_polygons.newRow()
176:     feature_point = attribute_table_points.newRow()
177:     for index, attribute in enumerate(additional_property_fields):
178:         value = feature[index]
179:         if attribute in ['source', 'pid', 'rec_fitnes']: continue

```

 extract_clean_files.py

```

180:         elif attribute == 'multipurp': attribute = 'multiprpse'
181:         elif attribute == 'AREA_GEO': attribute = 'size'
182:         elif attribute == 'prpty_type':
183:             attribute = 'type'
184:             if value == 'Park':
185:                 feature_polygon.setValue('managed_by', 'Private')
186:                 feature_point.setValue('managed_by', 'Private')
187:             else:
188:                 feature_polygon.setValue('managed_by', 'HOA')
189:                 feature_point.setValue('managed_by', 'HOA')
190:                 feature_polygon.setValue('status', 'Developed')
191:                 feature_point.setValue('status', 'Developed')
192:         elif attribute == 'yearbuilt':
193:             attribute = 'age'
194:             value = None if value == '0' else 2015 - int(value)
195:             feature_polygon.setValue(attribute, value)
196:             feature_point.setValue(attribute, value)
197:         unique_id = 'add_%s' % fid
198:         feature_point.setValue('unique_id', unique_id)
199:         feature_point.shape = arcpy.Point(*feature[-1])
200:         attribute_table_points.insertRow(feature_point)
201:         feature_polygon.setValue('unique_id', unique_id)
202:         feature_polygon.shape = feature[-2]
203:         attribute_table_polygons.insertRow(feature_polygon)
204:         fid += 1
205:
206: del attribute_table
207: del attribute_table_polygons
208: del attribute_table_points
209:
210:
211: # Public parks
212: locations_final_developed_lyr
213: arcpy.MakeFeatureLayer_management(locations_final_shp, locations_final_lyr)
214: query = """status" = 'Developed'"""
215: arcpy.SelectLayerByAttribute_management(locations_final_lyr, "NEW_SELECTION", query)
216: arcpy.CopyFeatures_management(locations_final_lyr, locations_final_developed_shp)
217:
218: arcpy.MakeFeatureLayer_management(locations_final_developed_shp, locations_final_developed_lyr)
219: query = """type" = 'Residential' OR "type" = 'Park'"""
220: arcpy.SelectLayerByAttribute_management(locations_final_developed_lyr, "NEW_SELECTION", query)
221: arcpy.CopyFeatures_management(locations_final_developed_lyr, locations_final_private_shp)
222: arcpy.SelectLayerByAttribute_management(locations_final_developed_lyr, "SWITCH_SELECTION")
223: query = """type" = 'Special Facility' OR "type" = 'Historic Site'"""
224: arcpy.SelectLayerByAttribute_management(locations_final_developed_lyr, "REMOVE_FROM_SELECTION",
    query)
225: arcpy.CopyFeatures_management(locations_final_developed_lyr, locations_final_public_shp)

```

12. Generate Crime Location Points

```

crime.py

1: # Author:                               Coline C. Dony
2: # Date first written:                   May 8, 2016
3: # Date last updated:                   May 28, 2016
4: # Purpose:                             Extract crimes in Mecklenburg County
5: #
6: # Problem statement:
7: # Scrape all crimes in Mecklenburg County
8: #
9:
10: """
11:     1. GENERATE INPUT FILES
12:     collect crime reports in Mecklenburg County for the years 2014 and 2015
13: """
14:
15: # Execute script that scrapes crime reports from spotcrime.com
16: execfile(r'crime\scrape_crime_data.py')
17:
18: # Execute script that scrapes the geocodes of each crime report from spotcrime.com
19: execfile(r'crime\scrape_geocodes_of_crimes.py')
20:
21: """
22:     1. INPUT FILES
23:     original, already existing files, used in this script
24: """
25:
26: # All criminal activity reported by the Mecklenburg Police Department that appears
27: # on the webpage "spotcrime.com"
28: # Source: http://spotcrime.com/nc/charlotte/daily
29: crime_original_txt = r'crime\Mecklenburg_Crimes_2014_2015_geocoded.txt'
30:
31: """
32:     2. TEMPORARY FILES
33:     files that are generated in this script but are not the end product / output file.
34:     This segment also presents the thought process of the cleaning and editing process.
35: """
36:
37: # All cadastral tax parcel boundaries (polygons) from the planning department
38: # Source: http://maps.co.mecklenburg.nc.us/openmapping/data.html
39: crime_shp = r'temp\Mecklenburg_Crimes_2014_2015.shp'
40:
41: # For certain functions, some shapefiles will be converted to layer files
42: crime_lyr = crime_shp[:-4]
43:
44: """
45:     3. OUTPUT FILE
46:     additional park and recreational properties from tax parcel data and OSM data
47: """
48:
49: crime_violent_shp = r'temp\Mecklenburg_Crimes_violent.shp'
50: crime_violent_prj_shp = r'crime\Mecklenburg_Crimes_violent_prj.shp'
51: # For certain functions, some shapefiles will be converted to layer files
52: crime_violent_prj_lyr = crime_violent_prj_shp[:-4]
53:
54: """
55:     4. SOLVE PROBLEM 1
56:     extract OSM leisure polygons
57: """
58:
59: crime_data = [crime.strip().split('\t') for crime in open(crime_original_txt)]
60:
61: # Generate an empty shapefile (polygons) in which all features associated with potential

```

crime.py

```

62: # recreational activities (OSM and impervious surfaces) will be inserted
63: arcpy.CreateFeatureclass_management(workfolder, crime_shp, 'POINT', '', '', 'sr_wgs')
64:
65: # Add three empty attributes to the empty shapefile (polygons)
66: # NOTE: the source of each feature (OSM or impervious) appears in the source attribute
67: # and the Leisure Label that OSM gives to each feature is in the OSM_Label attribute
68: crime_fields = ['case_numbr', 'type', 'day', 'month', 'year', 'hour', 'minute', 'meridiem',
69:                 'address', 'details', 'source']
70: for fieldname in crime_fields:
71:     if fieldname in ['day', 'month', 'year', 'hour', 'minute']: arcpy.AddField_management(crime_shp,
72:                                             fieldname, 'SHORT')
73:     else: arcpy.AddField_management(crime_shp, fieldname, 'TEXT')
74:
75: attribute_table = arcpy.da.InsertCursor(crime_shp, crime_fields + ['SHAPE@XY'])
76: for crime_data[1:]:
77:     if len(crime) == 12: crime = [''] + crime
78:     case, crime_type, day, month, year, hour, minute, meridiem, address, lat, lng, details, source
79:     = crime
80:     geometry = (float(lng), float(lat))
81:     hour = 99 if hour == '' else hour
82:     minute = 99 if minute == '' else minute
83:     attribute_table.insertRow([case, crime_type, int(day), int(month), int(year), int(hour),
84:                               int(minute), meridiem, address, details, source, geometry])
85:
86: del attribute_table
87: del crime_data
88:
89: arcpy.MakeFeatureLayer_management(crime_shp, crime_lyr)
90: crime_types = ['Arrest', 'Arson', 'Assault', 'Shooting', 'Vandalism']
91: query = ' OR '.join(["type" = '%s'" % crime for crime in crime_types])
92: arcpy.SelectLayerByAttribute_management(crime_lyr, 'NEW_SELECTION', query)
93: arcpy.CopyFeatures_management(crime_lyr, crime_violent_shp)
94: arcpy.Project_management(crime_violent_shp, crime_violent_prj_shp, sr_nc)
95:
96: # Select violent crime within distance of each park
97:
98: # Shapefile (polygons) that contains a more complete set of park and recreational
99: # properties and the edits that were made to each feature (the identifiers are not
100: # unique).
101: locations_final_shp = r'parks\park_locations\Park_Locations_final.shp'
102: arcpy.AddField_management(locations_final_shp, 'crime_vlnt', 'DOUBLE')
103: attribute_table = arcpy.UpdateCursor(locations_final_shp)
104: crime_max = 0
105: for park in attribute_table:
106:     locations_final_lyr = locations_final_shp[:-4]
107:     query = "FID" = %s" % park.fid
108:     arcpy.MakeFeatureLayer_management(locations_final_shp, locations_final_lyr, query)
109:     arcpy.MakeFeatureLayer_management(crime_violent_prj_shp, crime_violent_prj_lyr)
110:     arcpy.SelectLayerByLocation_management(crime_violent_prj_lyr, 'WITHIN_A_DISTANCE',
111:     locations_final_lyr, '200 Meters')
112:     crime_count = int(arcpy.GetCount_management(crime_violent_prj_lyr).getOutput(0))
113:     park.crime_vlnt = crime_count
114:     if crime_count > crime_max: crime_max = crime_count
115:     attribute_table.updateRow(park)
116: del attribute_table

```

13. Scrape Crime Data in Mecklenburg County

scrape_crime_data.py

```

1: import urllib2
2: import datetime
3: import time
4:
5: numdays = 365 * 2
6: date = datetime.date(2014, 1, 1)
7: textfile = open(r'C:\Dissertation_Materials\Data\core_datasets\Mecklenburg_Crimes_2014_2015.txt',
8: 'w')
9: textfile.write('Type\tDay\tMonth\tYear\tHour\tMinute\tMeridiem\tAddress\tDetails\n')
10: while date < datetime.date(2016, 1, 1):
11:     try:
12:         url = r'http://spotcrime.com/nc/charlotte/daily/%s-%s-%s' % (date.year, date.month, date.day)
13:         response = urllib2.urlopen(url).read().split('\n')
14:         for line in response:
15:             if r'#000000' in line:
16:                 crime_data_raw = line.split('>')
17:                 crime_data_clean = []
18:                 for data in crime_data_raw:
19:                     if data.endswith('</td') and len(data) > 5:
20:                         crime_data_clean.append(data[:-4])
21:                     elif data.startswith('<a')':
22:                         crime_data_clean.append(r'http://spotcrime.com' + data[9:-17])
23:                 crime_type = crime_data_clean[0]
24:                 crime_day = crime_data_clean[1][3:5]
25:                 crime_month = crime_data_clean[1][:2]
26:                 crime_year = '20' + crime_data_clean[1][6:8]
27:                 crime_hour = crime_data_clean[1][10:12]
28:                 crime_minute = crime_data_clean[1][13:15]
29:                 crime_meridiem = crime_data_clean[1][16:18]
30:                 crime_location = crime_data_clean[2]
31:                 crime_details = crime_data_clean[3]
32:                 textfile.write('%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\n' % (crime_type, crime_day,
33: crime_month, crime_year, crime_hour, crime_minute, crime_meridiem, crime_location, crime_details))
34:                 print "%s done!" % date
35:                 date += datetime.timedelta(days=1)
36:                 time.sleep(5)
37:             except:
38:                 print "%s skipped!" % date
39:                 dates_skipped.append(date)
40:                 date += datetime.timedelta(days=1)
41: textfile.close()
42:

```



```

1: import urllib2
2: import time
3:
4: textfile_in =
5:     open(r'C:\Users\cdony\Dropbox\Dissertation_Materials\Data\core_datasets\Mecklenburg_Crimes_Jan2014_Dec20
6:         'a')
7:
8: for crime_index, crime in enumerate(textfile_in):
9:     crime_data = crime.strip().split('\t')
10:    url = crime_data[-1]
11:    wait = 1
12:    while True:
13:        try:
14:            url_open = urllib2.urlopen(url)
15:            print 'wait-time: %s (level 1)' % wait
16:            break
17:        except:
18:            url_open.close()
19:            time.sleep(wait)
20:            wait *= 2
21:    response = url_open.read().split('\n')
22:    crime_details = []
23:    for line in response:
24:        if r'itemprop' in line:
25:            line_list = line.split('"')
26:            detail_key = line_list[1]
27:            if detail_key in ['description', 'latitude', 'longitude']:
28:                if detail_key == 'description': crime_description = line_list[2][1:-5]
29:                elif detail_key == 'latitude': crime_lat = line_list[3]
30:                elif detail_key == 'longitude': crime_lng = line_list[3]
31:            if r'Case number' in line:
32:                line_list = line.split(' ')
33:                crime_case_ID = line_list[3][:-1]
34:    crime_type = crime_data[0]
35:    crime_day = crime_data[1]
36:    crime_month = crime_data[2]
37:    crime_year = crime_data[3]
38:    crime_hour = crime_data[4]
39:    crime_minute = crime_data[5]
40:    crime_meridien = crime_data[6]
41:    crime_location = crime_data[7]
42:    crime_details = crime_data[8]
43:
44:    textfile_out.write('%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\n' % (crime_case_ID,
45:        crime_type, crime_day, crime_month, crime_year, crime_hour, crime_minute, crime_meridien,
46:        crime_location, crime_lat, crime_lng, crime_description, crime_details))
47:    print "%.5f %% completed (at %s-%s-%s)\n" % (float(crime_index) / len(textfile_in), crime_month,
48:        crime_day, crime_year)
49:
50: textfile_out.close()

```

15. Clean Quality of Life

clean_quality_of_life.py

```

1: qol_original_shp = r'quality_of_life\NPA_2014_meck.shp'
2: qol_shp = r'temp\NPA_2014_meck.shp'
3: qol_data_p1_txt = r'quality_of_life\QOL Dashboard Data Download File Jan 2016 Data Part 1.txt'
4: qol_data_p2_txt = r'quality_of_life\QOL Dashboard Data Download File Jan 2016 Data Part 2.txt'
5:
6: qol_data_p1 = open(qol_data_p1_txt).readlines()
7: fieldnames_p1 = qol_data_p1[0].strip().split('\t')
8: for i in range(len(fieldnames_p1)):
9:     fieldname = fieldnames_p1[i]
10:    if len(fieldname) > 10: fieldnames_p1[i] = fieldname.replace('_20', '_')
11: fieldnames_p1_long = qol_data_p1[1].strip().split('\t')
12:
13: qol_data_p1 = [data.strip().split('\t') for data in qol_data_p1[2:]]
14: for i in range(len(qol_data_p1)):
15:     if len(qol_data_p1[i]) < len(fieldnames_p1): qol_data_p1[i] += ['', '']
16: qol_data_p1 = {data[0]:dict(zip(fieldnames_p1, data)) for data in qol_data_p1}
17:
18:
19: qol_data_p2 = open(qol_data_p2_txt).readlines()
20: fieldnames_p2 = qol_data_p2[0].strip().split('\t')
21: for i in range(len(fieldnames_p2)):
22:     fieldname = fieldnames_p2[i]
23:     if len(fieldname) > 10: fieldnames_p2[i] = fieldname.replace('_20', '_')
24: fieldnames_p2_long = qol_data_p2[1].strip().split('\t')
25:
26: qol_data_p2 = [data.strip().split('\t') for data in qol_data_p2[2:]]
27: for i in range(len(qol_data_p2)):
28:     if len(qol_data_p2[i]) < len(fieldnames_p2): qol_data_p2[i] += ['', '', '', '', '', '']
29: qol_data_p2 = {data[0]:dict(zip(fieldnames_p2, data)) for data in qol_data_p2}
30: qol_data = qol_data_p1.copy()
31: for npa_id, data in qol_data.items():
32:     try: qol_data[npa_id].update(qol_data_p2[npa_id])
33:     except: continue
34:
35: arcpy.CopyFeatures_management(qol_original_shp, qol_shp)
36: for fieldname in fieldnames_p1[1:] + fieldnames_p2[1:]: arcpy.AddField_management(qol_shp, fieldname,
    'DOUBLE')
37:
38: attribute_table = arcpy.UpdateCursor(qol_shp)
39: for feature in attribute_table:
40:     npa_id = feature.NPA
41:     for fieldname in fieldnames_p1[1:] + fieldnames_p2[1:]:
42:         value = qol_data[str(npa_id)][fieldname]
43:         try: feature.setValue(fieldname, float(value))
44:         except: continue
45:     attribute_table.updateRow(feature)
46:
47: del attribute_table
48:

```


16. Extend Survey Data with Location Information

```

                                extend_survey_data.py
1: # Author:                      Coline C. Dony
2: # Date first written:          April 12, 2016
3: # Date last updated:           May 23, 2016
4: # Purpose:                     Create a unique identifier for park locations
5:
6: """
7:     1. INPUT FILES
8:     original, already existing files, used in this script
9: """
10:
11: # Shapefile (points) that contains all property entrances whether developed or
12: # undeveloped, and managed by the Department of Park and Recreation (DPR)
13: # Source: http://maps.co.mecklenburg.nc.us/openmapping/data.html
14: survey_data_txt = r'park_survey\survey_locations.txt'
15: survey_data_extended_txt = r'park_survey\survey_locations_extended.txt'
16: survey_data_shp = r'temp\survey_locations.shp'
17: survey_data_prj_shp = r'park_survey\survey_locations_prj.shp'
18: survey_data_prj_lyr = survey_data_prj_shp[:-4]
19:
20: mecklenburg_boundary_shp =
21:     'census\counties\Mecklenburg_county\mecklenburgcounty_boundary\MecklenburgCounty_Boundary.shp'
22: zipcodes_shp = 'boundaries\zipcode\Zipcode.shp'
23: zipcodes_lyr = zipcodes_shp[:-4]
24: npa_shp = 'quality_of_life\NPA_2014_meck.shp'
25: npa_lyr = npa_shp[:-4]
26:
27: arcpy.MakeXYEventLayer_management(survey_data_txt, 'lng', 'lat', survey_data_shp, sr_wgs)
28: arcpy.Project_management(survey_data_shp, survey_data_prj_shp, sr_nc)
29:
30: arcpy.AddField_management(survey_data_prj_shp, 'in_meck', 'TEXT')
31:
32: arcpy.MakeFeatureLayer_management(survey_data_prj_shp, survey_data_prj_lyr)
33: arcpy.SelectLayerByLocation_management(survey_data_prj_lyr, 'intersect', mecklenburg_boundary_shp)
34: attribute_table = arcpy.UpdateCursor(survey_data_prj_lyr)
35: for survey in attribute_table:
36:     survey.in_meck = 'YES'
37:     attribute_table.updateRow(survey)
38: del attribute_table
39:
40: arcpy.AddField_management(survey_data_prj_shp, 'in_zipcode', 'TEXT')
41:
42: attribute_table = arcpy.UpdateCursor(survey_data_prj_shp)
43: for survey in attribute_table:
44:     fid = survey.fid
45:     query = """fid" = %s""" % fid
46:     arcpy.MakeFeatureLayer_management(survey_data_prj_shp, survey_data_prj_lyr, query)
47:     arcpy.MakeFeatureLayer_management(zipcodes_shp, zipcodes_lyr)
48:     arcpy.SelectLayerByLocation_management(zipcodes_lyr, 'intersect', survey_data_prj_lyr)
49:     attribute_table_lyr = arcpy.da.SearchCursor(zipcodes_lyr, 'zip')
50:     try: zipcode = attribute_table_lyr.next()[0]
51:     except: continue
52:     survey.in_zipcode = zipcode
53:     del attribute_table_lyr
54:     attribute_table.updateRow(survey)
55: del attribute_table
56:
57: arcpy.AddField_management(survey_data_prj_shp, 'in_npa', 'TEXT')
58: attribute_table = arcpy.UpdateCursor(survey_data_prj_shp)
59: for survey in attribute_table:
60:     fid = survey.fid

```

 extend_survey_data.py

```

61:     query = """fid" = %s""" % fid
62:     arcpy.MakeFeatureLayer_management(survey_data_prj_shp, survey_data_prj_lyr, query)
63:     arcpy.MakeFeatureLayer_management(npa_shp, npa_lyr)
64:     arcpy.SelectLayerByLocation_management(npa_lyr, 'intersect', survey_data_prj_lyr)
65:     attribute_table_lyr = arcpy.da.SearchCursor(npa_lyr, 'NPA')
66:     try: npa = attribute_table_lyr.next()[0]
67:     except: continue
68:     survey.in_npa = npa
69:     del attribute_table_lyr
70:     attribute_table.updateRow(survey)
71: del attribute_table
72:
73:
74: field_names = map(str, [f.name for f in arcpy.ListFields(survey_data_prj_shp)])[2:]
75: out_text = open(survey_data_extended_txt, 'w')
76: out_text.write('\t'.join(field_names) + '\n')
77: attribute_table = arcpy.da.SearchCursor(survey_data_prj_shp, '*')
78: for survey in attribute_table:
79:     survey = map(str, survey)[2:]
80:     out_text.write('\t'.join(survey) + '\n')
81: del attribute_table
82: out_text.close()
83:
84:
85:

```

17. Calculate Park Densities and Convert to Contours

calculate_park_densities_and_convert_to_contours.py

```

1: # Shapefile (polygons) that contains a more complete set of park and recreational
2: # properties and the edits that were made to each feature (the identifiers are not
3: # unique).
4: locations_final_shp = r'parks\park_locations\Park_Locations_final.shp'
5:
6: locations_all_shp = r'C:\Dissertation_Materials\Results\park_densities\Park_Locations_all.shp'
7:
8: locations_non_public_shp =
9:     r'C:\Dissertation_Materials\Results\park_densities\Park_Locations_non_public.shp'
10: locations_public_shp = r'C:\Dissertation_Materials\Results\park_densities\Park_Locations_public.shp'
11:
12: locations_final_lyr = locations_final_shp[:-4]
13: locations_all_lyr = locations_all_shp[:-4]
14:
15:
16: # Extract features (polygons) from the OSM data that are associated with Leisure
17: query = """status = 'Developed'"""
18: arcpy.MakeFeatureLayer_management(locations_final_shp, locations_final_lyr, query)
19: arcpy.CopyFeatures_management(locations_final_lyr, locations_all_shp)
20:
21: number_parks = int(arcpy.GetCount_management(locations_final_lyr).getOutput(0))
22: for field in ['size', 'am_total', 'crime_vlnt']:
23:     rank_field = 'rank_' + field[:5]
24:     arcpy.AddField_management(locations_all_shp, rank_field, 'DOUBLE')
25:     order = 'D' if field.startswith('crime') else 'A'
26:     attribute_table = arcpy.UpdateCursor(locations_all_shp, sort_fields="%s %s" % (field, order))
27:     for index, park in enumerate(attribute_table):
28:         rank = (index / (number_parks / 4))
29:         if rank >= 3: rank = 1.75
30:         elif rank == 2: rank = 1.25
31:         elif rank == 1: rank = 0.75
32:         else: rank = 0.25
33:         park.setValue(rank_field, rank)
34:         attribute_table.updateRow(park)
35:     del attribute_table
36:
37: arcpy.AddField_management(locations_all_shp, 'attraction', 'DOUBLE')
38: attribute_table = arcpy.UpdateCursor(locations_all_shp)
39: for park in attribute_table:
40:     park.attraction = (park.rank_size + (2 * park.rank_am_to) + (3 * park.rank_crime)) / 6
41:     attribute_table.updateRow(park)
42: del attribute_table
43:
44: # Extract features (polygons) from the OSM data that are associated with Leisure
45: query = """type = 'Residential'"""
46: arcpy.MakeFeatureLayer_management(locations_all_shp, locations_all_lyr, query)
47: arcpy.CopyFeatures_management(locations_all_lyr, locations_non_public_shp)
48:
49: # Extract features (polygons) from the OSM data that are associated with Leisure
50: query = """type <> 'Residential'"""
51: arcpy.MakeFeatureLayer_management(locations_all_shp, locations_all_lyr, query)
52: arcpy.CopyFeatures_management(locations_all_lyr, locations_public_shp)
53:
54: for park_type in ['public', 'non_public', 'all']:
55:     for population in ['NONE', 'size', 'am_total', 'crime_vlnt', 'attraction']:
56:         in_features = vars()[f'locations_{park_type}_shp'] % park_type
57:         if population in ['NONE', 'attraction']: population_field = population
58:         else: population_field = 'rank_' + population[:5]
59:         kernel_density_result = arcpy.sa.KernelDensity(in_features, population_field, '100', '',
60:             'SQUARE_MILES')

```

calculate_park_densities_and_convert_to_contours.py

```
60:     out_raster = r'C:\Dissertation_Materials\Results\park_densities\KD_%s_%s' % (park_type[0],
population_field[:7])
61:     kernel_density_result.save(out_raster)
62:     # Replace a layer/table view name with a path to a dataset (which can be a layer file) or
create the layer/table view within the script
63:     # The following inputs are layers or table views: "res_park7"
64:     out_contour =
r'C:\Dissertation_Materials\Results\park_densities\contours_density_locations_%s_%s' % (park_type,
population_field)
65:     arcpy.sa.Contour(out_raster, out_contour, "1")
```


APPENDIX J: COLLECT REVIEWS

1. Collect Google Reviews

```

google places reviews_to_share.py

1: ## Script Title: Printing highest rated sushi place on Google
2: ## Author(s): CoDo
3: ## Date: December 7, 2015
4:
5: # Google API key (get it at https://code.google.com/apis/console)
6: google_APIkey = '<Your API Key here>'
7:
8: # Environment setup: the C:\Temp folder is set as default directory
9: # Two folders are created (tips and tags), in which a textfile per venue will be stored
10: # that contains all tips / tags associated with the venue
11: google_folder = os.path.join(data_folder, r'reviews\google')
12:
13: # This information will be stored in a text file (TAB-delimited)
14: input_filename_places = os.path.join(google_folder, r'Google_Parks.txt')
15: input_file_places = open(input_filename_places)
16: input_file_places.next()
17:
18: output_filename_places_details = os.path.join(google_folder, r'Google_Parks_details.txt')
19: if not os.path.exists(output_filename_places_details):
20:     output_file_places_details = open(output_filename_places_details, 'w')
21:     # Headers for the data are added to the textfile
22:     output_file_places_details.write("\t".join( ['place_id', 'place_name', 'street', 'city',
23: 'state', 'zipcode', 'country',
24: 'place_lat', 'place_lng', 'formatted_address',
25: 'formatted_phone_number', 'place_rating', 'total_rating', 'park',
26: 'establishment', 'point_of_interest', 'school', 'parking',
27: 'zoo', 'university',
28: 'place_of_worship', 'rv_park', 'spa', 'campground',
29: 'cemetery', 'church',
30: 'gym', 'health']) + "\n")
31:     output_file_places_details.close()
32: else:
33:     num_lines = open(output_filename_places_details).read().count('\n')
34:     for i in range(num_lines - 1): input_file_places.next()
35:
36: # Loop that goes through each element in the results dictionary of the url's source code
37: for place in input_file_places:
38:     place_id = place.split('\t')[0]
39:
40:     # Read the response url of our request to search sushi restaurants in the Charlotte area
41:     url_address = 'https://maps.googleapis.com/maps/api/place/details/json?placeid=%s&key=%s' %
42: (place_id, google_APIkey)
43:     url_sourceCode = urllib2.urlopen(url_address).read()
44:
45:     # Convert the url's source code from a string to a json format (i.e. dictionary type)
46:     google_place_status = json.loads(url_sourceCode)['status']
47:     if google_place_status == 'NOT_FOUND':
48:         # SAVE INFORMATION TO TEXTFILE
49:         output_file_places_details = open(output_filename_places_details, 'a')
50:         output_file_places_details.write("%s\n" % place_id)
51:         output_file_places_details.close()
52:         continue
53:     output_file_places_details.close()
54:     google_place_details = json.loads(url_sourceCode)['result']
55:
56:     # Place Details
57:     formatted_address = google_place_details['formatted_address']
58:     address_parts = google_place_details['formatted_address'].split(',')
59:     address_parts = [part for part in address_parts if len(part) > 0]
60:     if len(address_parts) == 5:
61:         number, street, city, state_zipcode, country = address_parts

```

google places reviews_to_share.py

```

57:     street = '%s %s' % (number, street)
58:     elif len(address_parts) == 4: street, city, state_zipcode, country = address_parts
59:     else: street, city, state_zipcode, country = [''] + address_parts
60:     index_zipcode = state_zipcode.find('2')
61:     state, zipcode = state_zipcode[:index_zipcode].strip(), state_zipcode[index_zipcode:].strip()
62:     try: formatted_phone_number = google_place_details['formatted_phone_number']
63:     except: formatted_phone_number = ''
64:     place_lat = google_place_details['geometry']['location']['lat']
65:     place_lng = google_place_details['geometry']['location']['lng']
66:     place_name = google_place_details['name']
67:     place_id = google_place_details['place_id']
68:     try: place_rating = google_place_details['rating']
69:     except: place_rating = 'N/A'
70:     try: place_ratings = google_place_details['user_ratings_total']
71:     except: place_ratings = 0
72:     place_categories = google_place_details['types']
73:     park = 1 if 'park' in place_categories else 0
74:     establishment = 1 if 'establishment' in place_categories else 0
75:     point_of_interest = 1 if 'point_of_interest' in place_categories else 0
76:     school = 1 if 'school' in place_categories else 0
77:     parking = 1 if 'parking' in place_categories else 0
78:     zoo = 1 if 'zoo' in place_categories else 0
79:     university = 1 if 'university' in place_categories else 0
80:     place_of_worship = 1 if 'place_of_worship' in place_categories else 0
81:     rv_park = 1 if 'rv_park' in place_categories else 0
82:     spa = 1 if 'spa' in place_categories else 0
83:     campground = 1 if 'campground' in place_categories else 0
84:     cemetery = 1 if 'cemetery' in place_categories else 0
85:     church = 1 if 'church' in place_categories else 0
86:     gym = 1 if 'gym' in place_categories else 0
87:     health = 1 if 'health' in place_categories else 0
88:
89:     # SAVE INFORMATION TO TEXTFILE
90:     output_file_places_details = open(output_filename_places_details, 'a')
91:     output_file_places_details.write("\t".join(    map(str, [place_id, place_name, street.strip(),
city.strip(), state.strip(), zipcode, country,
92:         place_lat, place_lng, formatted_address, formatted_phone_number,
place_rating, place_ratings, park,
93:         establishment, point_of_interest, school, parking, zoo, university,
94:         place_of_worship, rv_park, spa, campground, cemetery, church,
95:         gym, health]))) + "\n")
96:     output_file_places_details.close()
97:
98:     # Save reviews
99:     output_filename_place_reviews = os.path.join(google_folder, r'google_reviews',
r'Google_Parks_%s.txt' % place_name.replace(r'/', ' '))
100:     output_file_place_reviews = open(output_filename_place_reviews, 'w')
101:     output_file_place_reviews.write("\t".join(    ['place_id', 'place_name', 'reviewer_place_rating',
'review_time', 'review_message',
102:         'reviewer_overall_rating', 'review_language',
'reviewer_name', 'reviewer_url'])) + "\n")
103:     output_file_place_reviews.close()
104:     try:
105:         for review in google_place_details['reviews']:
106:             reviewer_overall_rating = review['aspects'][0]['rating']
107:             reviewer_name = review['author_name']
108:             try: reviewer_url = review['author_url']
109:             except: reviewer_url = ''
110:             review_language = review['language']
111:             reviewer_place_rating = review['rating']
112:             review_message = review['text']

```

google places reviews_to_share.py

```
113:         review_time = review['time']
114:
115:         # SAVE INFORMATION TO TEXTFILE
116:         # This information will be stored in a text file (TAB-delimited)
117:         output_file_place_reviews = open(output_filename_place_reviews, 'a')
118:         output_file_place_reviews.write("\t".join(map(str,[place_id, place_name,
reviewer_place_rating, review_time, review_message,
119:                                                         reviewer_overall_rating, review_language,
reviewer_name, reviewer_url]))) + "\n")
120:         output_file_place_reviews.close()
121:     except: pass
122:     print 'place saved'
123: # Close written file.
124: input_file_places.close()
```

2. Collect Foursquare Reviews

4square_toshare.py

```

1: ## Script Title:   Scrape Park Locations from Foursquare and also
2: ##               extract comments and tags associated with each park
3: ## Author(s):     CoDo
4: ## Date:          Dec 10, 2015
5:
6: # Import functions from the foursquare and datetime Libraries
7: import foursquare
8: import datetime
9: import os
10:
11: # API identification codes (get keys at https://developer.foursquare.com)
12: client_id = '<your_client_id>'
13: client_secret = '<your_client_secret_id>'
14:
15: # Generate access to the Foursquare API
16: client = foursquare.Foursquare(client_id, client_secret)
17:
18: # PARK SEARCH on FOURSQUARE - APPROACH DESCRIPTION
19:
20: # FOURSQUARE LIMITATIONS:
21: # Extracting information from Foursquare is Limited to 50 Locations (i.e. venues) per search
22: # Therefore, multiple searched are necessary to extract the full list of parks available
23: # on Foursquare in Charlotte (and surroundings).
24:
25: # OVERCOMING THOSE LIMITATIONS:
26: # In this script a bounding box is defined (in WGS84) in which we want to extract all Foursquare
venues
27: # that are categorized as "parks". However, that bounding box contains more than 50 venues (parks).
28: # Therefore, we define smaller bounding boxes in which we separately search for venues. Starting in
the
29: # Lower Left corner of our whole search area, a smaller bounding box is defined, which is the
initial
30: # search area (in which a max. of 50 parks can be extracted). When the initial search is done,
31: # another small bounding box is defined adjacent to the one we just searched (using the
32: # frange() function), and a new search is performed within that box. A search is performed
33: # until all smaller bounding boxes within the whole search area have been searched.
34:
35: # These four coordinates define the bounding box that contain the whole search area we want to
36: # extract park venues from Foursquare
37: lat_lowerLeft, lng_lowerLeft = (34.5498, -81.4884)
38: lat_upperRight, lng_upperRight = (36.0097, -80.1149)
39:
40: # Each smaller bounding box within the search area starts where the previous bounding box ended
41: # (it follows a grid) The Lower Left (SW) and upper right (NE) coordinates defining that
42: # smaller bounding box are determined using equal steps in Latitude and Longitude.
43: lat_steps = (lat_upperRight - lat_lowerLeft)/20
44: lng_steps = (lng_upperRight - lng_lowerLeft)/40
45:
46: # This function generates a floating point number (representing an angle in Latitude or Longitude)
47: # within a certain range of angles. It is similar to the range() function but allows floating points
48: def frange(start_angle, stop_angle, step):
49:     angle = start_angle
50:     while angle < stop_angle:
51:         yield angle
52:         angle += step
53:
54: # PARK SEARCH ON FOURSQUARE - DATA EXTRACTED
55:
56: # DUPLICATE PARKS PROBLEM:
57: # Foursquare provides up to 50 parks within a certain radius of a coordinate. Since we make searches
58: # around coordinates that are relatively close to one another (to make sure we don't miss any
parks),

```

4square_toshare.py

```

59: # it is very likely that the same park is extracted from two different searches.
60:
61: # CHECKING FOR DUPLICATES:
62: # In this code, we keep track of the Foursquare venue identification numbers that are extracted from
63: # every search, by storing those ID's in a list. The IDs are unique (a unique ID for each venue).
64: # When going through each venue extracted from a search, before we add the park to our dataset,
65: # we first check whether this park has been extracted in a previous search by checking if the ID
66: # already appears in the list of IDs.
67:
68: # List which will keep track of the Foursquare venue identification numbers that have already been
   extracted
69: venue_ids = []
70:
71: # STORING DATA
72: # The Foursquare venues that have been extracted from our search, will be stored in a textfile (TAB-
   delimited).
73: # These are the Foursquare venue details that are stored in that textfile:
74: # 1. venue id: A unique string identifier for this venue.
75: # 2. venue name: The best known name for this venue.
76: # 3. venue street: From the Location response (if available).
77: # 4. venue city: From the Location response (if available).
78: # 5. venue state: From the Location response.
79: # 6. venue Latitude: From the Location response.
80: # 7. venue Longitude: From the Location response.
81: # 8. venue checkins compact: Total checkins ever at this venue (from the compact response).
82: # 9. venue checkins complete: Total checkins ever at this venue (from the complete venue response).
83: # 10. venue user count compact: Total users who have ever checked in at this venue (from the
   compact response).
84: # 11. venue checkins complete: Total users who have ever checked in at this venue (from the
   complete venue response).
85: # 12. venue visits: No metadata available.
86: # 13. venue tips: number of tips (comments left) at this venue.
87: # 14. at venue now compact: users who who are currently checked in at this venue (from the compact
   response).
88: # 15. at venue now complete: users who who are currently checked in at this venue (from the
   complete venue response).
89: # 16. day venue created: No metadata available.
90: # 17. venue page updates: No metadata available.
91: # 18. venue dislikes: No metadata available.
92: # 19. venue Likes: No metadata available.
93: # 20. venue webpage: No metadata available.
94: # 21. venue tags: No metadata available.
95: # 22. number of tags for this venue: No metadata available.
96: # 23. venue tips: No metadata available.
97: # 24. number of tips for this venue: No metadata available.
98: # See more detailed metadata about each field at:
   https://developer.foursquare.com/docs/responses/venue
99:
100: # Environment setup: the C:\Temp fllder is set as default directory
101: # Two folders are created (tips and tags), in which a textfile per venue will be stored
102: # that contains all tips / tags associated with the venue
103: os.chdir(r'C:\Temp')
104: os.mkdir('tips')
105: os.mkdir('tags')
106:
107: # This information will be stored in a text file (TAB-delimited)
108: output_filename_venues = r'foursquare_Parks.txt'
109: output_file_venues = open(output_filename_venues, 'w')
110: # Headers for the data are added to the textfile
111: output_file_venues.write("\t".join(    ['day', 'date', 'month', 'year', 'hour', 'minute', 'meridiem',
112:                                         'id_4square', 'venue_name', 'street', 'city', 'state',

```

```

4square_toshare.py
113:         'checkins', 'checkins_d', 'users', 'users_diff', 'visits',
114:         'tipCount',
115:         'likes',
116:         'hereNow', 'hereNow_df', 'dayCreated', 'pageUpdats', 'dislikes',
117:         'url', 'tags', 'number_tags', 'tips', 'number_tips', 'lat',
118:         'lng']) + "\n")
119:
120: # Performs the search for each bounding box within our search area
121: for LAT in frange(lat_lowerLeft, lat_upperRight, lat_steps):
122:     for LNG in frange(lng_lowerLeft, lng_upperRight, lng_steps):
123:         # SENDING SEARCH QUERY THROUGH THE FOURSQUARE API
124:         # Search parameters include the search query, the coordinates defining the
125:         # bounding box, the number of venues to extract and the search type (Browse), "browse"
126:         finds venues
127:         # within a given area instead of only finding venues closest to a point
128:         # (see: https://deveLoper.foursquare.com/docs/venues/search)
129:         responses = client.venues.search(params={'query': 'park', 'sw': '%s,%s' % (LAT, LNG), 'ne':
130:         '%s,%s' % (LAT + lat_steps, LNG + lng_steps), 'limit': 50, 'intent': 'browse'})
131:
132:         # READ DETAILED INFORMATION FROM EACH VENUE RETURNED FROM THE QUERY (read JSON / Dictionary
133:         format)
134:         for response in responses['venues']:
135:             if response['id'] not in venue_ids:
136:                 categoryList = [category['name'] for category in response['categories']]
137:                 if 'Park' in categoryList:
138:                     day, date, month, year, hour, minute, meridiem =
139:                     datetime.datetime.now().strftime("%A\t%d\t%B\t%Y\t%I\t%M\t%p").split()
140:                     id_4square = response['id']
141:                     venue_name = response['name']
142:                     try: street = " ".join(response['location']['address'].split())
143:                     except: street = "None"
144:                     try: city = response['location']['city']
145:                     except: city = "None"
146:                     state = response['location']['state']
147:                     lat = str(response['location']['lat'])
148:                     lng = str(response['location']['lng'])
149:                     venue_info = client.venues(id_4square)
150:                     checkins, checkins_d = str(response['stats']['checkinsCount']),
151:                     str(response['stats']['checkinsCount'] - venue_info['venue']['stats']['checkinsCount'])
152:                     users, users_diff = str(response['stats']['usersCount']),
153:                     str(response['stats']['usersCount'] - venue_info['venue']['stats']['usersCount'])
154:                     visits = str(venue_info['venue']['stats']['visitsCount'])
155:                     tipCount = str(venue_info['venue']['stats']['tipCount'])
156:                     hereNow, hereNow_df = str(response['hereNow']['count']),
157:                     str(response['hereNow']['count'] - venue_info['venue']['hereNow']['count'])
158:                     dayCreated = str(venue_info['venue']['createdAt'])
159:                     pageUpdats = str(venue_info['venue']['pageUpdates']['count'])
160:                     dislikes = str(venue_info['venue']['dislike'])
161:                     likes = str(venue_info['venue']['likes']['count'])
162:                     url = venue_info['venue']['shortUrl']
163:
164:         # TAGS (keywords associated with a venue) are extracted and saved to a separate
165:         file in the tags folder
166:         try:
167:             tags = venue_info['venue']['tags']
168:             number_tags = str(len(tags))
169:             output_filename_tags = r'tags\tags_%s.txt' % venue_name
170:             output_file_tags = open(output_filename_tags, 'w')
171:             output_file_tags.write(" ".join(tags))
172:             output_file_tags.close()

```

```

163:         tags = 'Yes'
164:     except: tags, number_tags = 'No', '0'
165:
166:         # TIPS (comments associated with a venue) are extracted and saved to a separate
file in the tips folder
167:         try:
168:             tips = venue_info['venue']['tips']
169:             number_tips = str(tips['count'])
170:             if tips['count'] != tips['groups'][0]['count']: print "this venue has more
tips outside of the 'group' category: %s" % id_4square
171:             output_filename_tips = r'tips\tips_%s.txt' % venue_name
172:             output_file_tips = open(output_filename_tips, 'w')
173:             for group in tips['groups']:
174:                 for item in group['items']:
175:                     tip_date = item['createdAt']
176:                     text = item['text']
177:                     tip_likes = item['likes']['count']
178:                     output_file_tips.write('\t'.join(map(str, [tip_date, tip_likes,
text])) + '\n')
179:             output_file_tips.close()
180:             tips = 'Yes'
181:         except: tips, number_tips = 'No', '0'
182:
183:         # SAVE INFORMATION TO TEXTFILE
184:         output_file_venues.write("\t".join( [day, date, month, year, hour, minute,
meridiem,
185:                                             id_4square, venue_name, street, city, state,
186:                                             checkins, checkins_d, users, users_diff, visits,
tipCount,
187:                                             hereNow, hereNow_df, dayCreated, pageUpdates,
dislikes, likes,
188:                                             url, tags, number_tags, tips, number_tips, lat,
lng]) + "\n")
189:
190:         # SAVE THE VENUE ID (to allow checking for duplicates)
191:         venue_ids.append(id_4square)
192:
193:         # visual check that the code is working
194:         print "venue %s added" % len(venue_ids)
195:
196:     # Close written file.
197:     output_file_venues.close()

```

APPENDIX K: MEASURE SPATIAL ACCESSIBILITY (VFCA)

Accessibility_Parks.py

```

1: import arcpy, string, os
2: from arcpy import env
3: arcpy.OverWriteOutput = True
4:
5: ## user specific #####
6:
7: ## 1 ## choice of transport mode and ##
8: mode = "car"
9:
10: ## 2 ## files and fields ##
11: ## 2.a ## workspace
12: arcpy.env.workspace = "C:\Users\cdony\Desktop\AccessParksCLT\Data"
13:
14: ## 2.b ## ORIGIN DATA
15: inFile_Origins = "Meck_BLKGRP_Copy2.shp"
16: oID = "GEOID10"
17: Demand = "Total_Pop"
18:
19: ## 2.c ## DESTINATION DATA
20: inFile_Destinations = "ParksInCLT_edited.shp"
21: dID = "ParkOID"
22: Supply = "ACRES"
23: SupplyAdditional = ["SHELTER", "BASEBALL_S", "BOATING", "SOCCER", "TENNIS", "BASKETBALL",
    "PLAYGROUND", "SWIMMING", "DISCGOLF"]
24:
25: ## 2.d ## OD-MATRIX DATA
26: inFile_OD = "OD_car_transit.dbf"
27: od_oID = oID
28: od_dID = "OIDparks"
29: od_cost = "carTime" if mode == "car" else "transitTim"
30:
31: ## 3 ## ADDITIONAL PARAMETERS
32: distanceDecay = 1
33: SupplyImportance = 0.5
34: importanceSupplyAdditional = [1]*len(SupplyAdditional)
35: attractionZoneCoefficient = 1 if mode == "car" else 1.2
36: accessFieldName = "%sA_%s" % (str(mode)[:3], str(SupplyImportance)[2:])
37:
38: #####
39:
40: ## 1 ## Loading data into multi-dimensional arrays #####
41: #####
42:
43: ## def(1) ## function that will load the requested ##
44: ## fields based in the file name, a prefix
45: ## and the field names
46: def loadData(inFile, prefix, fields):
47:     for field in fields:
48:         globals()[prefix + field] = []
49:     for record in arcpy.SearchCursor(inFile):
50:         for field in fields:
51:             globals()[prefix + field].append(record.getValue(field))
52:         globals()["number" + prefix] = len(globals()[prefix + fields[0]])
53:
54: ## 1.a ## Loading the origin point ID's and Population ##
55: loadData(inFile_Origins, "Origin", [oID, Demand])
56:
57: ## 1.b ## Loading the destination point ID's,
58: ## Service information and amenities
59: loadData(inFile_Destinations, "Destination", [dID, Supply])
60: loadData(inFile_Destinations, "Additional", SupplyAdditional)

```

```

61: maxSupply = max(globals()["Destination" + Supply])
62: sumAdditionalSupply = [sum([importanceSupplyAdditional[AdditionalSupply] * globals()["Additional" +
SupplyAdditional[AdditionalSupply]][i] for AdditionalSupply in range(len(SupplyAdditional))]) for i
in range(numberDestination)]
63: maxAdditionalSupply = max(sumAdditionalSupply)
64:
65: ## 1.c ## Loading the OD-matrix information needed
66: ## (i.e. field of origins, field of destinations
67: and field of cost computation)
68: loadData(inFile_OD, "OD", [od_oID, od_dID, od_cost])
69: maxCost = max(globals()["OD" + od_cost])
70:
71: ## 2 ## Computing accessibility #####
72: #####
73:
74: ## 2.a ## Computing the attraction zone of each park
75: print "calculating attraction of supply"
76: attraction = []
77: for supplyIndex, supply in enumerate(globals()["Destination" + Supply]):
78:     supply = supply if supply < maxSupply*0.4 else maxSupply*0.4
79:     attraction.append(SupplyImportance*(supply/maxSupply) + (1-
SupplyImportance)*(sumAdditionalSupply[supplyIndex]/maxAdditionalSupply))
80:
81: normalizedAttraction = [attractionValue/max(attraction) for attractionValue in attraction]
82: attractionZone = []
83: for supplyIndex in range(numberDestination):
84:     if normalizedAttraction[supplyIndex] <= 0.2:
85:         attractionZone.append(5 * attractionZoneCoefficient)
86:     elif normalizedAttraction[supplyIndex] > 0.2 and normalizedAttraction[supplyIndex] <= 0.4:
87:         attractionZone.append(15 * attractionZoneCoefficient)
88:     elif normalizedAttraction[supplyIndex] > 0.4 and normalizedAttraction[supplyIndex] <= 0.6:
89:         attractionZone.append(30 * attractionZoneCoefficient)
90:     elif normalizedAttraction[supplyIndex] > 0.6 and normalizedAttraction[supplyIndex] <= 0.8:
91:         attractionZone.append(45 * attractionZoneCoefficient)
92:     else: attractionZone.append(maxCost * attractionZoneCoefficient)
93:
94: ## 2.b ## Computing the total demand weighed by distance
95: print "calculating Supply side"
96: sumVj = [1]*numberDestination
97: for OD in range(numberOD):
98:     try:
99:         destinationID = globals()["OD" + od_dID][OD]
100:         destinationIndex = globals()["Destination" + dID].index(destinationID)
101:         ODtime = globals()["OD" + od_cost][OD]
102:         if ODtime <= attractionZone[destinationIndex]:
103:             originID = globals()["OD" + od_oID][OD]
104:             demandIndex = globals()["Origin" + oID].index(originID)
105:             demandPopulation = globals()["Origin" + Demand][demandIndex]
106:             Vj = demandPopulation / pow(ODtime, distanceDecay)
107:             sumVj[destinationIndex] = sumVj[destinationIndex] + Vj
108:     except ValueError: pass
109:
110: ## 2.c ## Computing the Parks-to-Population ratio
111: Rj = [globals()["Destination" + Supply][destinationIndex]/sumVj[destinationIndex] for
destinationIndex in range(numberDestination)]
112:
113: ## 2.d ## Computing the accessibility to a service
114: ## for each origin point
115: print "calculating Accessibility"
116: sumAi = [0]*numberOrigin
117: for OD in range(numberOD):

```

```

118:     try:
119:         originID = globals()["OD" + od_oID][OD]
120:         demandIndex = globals()["Origin" + oID].index(originID)
121:         destinationID = globals()["OD" + od_dID][OD]
122:         destinationIndex = globals()["Destination" + dID].index(destinationID)
123:         ODtime = globals()["OD" + od_cost][OD]
124:         if ODtime <= attractionZone[destinationIndex]:
125:             sumAi[demandIndex] = sumAi[demandIndex] + Rj[destinationIndex]
126:     except ValueError: pass
127:
128:     ## 2.d ## Normalization of the accessibility measure
129:     ##         to a value between 0 and 1
130:     print "Normalizing Accessibility measure"
131:     Ai = [sumAi[demandIndex] / max(sumAi) for demandIndex in range(numberOrigin)]
132:
133:     ## 3 ## Assign accessibility measure to each origin #####
134:     #####
135:
136:     ## 3.a ## Create a new Field calles Accessibility #####
137:
138:     print "--- %s %s ---" % ("Creating Accessibility field in", inFile_Origins)
139:     arcpy.AddField_management(inFile_Origins, accessFieldName, "FLOAT")
140:
141:     ## 3.b ## assign computed accessibility values to each
142:     ##         origin #####
143:
144:     print "--- %s ---" % ("Adding accessibility measures into ""accessibility"" field")
145:     cursor = arcpy.UpdateCursor(inFile_Origins)
146:     cursor.reset()
147:     for row in cursor:
148:         originID = row.getValue(oID)
149:         demandIndex = globals()["Origin" + oID].index(originID)
150:         row.setValue(accessFieldName, sumAi[demandIndex])
151:         cursor.updateRow(row)
152:
153:     ## end #####
154:
155:     del cursor
156:     print "--- %s ---" % ("done")
157:

```