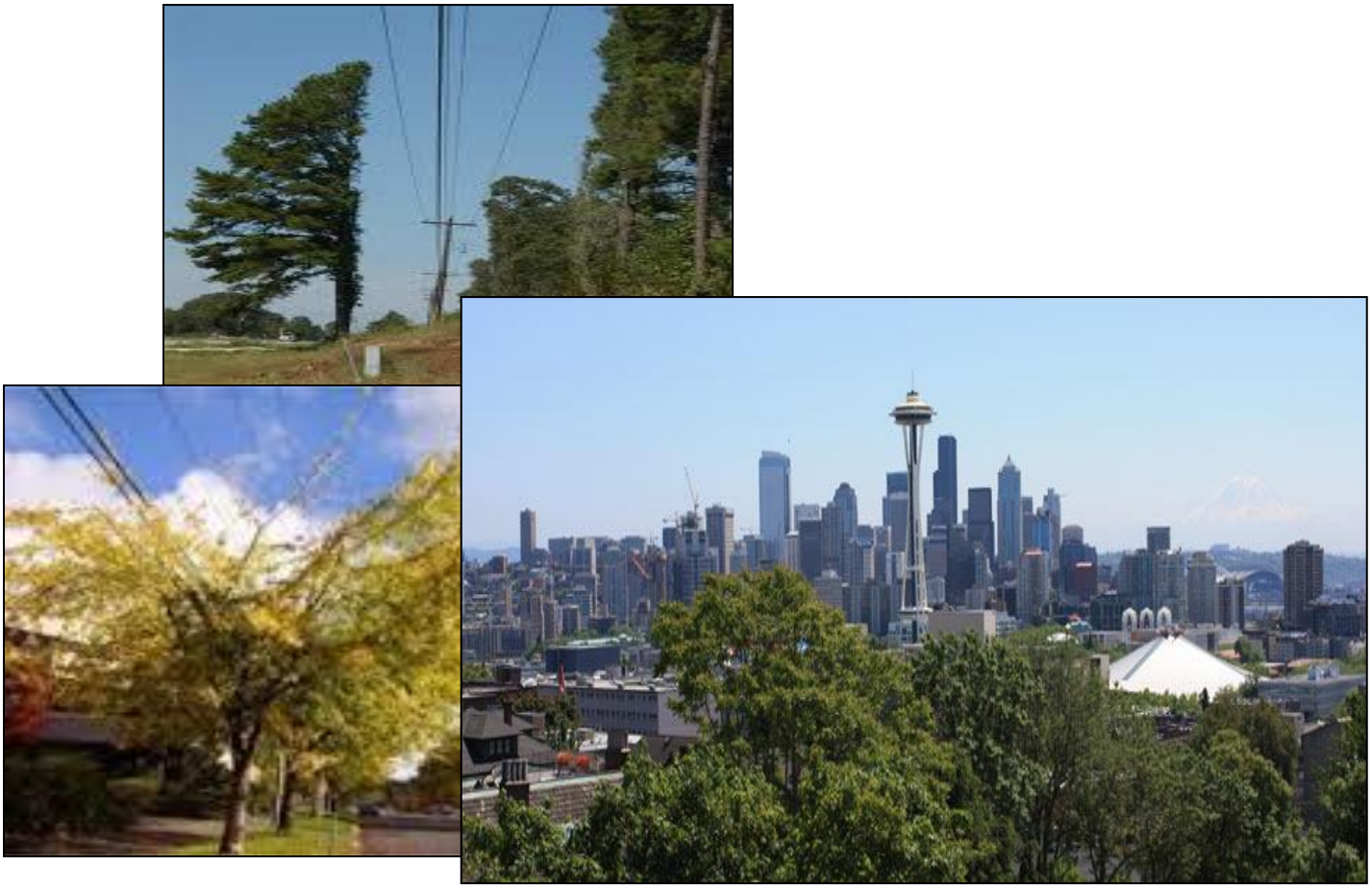

Trimming Trees and its Impacts on the Urban Forest

Geography 569 GIS Workshop
Group 3 – Capstone Project
August 22, 2014

Project Sponsor

Seattle City Light
David Bayard (Arborist)

Wally Koli

Sahar Pesaran Afsharian

Ben Hillam

Table of Contents

List of Figures

List of Tables

Recommended Course of Action

Through the course of defining a new method to estimate the amount of canopy that might possibly be remove from routine tree trimming by Seattle City Light we learned a lot of valuable lessons and have gained insight on how our methods can be used in the future. The methods and tool provided from this project can be implemented with more current data to get a more accurate estimation of total canopy cover. Also, by evaluating the inequity that exists throughout the greater Seattle area in relation to susceptibility to canopy loss we have established a base in which areas of concern can be identified. We recommend that Seattle City Light use these design, methods, and results to build from. First by obtaining more accurate data by recording trimming, using more robust tree inventory data to create samples to extrapolate from, and a GIS to manage all the data in relation to all this analysis.

Introduction

The world's population continues to migrate to urban locations and as a result an increased emphasis has been placed on maintaining, creating, and protecting urban forests. The urban forest and trees in general have been shown to benefit not only the biophysical but the economic and social systems they are a part of. The construction of city infrastructure to sustain its populations has left a large impact on the urban forests. As knowledge about the positive impacts of the urban forest and its ability to assist in sustaining healthy cities grows, more pressure is placed on city officials to maintain the urban forest and provide the necessary services to its residents. In regards to the services that the urban forest and tree canopy provide forester Dan Northrop (2013) stated, "It is becoming increasingly clear with each passing day that these services are critical to the health and well-being of our expanding urban population. In light of these findings, the conservation and restoration of these urban and interface forests must now be seen as a fundamental goal of any viable public works program or land-use planning process." (pp. 1) At the forefront of cities with this fundamental initiative is Seattle, Washington.

Seattle City Light (SCL) is a public utility that provides the electrical power to Seattle and a few other cities north/south of its boundaries. It is the 10th largest public utility in the country. SCL is committed to not only providing power in a safe and efficient manner, but they strive to also manage the vegetation that their utility network impacts (Bayard, 2014). Seattle is a city that prides themselves on their iconic green status and embraces its status as the "Emerald City" (Mapes and Mayor, 2014). SCL is aware of the amount of trees that are removed and planted across its service area every year. Although the amount of trees that are planted and removed each year is accounted for SCL does much more that impacts the tree canopy throughout its service area. SCL trims over 100,000 trees on a 4-year cycle. This activity accounts for a lot of the work that the department does on a regular basis but is non-existent in any analysis of the impacts that the department has on the city, people, environment, and overall aesthetics of the urban forest found within its service area. David Bayard, arborist for SCL, is our sponsor for this project. He has expressed the concern that SCL's vegetation management has in regards to the impacts of the entire tree trimming to allow for clearance of their high-voltage power lines. It is important to look at the picture as a whole to understand all the possible negative and positive influences that SCL has on the community it serves. Figure 1 gives an overview of all the socio-ecological systems that are involved in relation to SCL's impact on the urban forest. More generally it discusses the impacts of tree canopy for SCL and its surrounding systems. Understanding the impact that tree trimming for utilities helps fill any void in SCL's view of their entire system. To better understand the impacts that trees have on the systems around it, it is critical to not focus on one aspect of it. Professor Coder in his article about community tree canopy loss stated, "If you cannot see the whole picture (and its integral parts) in a community, then you understand community changes it only through a biased and limited sampling of areas . . . The more compartmentalized you are within your community, the less effective you can be in understanding natural resource changes like trees impacting whole community life." (2011, pp 1). SCL is trying to look at the total impact they have

Social-Ecological Systems Table | Goal: Assessment of the Impact of Canopy loss on the City of Seattle.				
Geographically		Biophysical	Economic	Social
Above Focal Scale (Extra-system effects)	King County	Impacts on carbon sequestration capability, flood control, ability to reduce pollution, impacts on green house absorption- climate change (Jones et al. 2013; Rowntree 2008)	Trees reduce heating and cooling costs, improve air quality and cleansing, help increase consumer patronage, boosts occupancy rates of offices. (Center for Urban Horticulture UW College of Forest Resources)	Urban forests increase property values, provide cultural services to communities. (Jones et. Al 2013, Rowntree 2008)
Focal Scale(System)	Seattle City Light Service Area	Well placed trees can convert streets, parking and walls into more aesthetically pleasing environments. (Dan Burden May, 2006)	The urban forest plays a role in saving more than 12.2 million dollars annually by removing pollution, from carbon storage, and reducing energy costs in residential building (Ciecko, Tenneson, Dilley, and Wolf 2012)	Human Values: less canopy impacts on outdoor activity i.e. to walk outdoors in parks becomes less desirable, outdoors becomes less inviting i.e because of crime
Finer Scale (Agents)	Canopy Loss At Census Block Level	Trees reduce wind speeds and reduce noise levels adding to the livability of urban environments.	Emotional Impacts, costs to community health, loss of canopy cover, may increase blood pressure, breathing and related health problems (Dunlap and Jones 2002)	Trees Provide social connectivity, in addition to contributing to aesthetics, public health and well-being, property values, and community stability (Tyrva¨inen et al. 2003; Wolf 2004)

Figure 1 Social Ecological Systems Table

on the community not simply one aspect of it. The difficulty is that the ability to analyze the amount of trimming as an aggregate across the entire service area is difficult.

For this project we will be looking at methods to estimate the amount of tree canopy that is impacted by SCL's trimming of trees. Our objectives include designing a model that will serve as a starting point for analyzing canopy loss due to trimming. Along with estimating the amount of canopy loss we will be looking at social inequality in relation to these activities. For SCL to fully understand the impacts that its actions have on the community it will be important to incorporate what has been

previously unaccounted for, trimming a lot of trees. Our design, methods, and analysis were all focused around helping SCL include canopy loss due to trimming into their method for evaluating the net impact they have on the different systems in their service area.

Design & Methods

There have been various studies conducted the tees, tree canopy, and the effects they have on the natural, social, and economic environment surrounding them. There are many foresters, arborists, scientists, etc. that have identified methods to assign a value to a tree. There have also been multiple studies that look at the benefits of trees as far as home values, health, ecosystem services, aesthetic quality, etc. However when it comes to evaluating the value of a tree that has been trimmed, this has been done on an individual tree basis (Hoyer, 2013). For our project and to assist in fulfilling our sponsor's requests we have designed a project to look at trimming trees at a much larger scale. We are also interested in looking at the social equity involved in the areas located within SCL's service area. We have designed two approaches.

Canopy Estimation

The first major obstacle in designing a method for evaluating the quantity of tree canopy impacted by SCL trimming for their feeders was obtaining data to represent the tree canopy. The percent canopy cover from the USGS 2011 layer was used in our first approach at evaluating the amount of canopy that SCL feeders affect. Figure 2 shows the model used to calculate the amount of canopy that could reside within a critical distance of each feeder. The percent canopy layer is a Digital Elevation Model (DEM) that contains a value for the percentage of canopy in each cell. This value must be converted to an integer to convert it to a vector data polygon layer. Once the percent canopy layer is converted to a polygon the area can be calculate for each grid. From the calculated area now a total area of canopy per grid can be derived by using the value, in our case the field was named "gridcode"

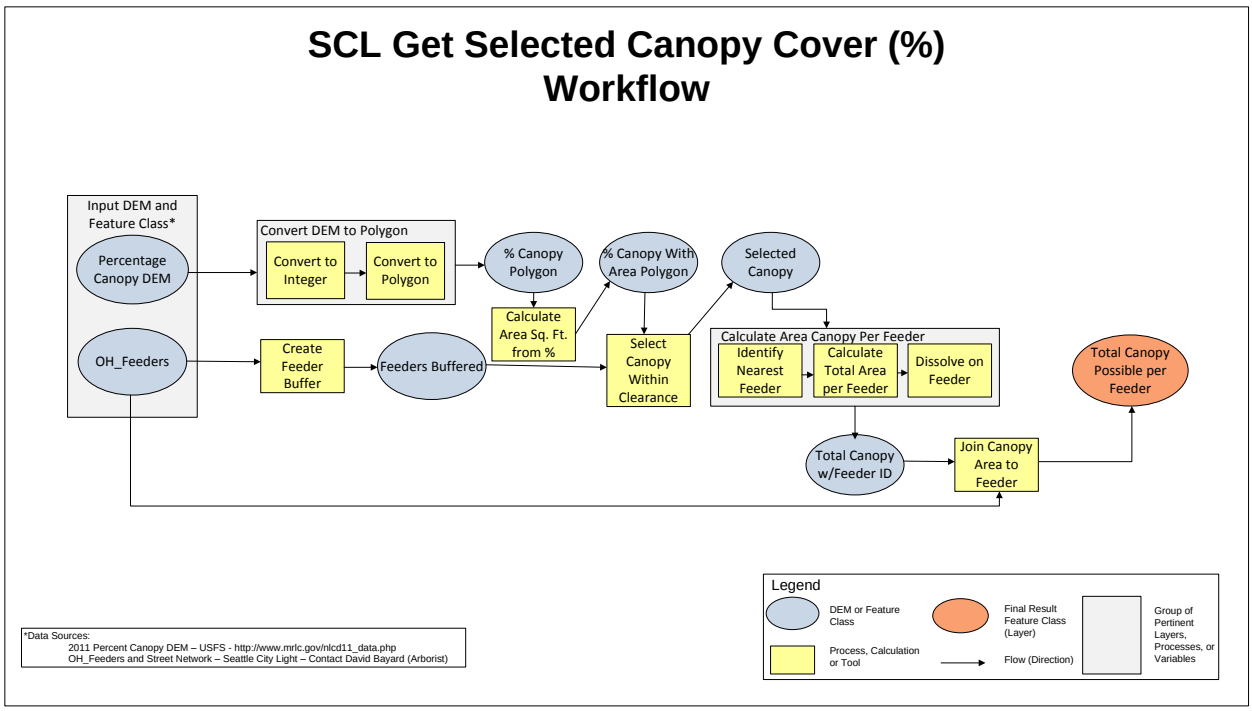

Figure 2 Percent Canopy Process

containing the value, multiplied by the total area. With the feeders layer, "OH_Feeders" Figure 2, a buffer is created for the input clearance distance by the user. For our study we looked only at a 10 Ft. clearance distance from the high-voltage lines. Once the amount of canopy is calculated and the clearance buffer is created the canopy grids which intersect the clearance buffer are selected for further analysis. With the selected canopy we can now identify the feeder line that it is nearest too, calculate the total canopy per feeder, and obtain a total canopy grid layer with the total area of canopy per feeder. We joined this result table to the "OH_Feeders" to get the result layer that is only has the area per feeder from qualified areas that intersected the feeder (Figure 2). The actual feature classes are named differently in the file geodatabase, but in Table 1 in Appendix A it outlines the file geodatabase with its associated name and description.

Our second and more detailed approach at analyzing the amount of canopy that could possibly have to be trimmed by SCL workers involved creating two Python script tools. The purpose for creating two script tools is that we can provide an easy user interface for users to simply add their own layers to

run the analysis on. To get a more accurate representation of trees that would fall within the clearance distance of the feeder we obtained LIDAR data from the Puget Sound LIDAR Consortium (PSLC). The data consists of a 6 ft. resolution base and top surface height DEM for the Puget Sound region from 2000 (PSLC, 2014). Figure 3 below shows the python script design for accomplishing the first task in using the PSLC data, getting the selected canopy cells.

As seen in Figure 3 there is a lot more involved in evaluating the 2000 height data. To begin with the base DEM is subtracted from the top surface DEM using the minus tool in ArcToolbox. This creates a result DEM that now has the height of each cell from the ground. After the height DEM is created the DEM must be converted to an integer value so that it can be converted into a vector polygon layer. We multiplied the value (height) by 10000 using the "Times" tool to preserve the decimal places up to 6 locations. Once all of this is converted to a polygon then the value is converted back to a double with its accurate height in feet above the ground. On the bottom portion of Figure 3 the second step in this analysis is using the feeders layer "OH_Feeders" and the street work layer "Street_Network" to calculate which type of road the feeder would run along. We need to know the type of road the feeder coincides with because if it is an arterial road than the height used to evaluate the high-voltage power is 45.5 ft. If the feeder is along a non-arterial road than the height used is 40.5 ft. Figure 1 in Appendix A shows the pole and line heights used to derive these heights for each pole. We split the feeder layer into individual segments and identified the nearest street with the near tool in ArcToolbox. Primary arterial roads were identified with a "1" in their "FEACODE" attribute column. All feeders nearest to a street with "FEACODE = 1" were classified as an arterial feeder while the remainder were classified as non-arterial. We used our height polygon to select grids that were within 10 ft. of each type of feeder. A feeder is always trimmed 15 feet above and then whatever the desired distance input by the user laterally and below. Using the height qualifiers we further refine our selection of valid height

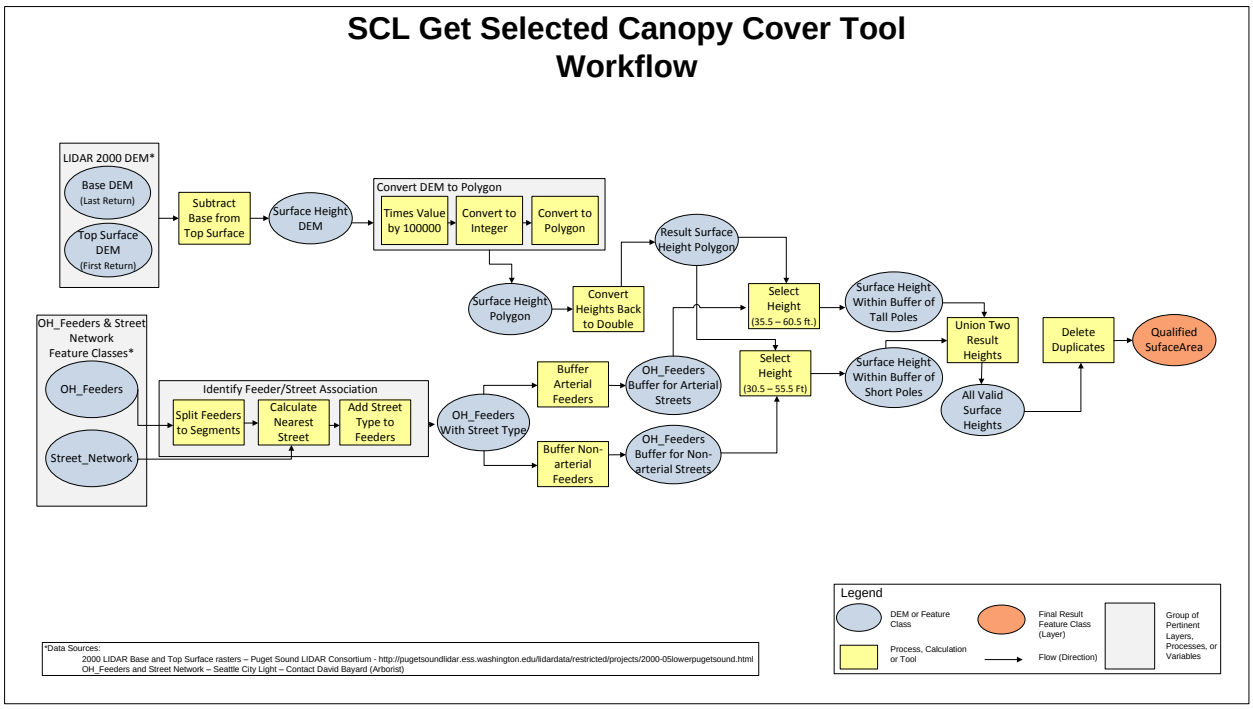

Figure 3 Get Selected Canopy Cover Tool Diagram

grid cells. The results are joined together, duplicates are removed, and we are left with our final qualifying height grid cells. Once all the qualifying areas are determined identified the next python

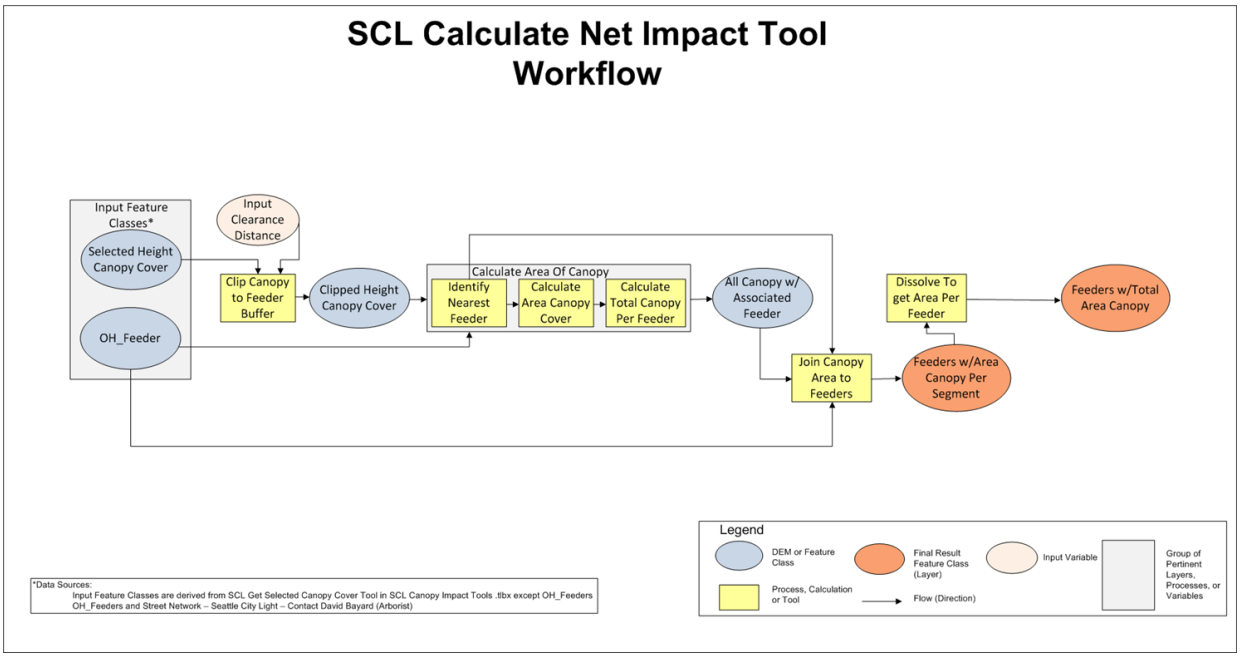

Figure 4 Calculate Net Impact Diagram

script tool calculates the area of each and all the feeders that might have canopy that would be trimmed. Figure 4 shows the second Python script tool workflow. It simply takes the qualified height grid, clips it to the input clearance distance the user inputs and calculates the amount of canopy for each feeder. The results are joined back to the feeder layer and dissolved to come up with the final resulting feeder layer that has the total area of canopy that would fall within the clearance distance and therefore need to be removed.

To assist user in implementing these two Python scripts a tool box was used to store the tools. There are two Python script tools that have Graphical User Interfaces (GUI) for users to input data. Figure 4 shows the interface and inputs to run the first tool. The user has to put in a base height raster DEM, a top surface raster DEM, a feeder feature class, a streets feature class, the desired clearance they want to use to do their evaluation, and output locations for the resulting layer. Figure 5 is the GUI for the SCL Calculate Canopy Net Impact Python tool. It takes the result qualified canopy height feature class from the SCL Get Selected Canopy Cover Tool, the feeder feature class, a desired clearance input (needs to be the same for both tools to get consistent results), and a folder or geodatabase location to save the results into. It is important to note that all the inputs should have the same spatial reference, the DEMs should have a value of “gridcode” and the streets need to have a “FEACODE” in the attribute table to help produce consistent and more accurate results.

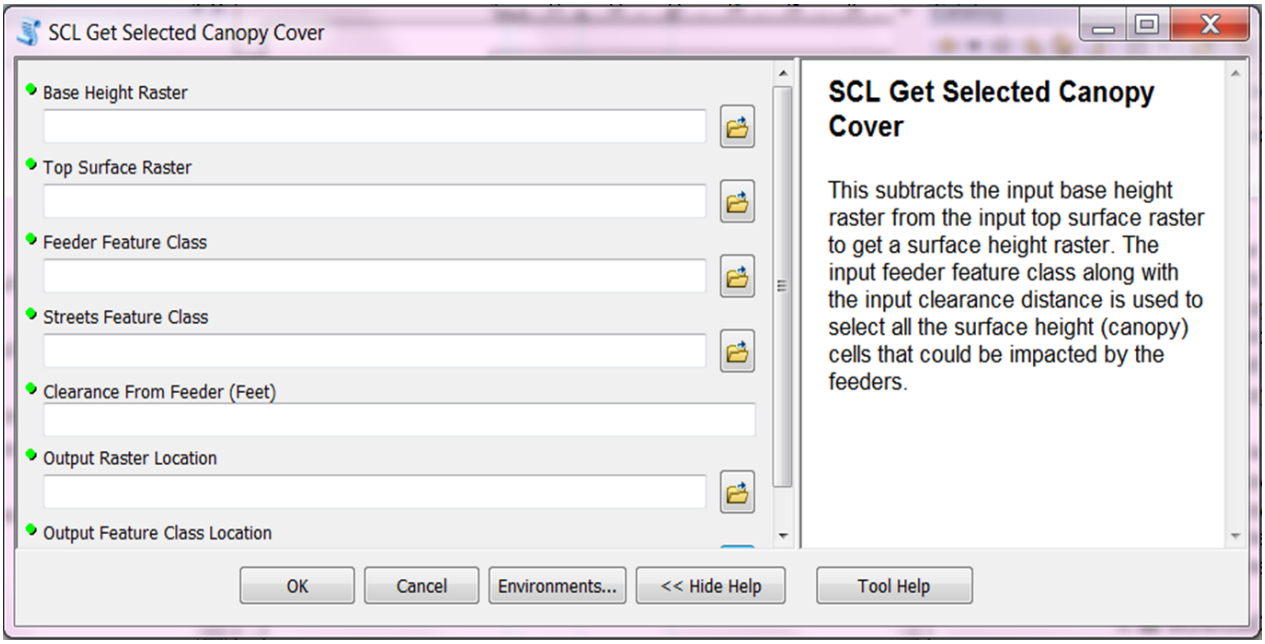

Figure 5 SCL Get Selected Canopy Cover GUI

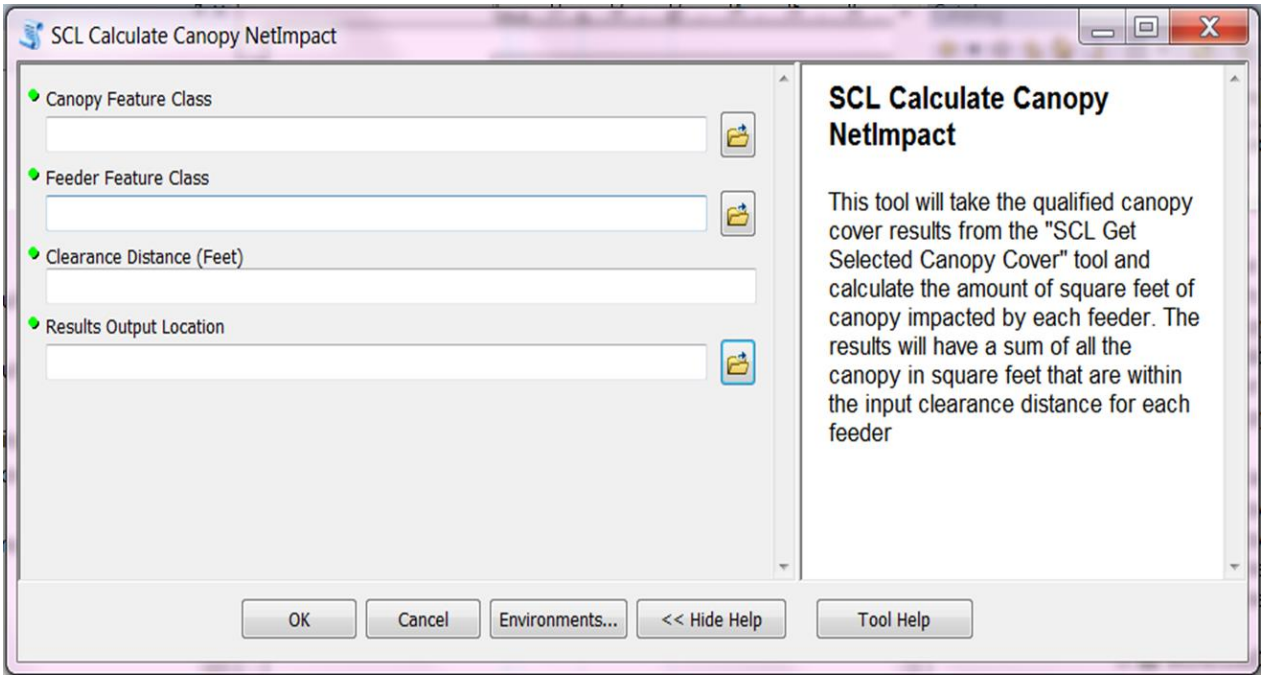

Figure 6 SCL Calculate Canopy Net Impact GUI

Urban forests provide important ecological benefits by modifying urban climate, improving air quality, storing and sequestering carbon, saving energy, decreasing storm water runoff, and increasing biodiversity and providing wildlife habitat. Urban forests also provide social benefits to urban residents such as improving urban aesthetics and reduction of stress, crime and traffic speed. Uneven distribution of urban forest across a city and its residents can create inequality (Akbari et al.2001; Nowak et al. 2006).

In this research project, we studied the level of inequity and social justice to determine socioeconomic status by tract and observed this distribution across the study area and determined vulnerable census tracts regarding to inequity index. Purposes of the index is to assist SCL identifying areas of concern to make more informed and effective decisions in regard to future trimming regimes, tree removing, tree planting and utility maintenance. For this analysis, we employed multi-criteria evaluation analysis (MCE) and developed an inequity index with nine most important criteria. These criteria were driven and selected from multiple potential alignments in the threshold matrix and documents provided by our sponsor. The threshold matrix includes potential variables representing biophysical, economic and social aspects at three different scales (King county, Seattle City Light service area and a census tract/neighborhood). These thresholds can be seen in Figure 7. The study relies on the 2010 census data at tract level in Seattle City Light service area. By intersecting the King county census tract with buffered feeders (100 ft. buffer for overhead feeders), 168 census tracts were exported to evaluate their inequity levels.

The selected input variables are combination of four major criteria, namely race, income, population and percent canopy cover and five minor criteria, namely numbers of tree removed, numbers of tree planted, percent of green space, proximity to industrial centers, and proximity to hospitals. Table 1 shows the list of selected criteria.

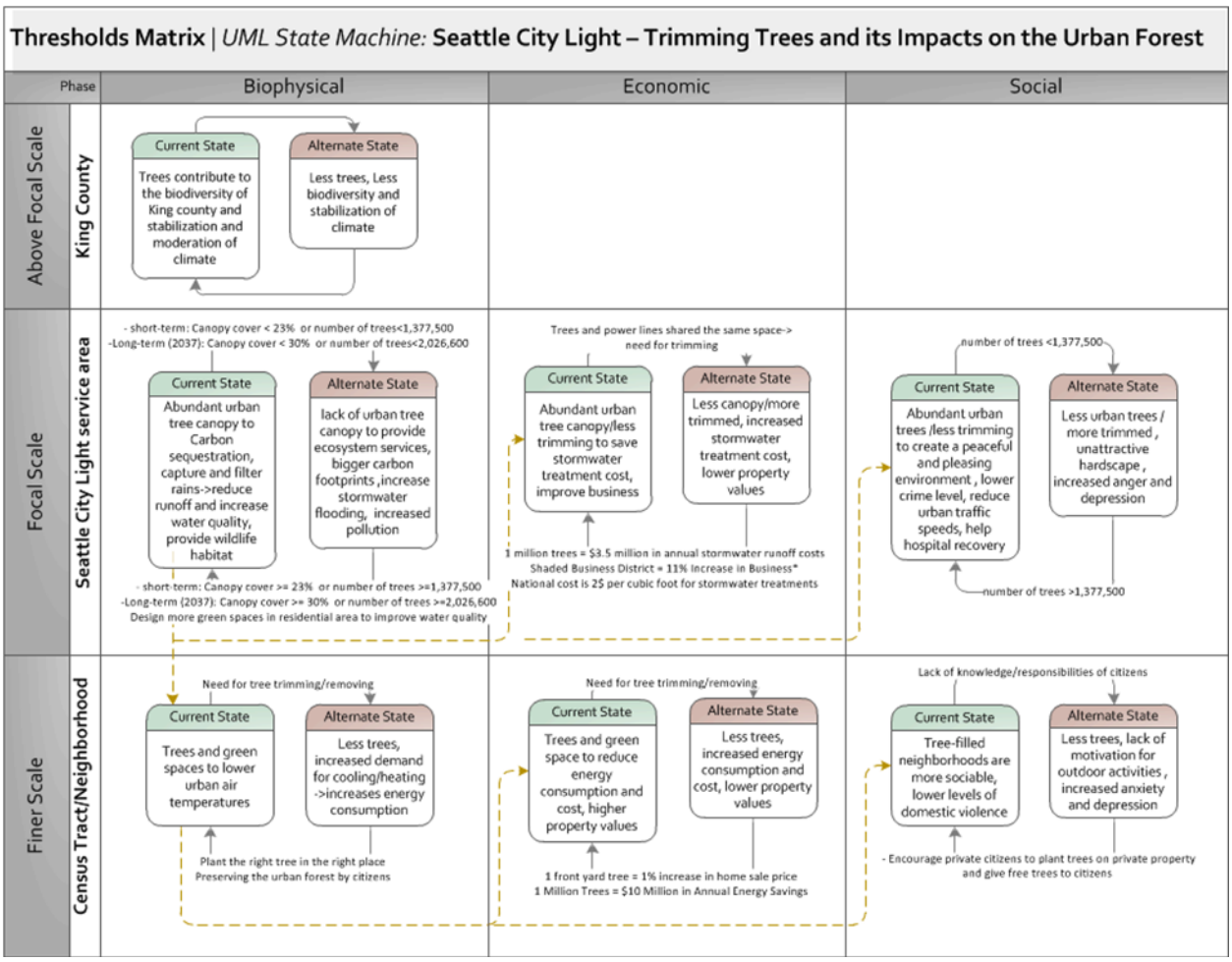

Figure 7 Threshold Matrix

Aspect	Variable Chosen
Socioeconomic	Race
	Income
	Population
	Proximity to Industrial Centers
	Proximity to Hospitals
Biophysical	% Canopy Cover
	% Green Space
	Number of Trees Removed
	Number of Trees Planted

Table 1 Nine Selected Criteria

Race: Trees improve the quality of life for all urban residents. The benefits of urban tree canopy cover are distributed equally among white and non-white residents. Race is a common indicator of inequality in urban environmental studies. For race we used percentage of minority/non-white population. Greater values indicate higher priority.

Income: Trees improve the quality of life for all urban residents. The benefits of urban tree canopy cover are distributed equally among residents with all income levels. Studies have shown that median household income and wealth are directly related to neighborhood tree cover. Income defines the possibility of spending money towards planting tree. The spent money on planting can be either in form of direct investment or taxes paid to the city. Unlike the other variables used, lower values of median household income indicate higher priority.

Population: Planting more trees/trimming less will improve environmental quality and provide public benefits and it shows higher impact in highly populated areas. Population includes all people, male and female, child and adult, living in a tract as defined by the U.S. Census Bureau. Greater values indicate higher priority.

Percent canopy: In order to determine if uneven distribution of canopy cover exists across tracts, we used the percent canopy cover as a function of socioeconomic variables to be able to compare canopy between tracts and all other criteria. Unlike the other variables used, lower percent canopy cover indicates higher priority.

Number of trees removed: The sum of number of trees removed by tract is used to measure the need for planting. Areas with higher values are prioritized for more tree canopy/planting. Tracts without tree removed are left zero. Greater values indicate higher priority.

Number of trees planted: The sum of number of trees planted by tract is used to measure the need for planting. Areas with lower values are prioritized for more tree canopy/planting. Tracts without tree

planted data were imputed by average number of trees planted in the city of Seattle area. Unlike the other variables used, lower number of trees planted indicates higher priority.

Green spaces: The rational for tree canopy/planting in and around existing green spaces is similar to the reasoning behind the number of tree planted variable. Planting trees in the area with less green spaces improve the inequity. Parks and wetlands data were merged together and normalized by tract area. Lower values indicate higher priority for tree canopy/tree planting.

Proximity industrial: Trees improve air quality directly and indirectly by reducing ambient air temperatures, removing air pollutants and by reducing the energy demand from cooling buildings (Akbari et al.2001; Nowak et al. 2006). Distance of industrial centers is used as an indicator measurement of air pollution. Lower values indicate higher priority for tree canopy/planting.

Proximity to hospital: Several studies have shown a positive relationship between patients' recovery in hospitals and proximity to trees and green space in urban areas. Lower values indicate higher priority for tree canopy/planting.

Data for this analysis come from a variety of sources, such as the King County GIS center, the City of Seattle (WAGDA) and Puget Sound Regional Council. Pre-processing includes clipping, merging, spatial join, intersecting was done to develop data for analysis. A workflow diagram showing pre-processing steps is shown in Figure 8.

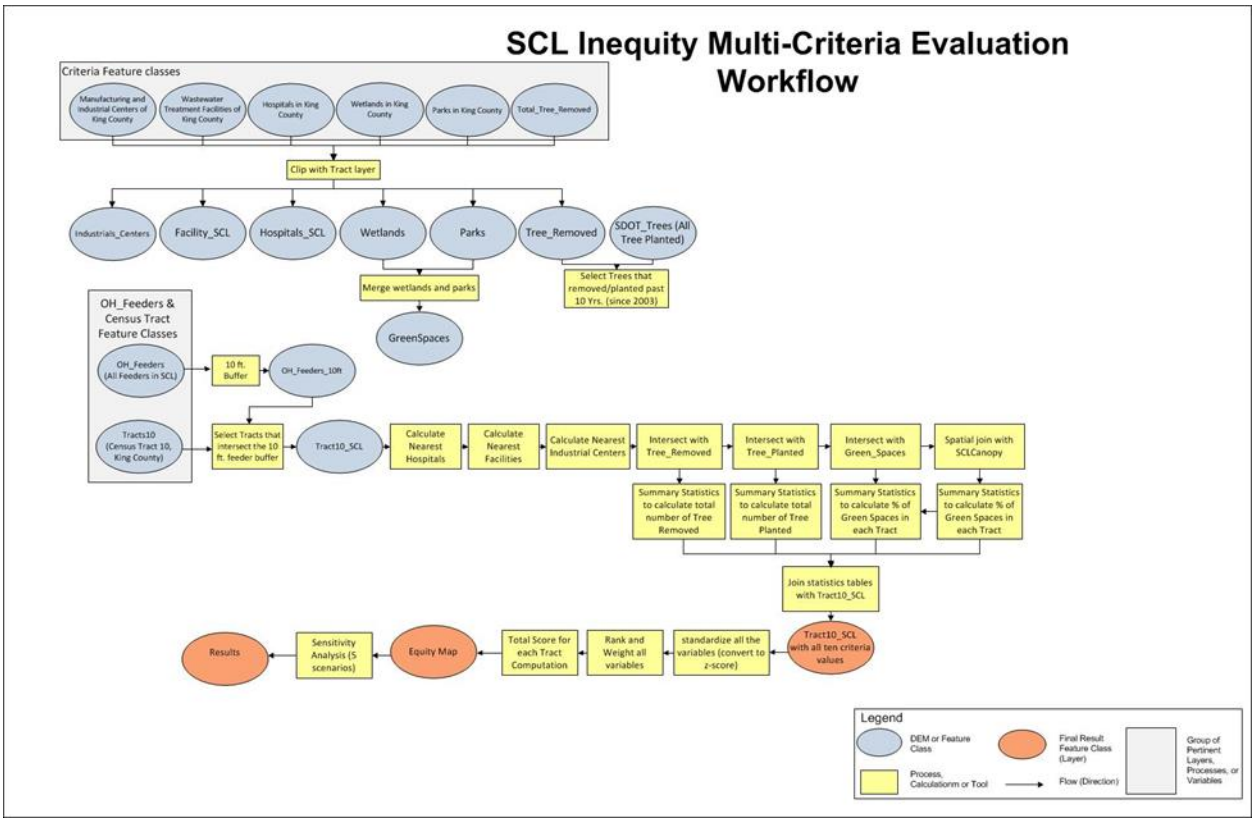

Figure 8 MCE Workflow

Statistical Analysis Methods

To run multi-criteria analysis with so many variables of different units, we first checked for distribution normality. The distribution of total population, income and percent canopy cover across tracts are normal. However, most of the variable distributions were positively skewed. To make analysis more reasonable and be able to compare and combine different indicator categories easier, we used z-scores to standardize all variables which are raw scores converted to standard deviation units. The standardized variables can then rank and weight relative to their importance. To convert data to standard units for each criterion, z-score were calculated by the following formula:

X= a raw value to be standardized

μ= mean

σ= standard deviation

$$z = \frac{x - \mu}{\sigma}$$

Some z-scores are multiplied by (-1) when a higher value means a lower priority, namely income, percentage of canopy cover, percentage of green spaces, number of trees planted, proximity to industrial centers and hospitals. In order to address inequity concerns, a tract with low income represents higher priority for having trees and tree canopy while a tract with a high income would reflect a lower priority tract. Similarly, an area with high green space percentage would reflect a lower priority tract.

We calculated the percentage of minority/non-white population by subtracting the percentage of white population from 100%. We selected the tree removed/planted since 2003 and the planted trees in fair, good and excellent condition. Wetlands and parks are identified and combined as green spaces. From two different sources manufacturing and industrial centers data were downloaded and combined to be a single shapefile as the industrial centers. We created columns for each variable in ArcMap that count each variable at the tract level.

We used intersect tool as a GIS tool to identify the number/amount of features of interest in each tract and we calculated the sum of number/area of a feature of interest in each tract by using summary statistic tool. Next, we joined the output tables of the summary statistic with the tract10 shapefile by tarct-ID. There were four criteria used in the intersect analysis: percent of canopy, number of tree removed, number of tree planted, and percent of green spaces.

Near Analysis as a GIS analysis method is used to calculate the distance of each tract from a feature of interest. There were two criteria used in the near analysis: proximity to industrial centers and proximity to hospitals. Figure 9-11 show the distribution of each criteria based on z-score.

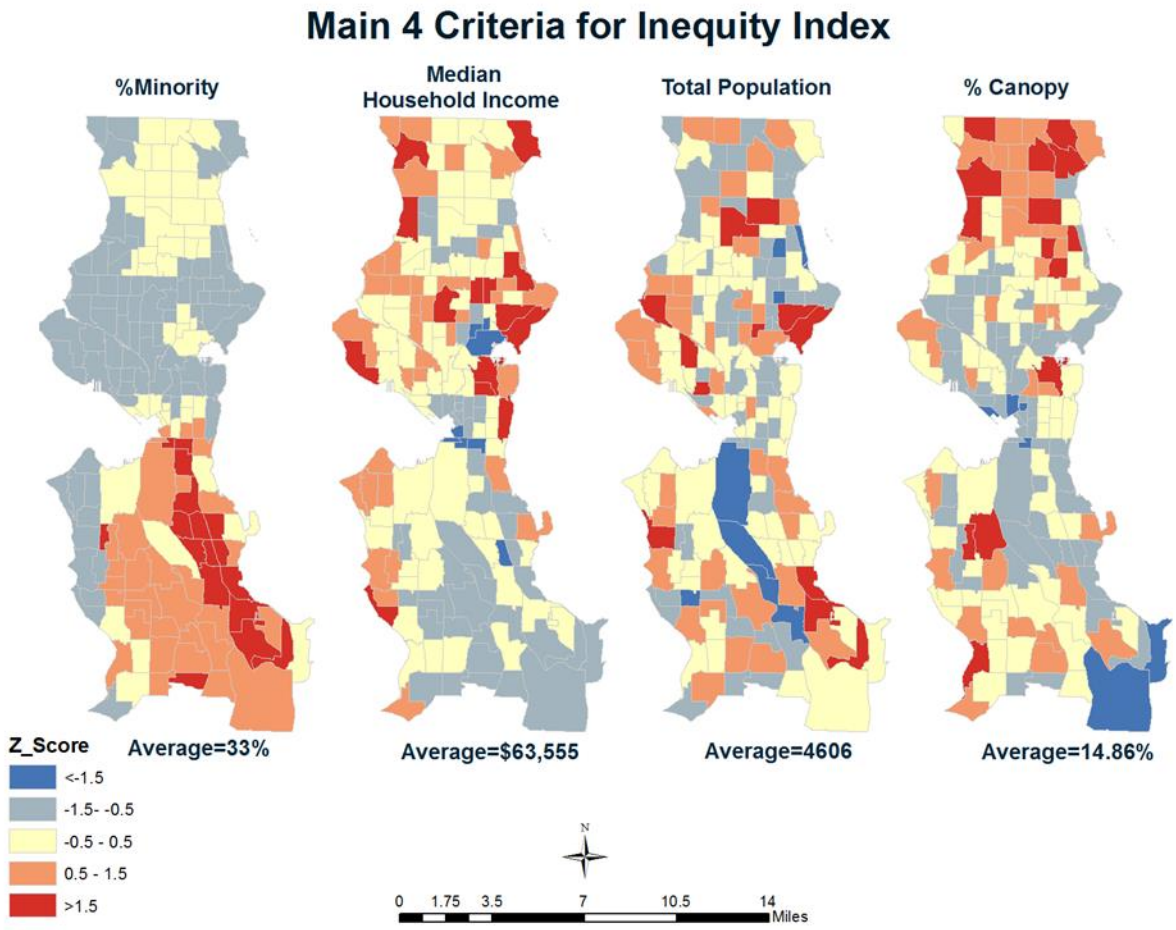

Figure 9 Main 4 Criteria z-Scores

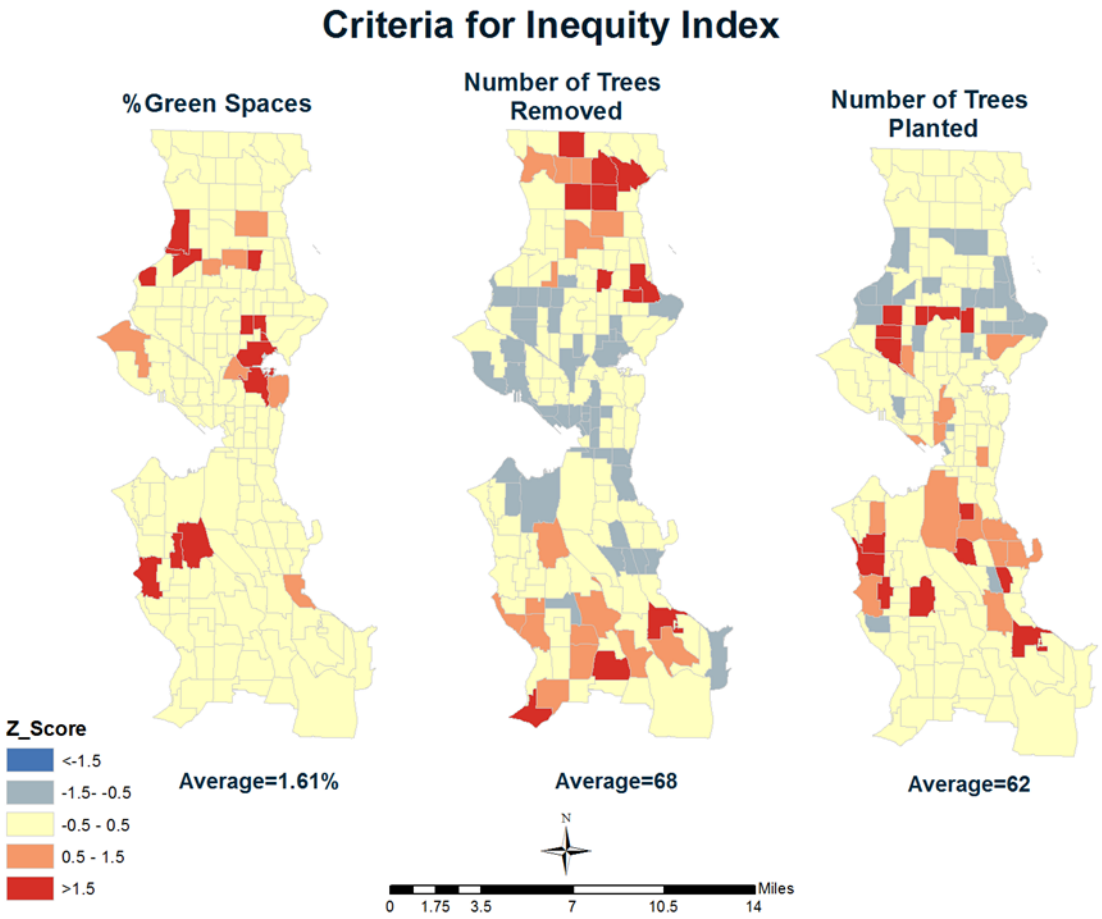

Figure 10 Additional Criteria

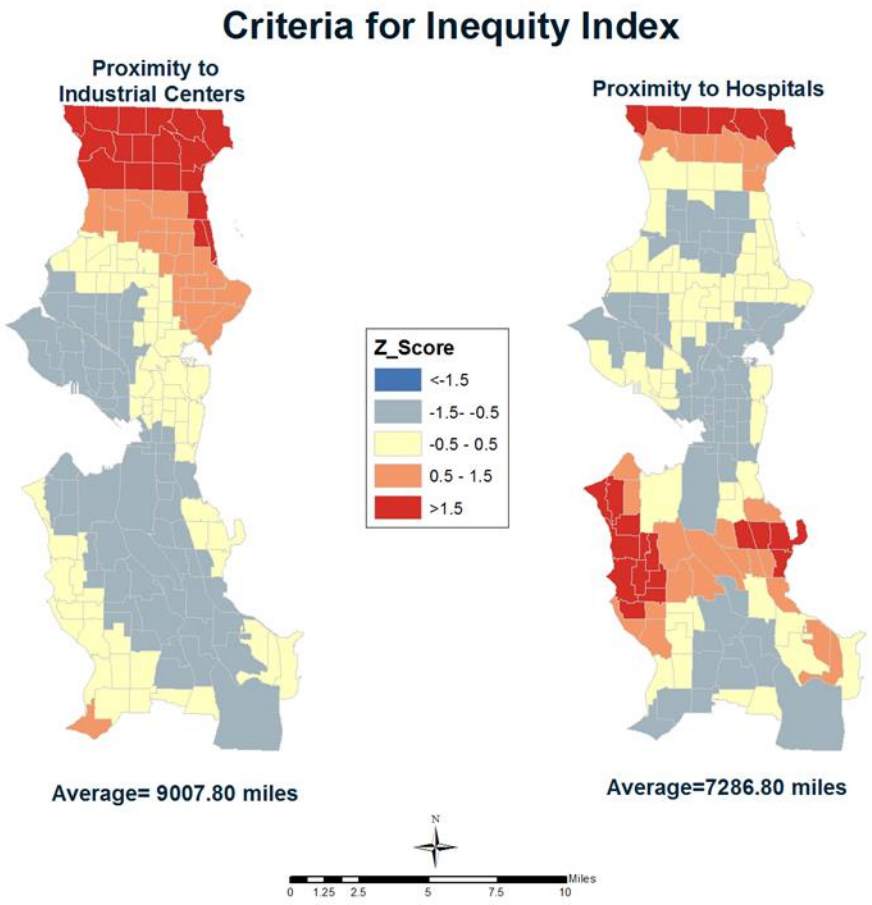

Figure 11 Final Criteria for Index

Rank:

Once all of the variables have been converted to standard units, they were assigned a rank risk of inequity from 1-5 based on z-score ranges. A value of 1 represents the lowest risk and 5 represent the highest risk.

The highest risk (rank 5) tracts are tracts with highest minority percentage, lowest income, highest total population, lowest canopy cover percentage, less green space, highest number of tree removed, lowest number of tree planted, closest to industrial and hospitals. This decision was based on the concept of inequity and social justice.

Weight:

We ordered criteria based on relative importance of criteria. We then manually defined weights for the criteria out of 100 to only show the important criteria from SCL perspective. We applied four different methods including sum ranking, reciprocal ranking, rating and pairwise comparison methods to achieve the accurate weights. Table2 shows the all results of different weighting methods.

		Rank Sum		Rank Reciprocal		Rating methods		Pairwise comparison	Manually defined
Criteria	Straight Rank	Inverse Rank (n-r+1)	Normalized Weight*100	Reciprocal Rank (1/r)	Normalized Weight*100	Rate	Weight	Weight	Weight
Race	1	10	13.7	1	23.33	100	**21.8**	19.5	**24**
Income	1	10	13.7	1	23.33	100	**21.8**	19.5	**24**
Population	2	9	12.3	0.5	11.67	60	**13**	14.64	**12**
% Canopy Cover	2	9	12.3	0.5	11.67	60	**13**	14.64	**12**
% Green Space	3	8	11	0.333	7.9	40	**8.7**	9.76	**8**
Number of Trees Removed	4	7	9.6	0.25	5.8	30	**6.5**	7.32	**6**
Number of Trees Planted	4	7	9.6	0.25	5.8	30	**6.5**	7.32	**6**
Proximity to Industrial Centers	4	7	9.6	0.25	5.8	30	**6.5**	7.32	**6**
Proximity to Hospitals	5	6	8.2	0.2	4.7	10	**2.2**	0	**2**
		73	100	4.283	100	460	100	100	100

Table 2: Inequity Index

In *Rank method*, criteria first arranged in straight rank order 1 to 5 where the rank 1 shows the most important, 2 next important, and 5 the least important one. There are different methods to drive weights after ranking such as rank sum, rank reciprocal, rank exponential. In this research project, rank sum and reciprocal methods were used. To produce the inverse rank values in rank sum method each criterion's rank subtracted from the sum of ranks and then summed with 1:

Inverse Rank = n-rj+1

Each criterion's rank was reversed to produce the reciprocal rank values in rank reciprocal:

Reciprocal Rank = 1/r

Then the weight for a criterion in rank sum and rank reciprocal methods was calculated by dividing the inverse/ reciprocal rank of that criterion by the total score of all criteria and multiplied 100. In *Rate method*, each criterion is assigned a rank 0 to 100 based on a relative scale of importance and number 100 was assigned to the variables with the highest importance (race and income) and number 10 to the variables with the least importance(proximity to hospitals). The weight for a criterion was then calculated by dividing the rank of that criterion by the total score of all criteria.

In *Pairwise comparison method*, we created a ratio matrix where every criterion was compared with every other and each cell was given a criterion representing the most important. For two criteria of equal importance we put both criteria in the corresponding cell. Table 3 shows the matrix. Next, for each criterion we counted the number of cells containing the criterion flag letter (Table4) and for weighting out of 100 we solved the obtained equation.

Criteria	A	B	C	D	E	F	G	H	I
Race	A	AB	A	A	A	A	A	A	A
Income	B		B	B	B	B	B	B	B
Population	C			CD	C	C	C	C	C
% Canopy Cover	D				D	D	D	D	D
% Green Space	E					E	E	E	E
Number of Trees Removed	F						FG	FH	F
Number of Trees Planted	G							GH	G
Proximity to Industrial Centers	H								H
Proximity to Hospitals	I								

Table 3 Criteria Change

Criteria	number of times of importance	Weight
Race (A) Income (B)	8	8x2.439=19.5
Population (C) Percent canopy (D)	6	6x2.439=14.64
Percent green space(G)	4	4x2.439=9.76
Number of trees removed (E) Number of trees planted(F) Percent green space(G), Proximity to industrial centers (H)	3	3x2.439=7.32
Proximity to hospitals(I)	0	0

Table 4 Criteria Times

100=8x+8x+6x+6x+4x+3x+3x+3x -> x=2.439

Between results of all examined weighting methods, we chose the Rating method because its result is the closest to desired weights outlined by our sponsor.

Once all of the variables have been weighted, we calculated a score for each criterion by following formula:

$Rank_i \times Weight_i = Score_i$

Each tract receives a total score by summing all criteria scores:

$$\text{Total\_Score} = \sum_{k=1}^{n} \text{Score} \qquad \text{n= number of criteria}$$

After calculating a final score for each tract unit, a final inequity ranking map was generated (Figure 12).

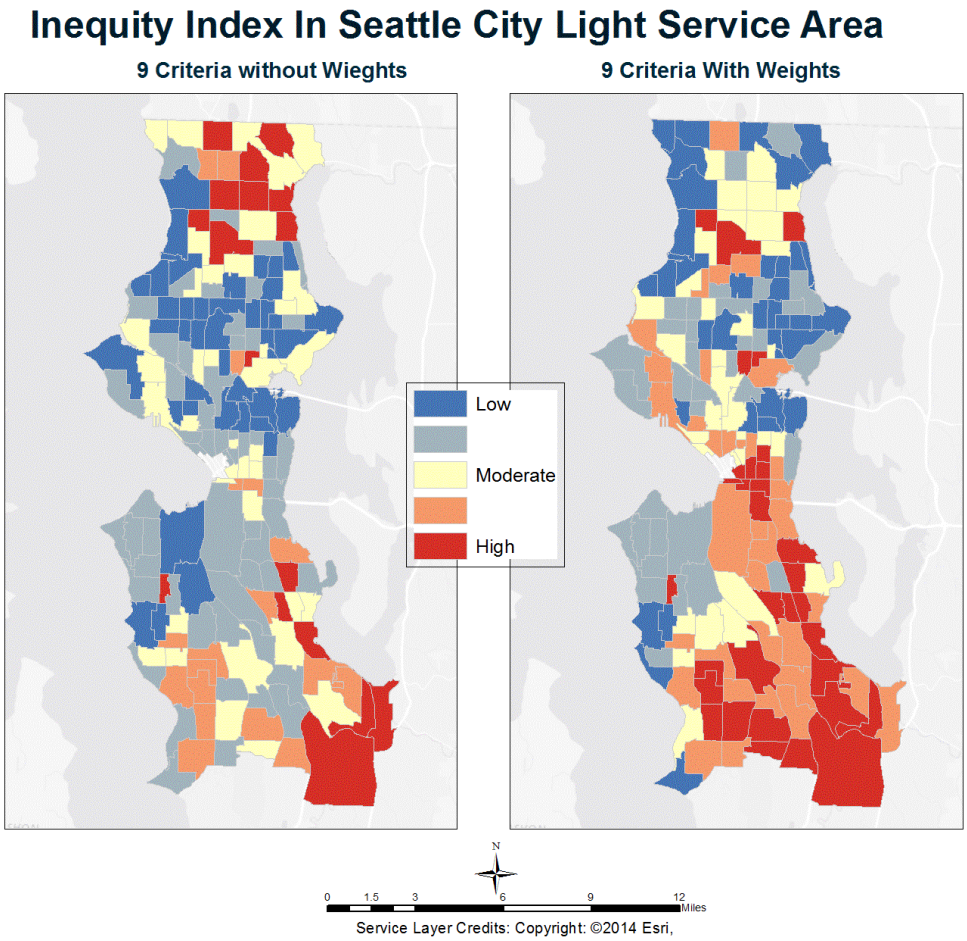

Figure 12 Inequity With & Without Weights

Results

Canopy Estimation

The results for area within a 10 Ft. clearance distance of each individual and all feeders were in square feet. The amount of canopy is much larger looking at our first approach using the 2011 USGS percent canopy cover dataset (Table 5). Table 5 shows that there was over487.58 million square ft. of

Summary Statisitcs SCL Service Area & Canopy: 2011 USFS Percent		
Area	Total Area Sq. Ft. (Million)	Total Area Acres
SCL Service Area	2558.56	58736.59
Canopy	487.58	11193.25
Possible Canopy Impacted from Feede	197.69	4538.25

Table 5 Results from Percent Canopy Dataset

canopy found within SCL's service layer in 2011. The model produces an estimate of over 4,000 acres of tree canopy that might possible fall within the clearance distance of the overhead feeders. The results for the 2000 LIDAR height data were much less in comparison to the percent canopy (Table 6). The

Canopy & SCL Feeder Results: 2000 LIDAR Height				
Canopy within 10 Ft. Clearance	Total Square Ft.	Average Per Feeder Sq. Ft	Total Acres	Average Per Feeder Acres
All Feeders in Service Area	2018359.91	11872.71	46.34	0.27

Table 6 Results from Height Derived Dataset

total amount of canopy residing within a 10 Ft. clearance is 46.32 acres. The average per feeder is indicated and shows how much impact the feeders may have across the entire service area.

Running the two models also produces results at the overall focal scale of our project. Figure 13 shows the results look like at a much finer scale. The dark green cells are canopy that within the clearance height for an arterial road which is 35.5 ft. and 60.5 ft. The light green cells or portions of cells are where the cell intersected with the 10 ft. clearance laterally but did not fall within the clearance of the feeder. For this sample area in the north eastern portion of SCL's service area there is a total area of over 16,000 square ft. of canopy found within the clearance distance of the feeder.

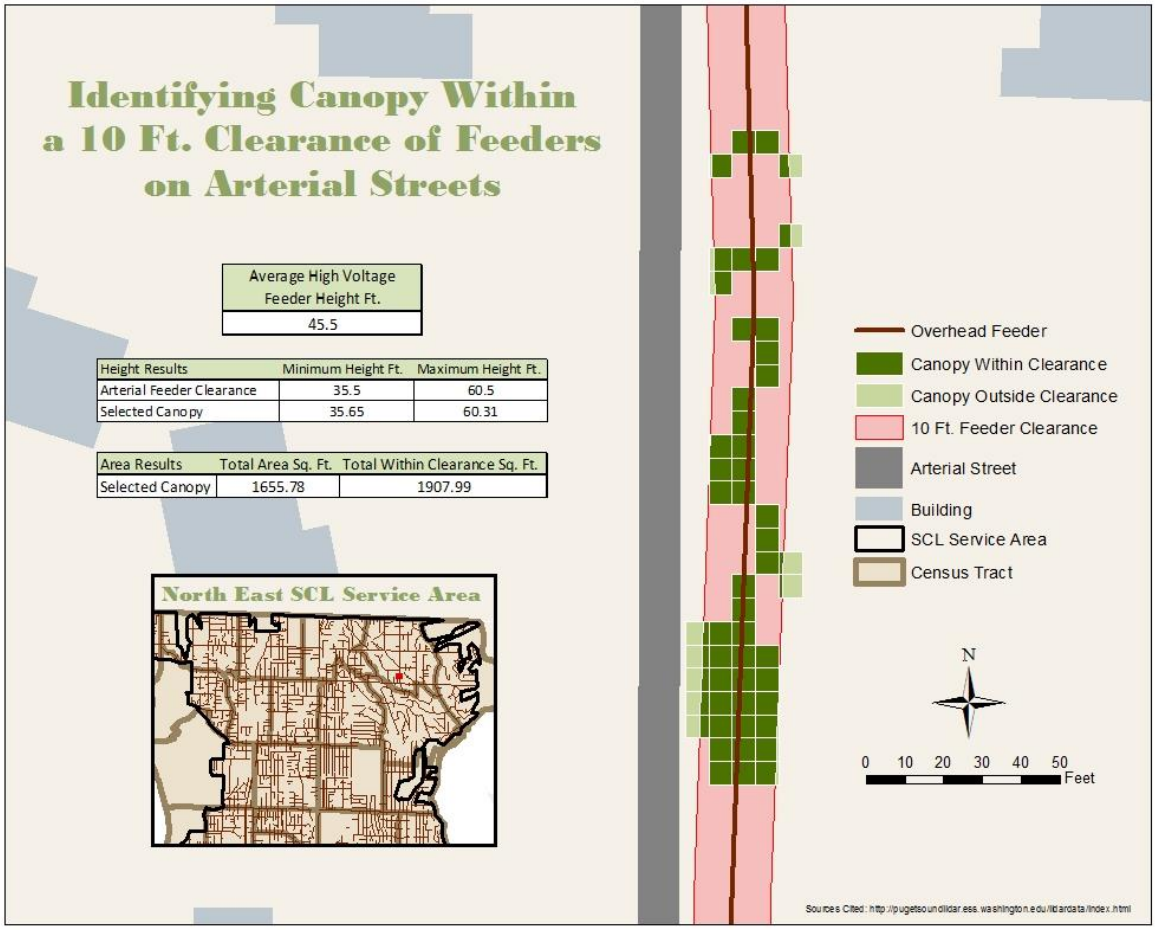

Figure 13 Example Result from Height Dataset

Inequity & MCE Analysis

The comprehensive inequity map represents a combined score based on these weighted criteria where the tract receiving the highest score (with red color) will be recommended for considering as at most risk of inequity area. To ensure the assigned weights are sensible, we developed a scenario (zero scenario) in which we considered all criteria to have equal importance, Therefore we attributed the same weight out of 100 and compared the result map (Figure 13) with inequity ranking map (Figure 13) of the basic scenario (scenario 1) where each variable were assigned proportional weights relative to their importance.

Discussion

Our study found significant spatial patterns of 9 criteria. We found significant clusters of both minority (percent non-white) and income. In the north region of study area (SCL service area) the minority is average, in the center is lowest and in south and south east is the highest. Income closely follows this race pattern. In income map we can see the highest income area lies on shorelines mainly where the property values are high and minority is lower. As we go to the south and south east part, the median income decreases and it correlates to higher minority. Northern side shows the average income that corresponds to average minority. In total population map, we have less specific trend but the most notable point is that at industrial neighborhood the population is the lowest. The canopy coverage map clearly shows the higher coverage at northern side, medium coverage in center and in the south part and lowest in south east. By visually comparing these four maps as our four main criteria, we conclude that most vulnerable areas for inequity are in south east of study area. The green space map shows mainly a uniform distribution of green spaces across the study area. In number of removed trees map we can see that in north and south more trees are removed but in the center fewer trees have been removed.

It is important to note that we only had data available for city of Seattle number of trees planted. Thus we Imputed non-data tracts to the average.

Comparing the removed and plant trees map we can find areas where more trees have been removed and less trees have been planted or the other way around.

The proximity to industrial centers and hospitals clearly show the distance in miles from industrial centers and hospitals.

By comparing scenario 0 and scenario 1 (maps 4a and 4b respectively) we can see that some of the tracts in north and some in southeast will not change and yet have the highest risk of inequity. It also shows that considering weighting can potentially change the outcome.

Sensitivity:

A sensitivity analysis is a method to show and understand the stability and robustness of the criteria and weights. For this project, we performed a sensitivity analysis to see the effects on the overall results by changes in key variables. For this study, we developed five scenarios (table5).
The MCE analysis first tests the combined effects of the criteria on the response. Then sensitivity analysis eliminates the four most important predictors one by one: without_Race (scenario 2), without_Income (scenario 3), without_Population (scenario 4) and without_Canopy (scenario 5) and it retains the four most important predictors and eliminates weakest predictors (scenario 6). Table 7 shows the all scenarios. A separate multi-criteria analysis is applied to the scenarios. After weighting the criteria by rating method, a score was calculated. The inequity score was provided as sum of all scores. The results of all six scenarios for each tract can been seen in appendix A.

The results of sensitivity analysis shows that chosen criteria are correct, however, weighting process in sensitivity analysis that we employed was not necessarily accurate thus there are no significant changes in outcome results was observed.

	Weights						
Criteria	Scenario 0 (equal weights)	Scenario 1 (basic scenario)	Scenario 2 (no-Race Scenario)	Scenario 3 (no-Income Scenario)	Scenario 4 (no-Population Scenario)	Scenario 5 (no-Canopy Scenario)	Scenario 6 (Primary)
Race	11.11	21.8	X	27.8	25	25	31.25
Income	11.11	21.8	27.8	X	25	25	31.25
Population	11.11	13	16.7	16.7	X	15	18.75
% Canopy Cover	11.11	13	16.7	16.7	15	X	18.75
% Green spaces	11.11	8.7	11.1	11.1	10	10	X
Number of Trees Removed	11.11	6.5	8.3	8.3	7.5	7.5	X
Number of Trees Planted	11.11	6.5	8.3	8.3	7.5	7.5	X
Proximity to Industrial Centers	11.11	6.5	8.3	8.3	7.5	7.5	X
Proximity to Hospitals	11.11	2.2	2.8	2.8	2.5	2.5	X

Table 7 Sensitivity

Limitations and Combined Results

Over the course of these 8 weeks in which we have been working on this project we have established methods, designs, and results that can serve as a basis for continued future implementation. One of the most important things that we have realized from our analysis is the many assumptions and limitations in our design. We can look at these limitations and understand what course of action should be taken in the future reduce the impacts of these limitations. Although the model results are imperfect they serve as a starting point for analyzing trimmed canopy at a larger scale.

In our first approach we looked at the percent canopy dataset. The results were substantially larger than those from the 2000 height data. The results found in Table 5 are actually an overestimation of the amount of canopy that would actual fall within the clearance distance and thus be removed. These results represent the max amount of canopy that possibly could be trimmed because of its proximity to the actual feeder line. These numbers are used in the business case outlined below simple to show an example of how having these quantities could help identify even more so the impact that trimming trees has on the social, economic, and biophysical systems found within SCL's service area. Because the data set is a percentage of canopy cover per area grid we don't know exactly where that canopy geographically is located. For this reason it is only a max amount of canopy that might have to be removed.

As seen in Figure 13 using the 2000 LIDAR derived height data produces more accurate results. The main limitation of this process is the fact that the data is 14 years old. The results are more accurate geographically. We know that that canopy cell is located in that location at a height that would put its branches in areas that needs to be trimmed. In the sense that we can get an accurate location and at least semi-accurate area of canopy the method is far superior to looking at merely the percent canopy. In the future when more current height or LIDAR data becomes available it can be run through this process to get more accurate results in the sense that the more current the data the more closely it represents reality. The other limitation that is of importance is that we are only looking at canopy from

the top looking down. The results produces are in sq. ft., acres, etc. The most effective way to measure the value of canopy would be to calculate the volume or tonnage. Another possible future implementation would be to incorporate the bottom height of the canopy so that it would be possible to get closer to estimating a volume of canopy that is actually taken out of the system.

The design that we have implemented allows for looking at the amount of canopy and the social equity involved in SCL's service area. Figure 14 shows a sample of the Inequity Index results in comparison to the estimation of quantity of canopy. The results are fascinating and actually display

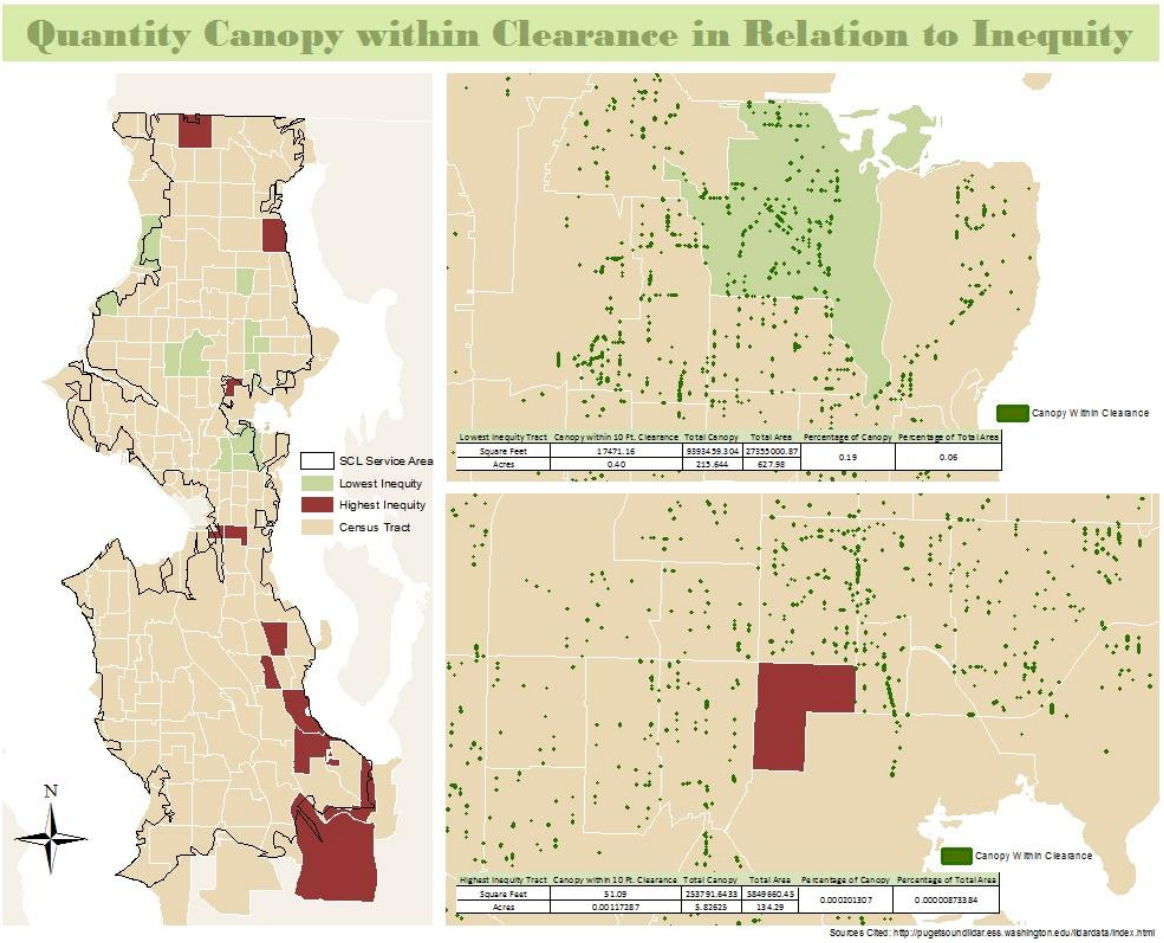

Figure 14 Comparing the Two Method Results

SCL trimming impacts doing well in regards to inequity. It is interesting to note that the areas with the highest inequity index scores are spread across the SCL service area more than those with

lower inequity. It is apparent by the percentages and quantities of area of canopy that is within a 10 ft. clearance distance in relation to the amount of canopy within those tracts that SCL would not remove large volumes of quantity in the highest area of inequity based on our index. The reverse is true for the area with the lowest inequity. Although these results might not be that accurate or telling because of the previous implementations mentions, they represent a structure in which you can use better data to analyze two different aspects of SCL maintenance work on trees to protect the power they provide to so many people.

Future Implementation

To our knowledge no one has conducted any analysis for evaluating the quantities of trimmed trees from utilities at the scale we attempted to for this project. We suggest for SCL to seek to obtain more current canopy data to apply to the design that we have tested. Our conceptual design, modelling, and representations hopefully will serve as a base where others can build from in more fully understanding the net impact that SCL has on the urban forest that also serves the people within SCL's service are in so many intangible ways.

Business Case

Project Name: Seattle City Light

Project Purpose: Our sponsor is the Seattle City Light vegetation management department that is responsible for over 1,700 miles of lines running through eight cities and Unincorporated King County, and 657 miles of transmission right-of-way spread across five counties. Part of the Vegetation Management unit's task is to trim trees and bushes off of the feeder lines to ensure interrupted power supply to their customers. Since the Seattle City Light trims large volumes of vegetation each year; and given the ecosystem services that urban tree canopy provides, Seattle City Light has commission our team to gauge the impact of its activities.

Project Scope Summary: To achieve our goal of assessing the overall impact of Seattle City Light's activities, our team is required to provide the following documents:

- Socio Ecological Systems Table
- Socio Ecological Systems Map
- Threshold Matrix
- Maps (including a Story Map on ArcGIS Online)

Basis of Our Financial Analysis

Our financial analysis is based on the metrics presented in a similar study called: Measuring Urban Green-High Resolution mapping of Chicago's tree canopy and Financial Valuation of Corresponding Environmental Services by Christopher Kowal.

Acres of UTC	Pollutant	Pounds Removed
1	Ozone (O3)	25.53
1	Sulfur Dioxide (SO2)	11.24
1	Nitrogen Dioxide	11.87
1	ParticulateMater <10 micrometers (PM10)	28.21
1	Carbon Monoxide(CO)	2.05

Table 7 Financial Metrics

We used the metrics of that study to compute the Potential Air Pollution Removal of Canopy Impacted from feeders.

Summary Statistics SCL SERVICE AREA & CANOPY: 2011 USGS PERCENT CANOPY COVER		
Area	Total Area Sq. Ft.	Total Area in Acres
SCL Service Area	2558555857.03	58736.59

Canopy	**487577846.85**	**11193.25**
Possible Canopy impacted from feeders	**197686150.46**	**4538.25**

Table 8 Summary Statistics

Potential Air Pollution Removal of Canopy Impacted From Feeders

Pollutant	Lbs Removed	Multiplier	Dollar Value
Ozone (O3)	**115868.5**	**3.375**	**391,056.18**
Sulphur Dioxide (SO2)	**51049.8**	**0.825**	**42116.08**
Nitrogen Dioxide (NO2)	**53885.84**	**3.375**	**181,864**
Particulate Matter < 10 micrometers (PM10	**128022.62**	**2.25**	**288050.89**
Carbon Monoxide (CO)	**9318.02**	**0.475**	**4426.06**
Total	**358,144.78**	**N/A**	**907513.21**

Table 9 Pollutants

Despite the fact that the activities of Seattle City Light are justified to ensure reliable supply of electrical power, it is important to show the functional and monetary value of the removed canopy.

Economic Benefits Summary

Annual Service	Annual Economic Benefit
Air pollution Removal	**$907,484.77**
Carbon Sequestration	**$6,079.97**
Storm Water Savings	**$2,662,275.77**
Total Annual Savings	**$3,575,840.51**

Table 10 Economic Benefits

Analysis

Based on our geospatial analysis and financial valuation of the trimming activities of Seattle City Light, we have reached the following conclusions: We would recommend Seattle City Light to invest to address the following issues:

- Social Inequity-in terms of UTC distribution within its jurisdiction and
- *Ecosystems Services that the UTC provides- SCL should at least adopt best practices in its canopy trimming activities. From our financial valuation, it became apparent that UTC has value in terms of the ecosystem services it provides; these intangible benefits include but are not limited to Water Shed Benefits, Interception Capacity, Energy Benefits-Shading, Transpiration, Wind Speed Reduction; Variations in Energy Benefits, Air Quality Benefits.*

Literature Cited

Akbari H., M. Pomerantz, and H. Taha. 2001. Cool surfaces and shade trees to reduce energy use and improve air quality in urban areas. Solar Energy, 70 (3): 295-310

Bayard, David. 2014. Seattle City Light's Vegetation Management. Presented to Seattle City meeting, Seattle, Washington, July.

Ciecko, Lisa., Tenneson, Karis., Dilley, Jada., and Kathleen Wolf. 2012. "Seattle's Forest Ecosystem Values Analysis of the Structure, Function, and Economic Benefits. Accessed August 10, 2014 http://www.fs.fed.us/pnw/research/gcra/pdfs/FEVSeattlePublicReport-20120830-final.pdf

Coder, Kim D., 2011. Community Tree Canopy Loss: Calculations & Perceptions. *Community Forestry Series.* Warnell School of Forestry and Natural Resources the University of Georgia. July.

Davis, Kimberly L., and Robert E. Jones. 2014. "Modeling Environmental Concern for Urban Tree Protection Using Biophysical and Social Psychological Indicators. *Society & Natural Resources: An International Journal*, 27:4, 362 – 388, DOI 10.1080/08941920.2013.861555

Hoyer, Eric H. 2013. Infrastructure and Eminent Domain – Appraising Trees When Damaged or Removed For Utilities or Roadways. The Council Quarterly Issue 4, pp 2 - 3. Florida Urban Forestry Council.

Kaplan, Rachel. "The Social Values of Forests and Trees in Ubanized Societies". Accessed August 15, 2014 http://courses.washington.edu/esrm200/Kaplan_Social_Values.pdf

Kowal, Christopher. 2007. "Measuring Urban Green – High Resolution Mapping of Chicago's Tree Canopy and Financial Valuation of Corresponding Environmental Services. Accessed August 15, 2014 ttp://www.pangaeatech.com/ckowal/AICPstudentproject.htm#tab6

Mapes, Lynda V., and Justin Mayor. 2014. "A fight for urban trees: Seattle's wealthier neighborhoods leafier. *Seattle Times*, August 13. Accessed August 14th, 2014 http://seattletimes.com/html/localnews/2024305935_treecanopyxml.html

Northrop, Rob. 2013. Reducing Conflicts Between Urban Infrastructure and Trees. The Council Quarterly Issue 4, pp 1. Florida Urban Forestry Council.

Nowak, D.J., D.E. Crane, and J.C. Stevens. 2006. Air pollution removal by urban trees and shrubs in the United States. Urban Forestry & Urban Greening 4 (3–4): 115-123

Portland's Urban Forest Canopy Assessment and Public Tree Evaluation. 2007. Accessed August 17, 2014 http://www.portlandonline.com/shared/cfm/image.cfm?id=171829

Appendix A

Sensitivity Tables

GEO_ID_TRT	Total_S0	Total_S1	Total_S2	Total_S3	Total_S4	Total_S5	Total_S6
53033000100	400.00	369.60	361.20	361.20	365.00	365.00	400.00
53033000200	322.22	321.80	327.80	300.00	295.00	355.00	331.25
53033000300	300.00	300.20	300.00	272.20	315.00	315.00	293.75
53033000401	355.56	361.00	350.00	350.00	355.00	385.00	362.50
53033000402	322.22	317.40	350.10	294.50	305.00	320.00	318.75
53033000500	233.33	186.90	183.20	211.00	185.00	200.00	150.00
53033000600	355.56	367.50	358.40	358.40	347.50	392.50	381.25
53033000700	288.89	304.60	277.80	277.80	320.00	320.00	325.00
53033000800	266.67	217.50	222.20	194.40	220.00	235.00	212.50
53033000900	300.00	226.00	233.30	233.30	245.00	200.00	218.75
53033001000	244.44	226.20	205.50	233.30	245.00	230.00	212.50
53033001100	244.44	208.70	211.00	211.00	210.00	225.00	181.25
53033001200	311.11	343.60	328.00	328.00	335.00	335.00	400.00
53033001300	311.11	330.60	311.20	311.20	350.00	320.00	362.50
53033001400	244.44	243.50	255.70	227.90	220.00	250.00	268.75
53033001500	233.33	217.30	222.10	222.10	220.00	220.00	200.00
53033001600	266.67	243.40	255.40	255.40	250.00	250.00	200.00
53033001701	311.11	330.40	366.60	311.00	350.00	320.00	300.00
53033001702	277.78	297.90	324.90	269.30	312.50	312.50	262.50
53033001800	300.00	273.80	294.50	294.50	270.00	255.00	256.25
53033001900	233.33	226.20	233.40	205.60	230.00	230.00	231.25
53033002000	300.00	254.30	269.30	269.30	262.50	262.50	200.00
53033002100	266.67	226.10	233.20	233.20	230.00	245.00	181.25
53033002200	311.11	251.90	266.60	294.40	230.00	245.00	225.00
53033002400	311.11	252.10	266.50	266.50	260.00	260.00	200.00
53033002500	255.56	213.00	216.60	244.40	215.00	215.00	168.75
53033002600	255.56	232.50	241.80	269.60	222.50	207.50	225.00
53033002700	255.56	280.30	302.90	302.90	262.50	262.50	275.00
53033002800	266.67	295.70	322.30	266.70	310.00	280.00	300.00
53033002900	266.67	273.80	294.40	294.40	285.00	255.00	237.50
53033003000	255.56	280.30	302.90	302.90	262.50	262.50	275.00
53033003100	288.89	299.80	327.80	327.80	285.00	285.00	275.00
53033003200	322.22	356.40	400.10	344.50	335.00	350.00	356.25
53033003300	255.56	280.30	302.90	302.90	262.50	262.50	275.00
53033003400	266.67	273.80	294.40	294.40	285.00	255.00	237.50
53033003500	244.44	247.80	261.00	261.00	255.00	255.00	200.00
53033003600	300.00	304.20	333.40	305.60	290.00	290.00	306.25
53033003800	255.56	221.80	227.70	227.70	240.00	225.00	181.25

GEO_ID_TRT	Total_S0	Total_S1	Total_S2	Total_S3	Total_S4	Total_S5	Total_S6
53033003900	277.78	247.80	261.10	261.10	255.00	240.00	218.75
53033004000	266.67	243.40	255.60	255.60	250.00	220.00	237.50
53033004100	311.11	278.00	300.10	327.90	245.00	260.00	262.50
53033004200	266.67	239.00	250.10	277.90	200.00	245.00	225.00
53033004301	233.33	208.70	211.10	211.10	210.00	210.00	200.00
53033004302	311.11	313.20	344.50	261.10	330.00	315.00	312.50
53033004400	277.78	278.30	300.10	244.50	260.00	290.00	300.00
53033004500	277.78	267.30	286.10	286.10	277.50	247.50	237.50
53033004600	233.33	204.20	205.40	233.20	205.00	205.00	168.75
53033004700	277.78	323.90	358.50	302.90	312.50	312.50	337.50
53033004800	277.78	286.80	311.10	311.10	285.00	270.00	256.25
53033004900	311.11	343.40	383.40	327.80	335.00	335.00	337.50
53033005000	311.11	313.00	344.40	288.80	330.00	300.00	300.00
53033005100	255.56	260.80	277.70	277.70	270.00	255.00	218.75
53033005200	333.33	387.00	383.40	383.40	385.00	385.00	400.00
53033005301	355.56	395.80	394.70	366.90	380.00	395.00	450.00
53033005302	311.11	356.70	344.70	316.90	350.00	350.00	431.25
53033005400	300.00	321.60	355.60	327.80	310.00	310.00	306.25
53033005600	277.78	256.20	272.20	300.00	235.00	235.00	243.75
53033005700	233.33	249.90	263.90	263.90	227.50	257.50	237.50
53033005801	311.11	343.40	383.40	327.80	335.00	335.00	337.50
53033005802	311.11	343.40	383.40	327.80	335.00	335.00	337.50
53033005900	277.78	286.80	311.10	311.10	255.00	300.00	256.25
53033006000	266.67	273.80	294.40	294.40	255.00	285.00	237.50
53033006100	266.67	282.40	305.60	305.60	265.00	265.00	275.00
53033006200	211.11	173.90	166.60	194.40	170.00	185.00	150.00
53033006300	255.56	243.40	255.70	255.70	220.00	235.00	256.25
53033006400	255.56	213.00	216.60	244.40	215.00	215.00	168.75
53033006500	244.44	247.80	261.00	261.00	255.00	255.00	200.00
53033006600	266.67	304.40	333.40	277.80	320.00	290.00	300.00
53033006700	288.89	299.80	327.80	327.80	285.00	285.00	275.00
53033006800	244.44	252.10	266.60	266.60	260.00	245.00	218.75
53033006900	266.67	243.40	255.40	255.40	250.00	250.00	200.00
53033007000	311.11	349.90	391.80	336.20	327.50	342.50	356.25
53033007100	288.89	317.40	350.00	294.40	335.00	305.00	300.00
53033007200	300.00	336.90	375.10	319.50	327.50	327.50	337.50
53033007300	288.89	330.40	366.80	311.20	335.00	305.00	337.50
53033007401	288.89	317.40	350.00	294.40	335.00	305.00	300.00

GEO_ID_TRT	Total_S0	Total_S1	Total_S2	Total_S3	Total_S4	Total_S5	Total_S6
53033007402	322.22	356.40	400.10	344.50	350.00	335.00	356.25
53033007500	300.00	336.90	375.10	319.50	327.50	327.50	337.50
53033007600	300.00	310.90	341.70	286.10	327.50	297.50	300.00
53033007700	266.67	297.90	297.20	297.20	312.50	297.50	281.25
53033007800	277.78	278.00	300.00	327.80	260.00	260.00	243.75
53033007900	311.11	358.70	375.10	347.30	352.50	352.50	368.75
53033008001	277.78	299.80	327.90	327.90	285.00	270.00	293.75
53033008300	288.89	317.40	350.00	294.40	335.00	305.00	300.00
53033008400	288.89	317.40	350.00	294.40	335.00	305.00	300.00
53033008500	322.22	382.80	377.80	350.00	410.00	380.00	393.75
53033008600	311.11	374.00	366.80	366.80	370.00	370.00	400.00
53033008700	311.11	361.00	350.00	350.00	385.00	355.00	362.50
53033008800	277.78	335.00	316.70	316.70	355.00	340.00	343.75
53033008900	300.00	336.90	319.50	375.10	327.50	327.50	337.50
53033009000	333.33	404.60	377.80	377.80	435.00	405.00	425.00
53033009100	344.44	417.60	394.50	394.50	450.00	405.00	443.75
53033009200	322.22	382.80	377.80	350.00	410.00	380.00	393.75
53033009300	277.78	335.00	316.70	316.70	370.00	325.00	343.75
53033009400	322.22	395.80	366.80	394.60	395.00	395.00	431.25
53033009500	288.89	330.40	311.20	366.80	320.00	320.00	337.50
53033009600	288.89	269.40	288.80	288.80	250.00	280.00	237.50
53033009701	300.00	275.80	297.20	297.20	257.50	257.50	275.00
53033009702	277.78	249.80	263.80	263.80	227.50	257.50	237.50
53033009800	288.89	282.40	305.60	305.60	265.00	265.00	275.00
53033009900	266.67	286.80	311.20	311.20	270.00	270.00	275.00
53033010001	300.00	339.40	294.40	322.20	360.00	360.00	356.25
53033010002	300.00	337.10	319.50	319.50	357.50	327.50	362.50
53033010100	333.33	369.60	361.20	361.20	365.00	365.00	400.00
53033010200	300.00	297.80	269.40	325.00	282.50	312.50	300.00
53033010300	355.56	389.20	358.40	386.20	387.50	387.50	431.25
53033010401	277.78	284.90	224.90	308.30	297.50	297.50	293.75
53033010402	300.00	328.30	280.60	364.00	332.50	317.50	350.00
53033010500	288.89	273.70	294.50	294.50	255.00	255.00	275.00
53033010600	300.00	280.20	302.80	302.80	247.50	277.50	275.00
53033010701	277.78	289.30	258.20	258.20	302.50	317.50	306.25
53033010702	355.56	376.20	341.60	369.40	402.50	372.50	393.75
53033010800	255.56	260.90	222.10	277.70	255.00	285.00	262.50
53033010900	300.00	300.00	327.70	272.10	330.00	285.00	281.25

GEO_ID_TRT	Total_S0	Total_S1	Total_S2	Total_S3	Total_S4	Total_S5	Total_S6
53033011001	355.56	400.20	372.20	372.20	430.00	400.00	425.00
53033011002	344.44	378.40	344.40	372.20	405.00	375.00	393.75
53033011101	311.11	358.90	319.50	347.30	382.50	352.50	393.75
53033011102	322.22	341.40	325.00	325.00	362.50	332.50	362.50
53033011200	288.89	326.20	305.60	305.60	345.00	315.00	362.50
53033011300	288.89	315.30	291.70	319.50	302.50	332.50	331.25
53033011401	311.11	315.20	291.60	319.40	332.50	302.50	331.25
53033011402	344.44	354.40	341.60	341.60	377.50	347.50	362.50
53033011500	266.67	247.70	261.10	261.10	255.00	225.00	237.50
53033011600	266.67	241.10	252.70	252.70	217.50	247.50	237.50
53033011700	311.11	356.70	316.70	344.50	350.00	380.00	393.75
53033011800	355.56	400.00	372.30	400.10	385.00	400.00	450.00
53033011900	333.33	404.40	377.90	405.70	390.00	405.00	450.00
53033012000	322.22	267.20	286.00	286.00	277.50	247.50	237.50
53033012100	288.89	215.10	219.30	247.10	217.50	217.50	168.75
53033020100	322.22	241.20	252.80	252.80	247.50	217.50	237.50
53033020200	322.22	234.70	244.40	244.40	210.00	255.00	218.75
53033020300	400.00	347.90	333.30	333.30	340.00	370.00	362.50
53033020401	322.22	228.20	236.00	236.00	232.50	232.50	200.00
53033020402	355.56	284.80	308.30	252.70	267.50	312.50	281.25
53033020500	355.56	293.50	319.40	263.80	277.50	322.50	281.25
53033020600	333.33	274.00	238.80	294.40	285.00	285.00	262.50
53033020700	344.44	295.80	294.40	266.60	310.00	310.00	293.75
53033020800	300.00	208.60	211.00	238.80	210.00	210.00	168.75
53033020900	266.67	206.60	208.30	208.30	207.50	222.50	181.25
53033021000	355.56	308.70	283.30	338.90	295.00	325.00	300.00
53033021100	355.56	302.30	302.70	274.90	317.50	317.50	293.75
53033021300	355.56	300.00	327.80	272.20	315.00	285.00	300.00
53033021400	311.11	223.90	230.40	230.40	227.50	242.50	181.25
53033021500	311.11	206.40	208.20	236.00	207.50	207.50	168.75
53033025302	355.56	350.10	336.20	336.20	372.50	327.50	381.25
53033026001	333.33	336.80	319.40	375.00	327.50	327.50	337.50
53033026002	366.67	410.90	386.20	414.00	397.50	412.50	450.00
53033026100	322.22	380.60	347.20	375.00	377.50	407.50	393.75
53033026200	355.56	406.50	408.40	408.40	407.50	392.50	418.75
53033026300	300.00	348.00	333.30	333.30	385.00	340.00	343.75
53033026400	322.22	367.50	358.30	358.30	362.50	392.50	362.50

GEO_ID_TRT	Total_S0	Total_S1	Total_S2	Total_S3	Total_S4	Total_S5	Total_S6
53033026500	333.33	356.60	344.40	344.40	380.00	350.00	362.50
53033026600	311.11	324.10	302.60	302.60	357.50	342.50	306.25
53033026700	333.33	341.30	352.70	324.90	332.50	362.50	331.25
53033026801	344.44	376.10	369.50	369.50	372.50	372.50	400.00
53033026802	344.44	376.10	369.50	369.50	372.50	372.50	400.00
53033027000	300.00	341.50	324.90	324.90	362.50	362.50	325.00
53033027100	300.00	341.50	324.90	324.90	362.50	362.50	325.00
53033027200	300.00	332.70	313.80	341.60	352.50	337.50	312.50
53033027300	333.33	374.00	366.60	366.60	370.00	400.00	362.50
53033027400	322.22	367.50	358.30	358.30	362.50	392.50	362.50
53033027500	344.44	367.50	358.50	358.50	362.50	362.50	400.00
53033027600	300.00	315.50	291.70	291.70	317.50	347.50	325.00
53033027800	288.89	247.80	261.00	261.00	255.00	255.00	200.00
53033027900	344.44	336.90	375.10	319.50	327.50	327.50	337.50
53033028000	300.00	354.50	341.70	341.70	377.50	347.50	362.50
53033028100	311.11	376.30	341.70	369.50	402.50	372.50	393.75
53033028200	344.44	393.50	391.70	391.70	392.50	392.50	400.00

Python Code for tools

SCL Get Selected Canopy Cover Python Script Tool code:

```
'''
   Name:         SCLGetSelectedCanopyCover.py
   Create Date:   08/16/2014
   Created by:    Ben Hillam
   Organization:  UW PMPGIS Capstone Project - Sponsor Seattle City Light
   Description:   This is the first of two python scripts to help SCL evaluate feeder impacts on canopy
cover
                  in its service area.  This script will take the dem height raster and get the qualifying
                  cells in association with the roads and feeder height.  The results can be put in the second
                  model to run to calculate the total area of canopy impacted
'''
# Import modules
import arcpy
from arcpy import env

# Local variables
inputBaseRaster = arcpy.GetParameterAsText(0)#r'C:\Grad_School\Geog_569\Seattle City Light
Project\Data\Lidar\baseElevation\basee'
inputTopSurfaceRaster = arcpy.GetParameterAsText(1)#r'C:\Grad_School\Geog_569\Seattle City Light
Project\Data\Lidar\topElevation\surfacee'
```

```
inputFeederFC = arcpy.GetParameterAsText(2) #r'C:\Grad_School\Geog_569\Seattle City Light
Project\Data\SCLCapstoneProject.gdb\OH_Feeders'
inputStreetsFC =
arcpy.GetParameterAsText(3)#r'C:\Users\bhillam\Documents\ArcGIS\Packages\Buildings and
Streets\v101\sclcgdb.gdb\Transportation\Street_Network'
inputBufferDistance = arcpy.GetParameterAsText(4)#'10'
outputLocationDEM = arcpy.GetParameterAsText(5)#r'C:\Grad_School\Geog_569\Seattle City Light
Project\Data\Lidar\testWorkspace'
outputLocationFCs = arcpy.GetParameterAsText(6)#r'C:\Grad_School\Geog_569\Seattle City Light
Project\Data\Lidar\testWorkspace\ValidCanopyResults.gdb'

# Set up workspace
env.workspace = outputLocationDEM
arcpy.env.overwriteOutput = True

try:
##   '''
##   ********************************************************************
##   CALCULATE THE TOP HEIGHT OF THE CANOPY
##   ********************************************************************
##   '''
##   #print 'Starting Calculations . . .\n'
##   arcpy.AddMessage('Starting Calculations . . .\n')
##   # Calculate the height from the base and surface rasters
##   #print 'Checking out license . .'
##   arcpy.AddMessage('Checking out license . .')
##   arcpy.CheckOutExtension("Spatial")
##   resultHeightDEMName = 'ResultSurfaceHeight.tif'
##   #print 'Creating the height DEM by subtracting top surface raster from the base raster . . .'
##   arcpy.AddMessage('Creating the height DEM by subtracting top surface raster from the base raster
. . .')
##   resultHeightDEM = arcpy.sa.Minus(inputTopSurfaceRaster, inputBaseRaster)
##   #print 'Times the value by 1000000 to preserve the decimal places 6 places . . . '
##   arcpy.AddMessage('Times the value by 1000000 to preserve the decimal places 6 places . . . ')
##   resultHeightDEMT6 = arcpy.sa.Times(resultHeightDEM, 1000000)
##   #print 'Converting values to Int in raster . . .'
##   arcpy.AddMessage('Converting values to Int in raster . . .')
##   resultHeightDEM_Int = arcpy.sa.Int(resultHeightDEMT6)
##   #print 'Saving result height DEM . . .'
##   arcpy.AddMessage('Saving result height DEM . . .')
##   resultHeightDEM_Int.save('{0}\\{1}'.format(outputLocationDEM, resultHeightDEMName))
##   #print 'Result height DEM complete.\n'
##   arcpy.AddMessage('Result height DEM complete.\n')
##   #print 'Checking license back in . . .'
##   arcpy.AddMessage('Checking license back in . . .')
##   arcpy.CheckInExtension("Spatial")
  resultSurfaceHeightDEM = '{0}\\{1}'.format(outputLocationDEM, 'ResultSurfaceHeight.tif')
```

```
    '''
    ****************************************************************
    CONVERT THE CANOPY RASTER TO A POLYGON
    ****************************************************************
    '''
    #print '\nAssigning feeder type by associated street classification . . .'
    arcpy.AddMessage('\nAssigning feeder type by associated street classification . . .')
    # Re-assign the workspace
    env.workspace = outputLocationFCs
    arcpy.env.overwriteOutput = True

##    #print 'Creating surface height polygon . . .'
##    arcpy.AddMessage('Creating surface height polygon . . .')
    resultSurfaceHeightPoly = '{0}\\{1}'.format(outputLocationFCs, 'ResultSurfaceHeight')
##    arcpy.RasterToPolygon_conversion(resultSurfaceHeightDEM, resultSurfaceHeightPoly,
"NO_SIMPLIFY")
##    #print 'Surface height polygon completed.'
##    arcpy.AddMessage('Surface height polygon completed')

##    '''
##    ****************************************************************
##    GET THE STREET TYPE THE FEEDER IS MOST LIKELY ON TO CALCULATE DISTANCE
##    ****************************************************************
##    '''
##    # First split the lines so they are all in segments
##    #print 'Spitting feeders into individual segments . . .'
##    arcpy.AddMessage('Spitting feeders into individual segments . . .')
##    feedersSplitFCName = 'FeedersSplit'
##    feedersSplitFC = '{0}\\{1}'.format(outputLocationFCs, feedersSplitFCName)
##    arcpy.FeatureToLine_management(inputFeederFC, feedersSplitFC)
##
##    # Next identify the nearest street/road to get what type of road it is
##    #print 'Finding the nearest road to the feeder . . .'
##    arcpy.AddMessage('Finding the nearest road to the feeder . . .')
##    arcpy.Near_analysis(feedersSplitFC, inputStreetsFC)
##
##    # Add a field to determine what type of road it is
##    #print 'Adding ArterialRoad field to feeders split FC . . .'
##    arcpy.AddMessage('Adding ArterialRoad field to feeders split FC . . .')
##    arcpy.AddField_management(feedersSplitFC, "ArterialFeeder", "SHORT")
##
##    # Join the "FEaCODE" field to the feeders split FC to calculate the
##    #print 'Joining street network to feeders . . .'
##    arcpy.AddMessage('Joining street network to feeders . . .')
##    arcpy.JoinField_management(feedersSplitFC, "NEAR_FID", inputStreetsFC, "OBJECTID", "FEACODE")
##
##    # Calculate whether the feeder is on an arterial street based on thejoin fiedl
##    #print 'Identifying feeders on arterial roads . . .'
```

```
##    arcpy.AddMessage('Identifying feeders on arterial roads . . .')
##    expression = 'getStreetType(!FEACODE!)'
##    codeBlock = '''def getStreetType(feaCode):
##       if feaCode == 1:
##          return 1
##       else:
##          return 0'''
##
##    arcpy.CalculateField_management(feedersSplitFC, "ArterialFeeder", expression, "PYTHON",
codeBlock)
##
##    '''
##    *****************************************************************
##    CREATE SEPERATE BUFFERS FOR ARTERIAL AND NON ARTERIAL FEEDERS
##    *****************************************************************
##    '''
##    # Need to make feature layer of feedersSplitFC first
##    feedersSplitFeatureLayer = 'feedersSplitFeatureLayer'
##    #print 'Creating feature layer for the feeders feature class . . . '
##    arcpy.AddMessage('Creating feature layer for the feeders feature class . . . ')
##    if arcpy.Exists(feedersSplitFeatureLayer) == False:
##       arcpy.MakeFeatureLayer_management(feedersSplitFC, feedersSplitFeatureLayer)
##    else:
##       arcpy.Delete_management(feedersSplitFeatureLayer)
##       arcpy.MakeFeatureLayer_management(feedersSplitFC, feedersSplitFeatureLayer)
##
##    # Select only primary arterial roads with "FEACODE" = 1 for first buffer
##    arcpy.SelectLayerByAttribute_management(feedersSplitFeatureLayer, "NEW_SELECTION",
"ArterialFeeder = 1")

    # Create buffer of only the arterial roads
    arterialBufferFCName = 'FeedersArterial_{0}FtBuff'.format(inputBufferDistance)
    arterialBufferFC = '{0}\\{1}'.format(outputLocationFCs, arterialBufferFCName)

##    #print 'Creating arterial feeder buffer . . .'
##    arcpy.AddMessage('Creating arterial feeder buffer . . .')
##    arcpy.Buffer_analysis(feedersSplitFeatureLayer, arterialBufferFC, inputBufferDistance + ' Feet')

    # Create arterial feeder buffer feature layer
    arterialFeedersBuffFeatureLayer = 'ArterialFeedersBuff'
    #print 'Createing feature layer for the arterial feeders buffer feature class . . .'
    arcpy.AddMessage('Createing feature layer for the arterial feeders buffer feature class . . .')
    if arcpy.Exists(arterialFeedersBuffFeatureLayer) == False:
       arcpy.MakeFeatureLayer_management(arterialBufferFC, arterialFeedersBuffFeatureLayer)
    else:
       arcpy.Delete_management(arterialFeedersBuffFeatureLayer)
       arcpy.MakeFeatureLayer_management(arterialBufferFC, arterialFeedersBuffFeatureLayer)
```

```
##    # Clear the selection to make sure nothing gets picked up that isn't wanted
##    arcpy.SelectLayerByAttribute_management(feedersSplitFeatureLayer, "CLEAR_SELECTION")
##
##    # Select only non primary arterial roads with "FEACODE" <> 1 for second buffer
##    arcpy.SelectLayerByAttribute_management(feedersSplitFeatureLayer, "NEW_SELECTION",
"ArterialFeeder <> 1")
##
   # Create buffer of only the arterial roads
   nonArterialBufferFCName = 'FeedersNonArterial_{0}FtBuff'.format(inputBufferDistance)
   nonAarterialBufferFC = '{0}\\{1}'.format(outputLocationFCs, nonArterialBufferFCName)
##
##    print 'Creating non-arterial feeder buffer . . .'
##    arcpy.Buffer_analysis(feedersSplitFeatureLayer, nonAarterialBufferFC, inputBufferDistance + ' Feet')
##
   # Create non-arterial feeder buffer feature layer
   nonArterialFeedersBuffFeatureLayer = 'NonArterialFeedersBuff'
   #print 'Createing feature layer for the non-arterial feeders buffer feature class . . .'
   arcpy.AddMessage('Createing feature layer for the non-arterial feeders buffer feature class . . .')
   if arcpy.Exists(nonArterialFeedersBuffFeatureLayer) == False:
      arcpy.MakeFeatureLayer_management(nonAarterialBufferFC, nonArterialFeedersBuffFeatureLayer)
   else:
      arcpy.Delete_management(nonArterialFeedersBuffFeatureLayer)
      arcpy.MakeFeatureLayer_management(nonAarterialBufferFC, nonArterialFeedersBuffFeatureLayer)
##
##    print 'Completed intermediate tasks'

   '''
   ****************************************************************
   GET ALL THE VALID HEIGHT AREAS THAT INTERSECT BOTH BUFFERS
   ****************************************************************
   '''
   #print '\nSelecting all qualifying surface heights by buffer . . .'
   arcpy.AddMessage('\nSelecting all qualifying surface heights by buffer . . .')
   # Create feature layer of the result heights polygon
   resultSurfaceHeightFeatureLayer = 'resultSurfaceHeightFeatureLayer'
   #print 'Creating surface hieght feature layer . . .'
   arcpy.AddMessage('Creating surface hieght feature layer . . .')
   if arcpy.Exists(resultSurfaceHeightFeatureLayer) == False:
      arcpy.MakeFeatureLayer_management(resultSurfaceHeightPoly, resultSurfaceHeightFeatureLayer)

   totalWhereClause = "gridcode >= 30500000 and gridcode <= 60500000"
   #print 'Selecting only surface heights within total range for arterial and non-arterial feeders . . .'
   arcpy.AddMessage('Selecting only surface heights within total range for arterial and non-arterial
feeders . . .')
   arcpy.SelectLayerByAttribute_management(resultSurfaceHeightFeatureLayer, "NEW_SELECTION",
totalWhereClause)

   #print 'Selecting valid heights for arterial feeder locations . . .'
```

```
   arcpy.AddMessage('Selecting valid heights for arterial feeder locations . . .')
   arcpy.SelectLayerByLocation_management(resultSurfaceHeightFeatureLayer, "INTERSECT",
arterialFeedersBuffFeatureLayer, "", "SUBSET_SELECTION")
   #print 'Completed selection'
   arcpy.AddMessage('Completed selection')
##
   # Export this to a feature class
   arterialFeederSurfaceHeightFCName = 'ArterialFeederSurfaceHeight'
   arterialFeederSurfaceHeightFC = '{0}\\{1}'.format(outputLocationFCs,
arterialFeederSurfaceHeightFCName)

   arterialWhereClause = "gridcode >= 35500000 and gridcode <= 60500000"
   #print 'Exporting selected arterial surface height cells . . . '
   arcpy.AddMessage('Exporting selected arterial surface height cells . . . ')
   arcpy.FeatureClassToFeatureClass_conversion(resultSurfaceHeightFeatureLayer, outputLocationFCs,
arterialFeederSurfaceHeightFCName, arterialWhereClause)
   #print 'Arterial canopy location exported to feature class'
   arcpy.AddMessage('Arterial canopy location exported to feature class')

   # Clear the selection to do new one for non-arterial feeders
   #print 'Clearing selection . . .'
   arcpy.AddMessage('Clearing selection . . .')
   arcpy.SelectLayerByAttribute_management(resultSurfaceHeightFeatureLayer, "CLEAR_SELECTION")

   # Make initial selection again
   #print 'Selecting only surface heights within total range for arterial and non-arterial feeders . . .'
   arcpy.AddMessage('Selecting only surface heights within total range for arterial and non-arterial
feeders . . .')
   arcpy.SelectLayerByAttribute_management(resultSurfaceHeightFeatureLayer, "NEW_SELECTION",
totalWhereClause)

   #print 'Selecting valid heights for non arterial feeder locations . . .'
   arcpy.AddMessage('Selecting valid heights for non arterial feeder locations . . .')
   arcpy.SelectLayerByLocation_management(resultSurfaceHeightFeatureLayer, "INTERSECT",
nonArterialFeedersBuffFeatureLayer, "", "SUBSET_SELECTION")
   #print 'Completed selection'
   arcpy.AddMessage('Completed selection')

   # Export this to a feature class
   nonArterialFeederSurfaceHeightFCName = 'NonArterialFeederSurfaceHeight'
   nonArterialFeederSurfaceHeightFC = '{0}\\{1}'.format(outputLocationFCs,
nonArterialFeederSurfaceHeightFCName)

   nonArterialWhereClause = "gridcode >= 30500000 and gridcode <= 55500000"
   #print 'Exporting selected non-arterial surface height canopy  . . . '
   arcpy.AddMessage('Exporting selected non-arterial surface height canopy  . . . ')
   arcpy.FeatureClassToFeatureClass_conversion(resultSurfaceHeightFeatureLayer, outputLocationFCs,
nonArterialFeederSurfaceHeightFCName, nonArterialWhereClause)
```

```
    #print 'Non-arterial canopy location exported to feature class'
    arcpy.AddMessage('Non-arterial canopy location exported to feature class')

    # Append the non-arterial results to the arterial results to get one final feature class
    #print 'Appending arterial and non-arterial qualified surface height feature classes . . . '
    arcpy.AddMessage('Appending arterial and non-arterial qualified surface height feature classes . . . ')
    arcpy.Append_management(nonArterialFeederSurfaceHeightFC, arterialFeederSurfaceHeightFC,
"TEST")

    # Delete any duplicates that may result from overlapping bufffers
    #print 'Deleting any duplicates from final feature class . . .'
    arcpy.AddMessage('Deleting any duplicates from final feature class . . .')
    arcpy.DeleteIdentical_management(arterialFeederSurfaceHeightFC, "Shape")

    # Add fied to calculate the correct height to 6 decimal places
    #print 'Adding final height field . . .'
    arcpy.AddMessage('Adding final height field . . .')

    # If HeightFt field already exists don't create a new one
    fieldExists = False
    resultFCFieldList = arcpy.ListFields(arterialFeederSurfaceHeightFC)
    for field in resultFCFieldList:
        if field.name == 'HeightFt':
            fieldExists = True
        else:
            fieldExists = False

    if fieldExists == False:
        arcpy.AddMessage("\tHeightFt field didn't not exists creating field . . .")
        arcpy.AddField_management(arterialFeederSurfaceHeightFC, "HeightFt", "DOUBLE")
    else:
        arcpy.AddMessage("\tHeightFt field already exist")

    # Calculate correct height value
    #print 'Calculating correct height to 6 decimal places . . .'
    arcpy.AddMessage('Calculating correct height to 6 decimal places . . .')
    expression = "!gridcode! / 1000000.00 "
    arcpy.CalculateField_management(arterialFeederSurfaceHeightFC, "HeightFt", expression, "PYTHON")
    print 'Qualified Surface Height Results Layer is complete'
    arcpy.AddMessage('Qualified Surface Height Results Layer is complete')

    #print '\nFINISHED'
    arcpy.AddMessage('\nFINISHED')

except Exception as e:
    #print '{0}'.format(e)
    arcpy.AddMessage('{0}'.format(e))
```

SCL Calculate Canopy Net Impact Pthon Script Tool code:

** This code needs to be updated. After the buffer is created and before secion three the buffer needs to be dissolved to be one continuous buffer for the entire area.

```
'''
   Name:         SCLCalculateCanopyNetImpact.py
   Create Date:   07/26/2014
   Created by:    Ben Hillam
   Organization:  UW PMPGIS Capstone Project - Sponser Seattle City Light
   Description:   This script was written to go through the Seattle City Light (SCL)
                  overhead feeders calculate total area of canopy possibly impacted by
                  the overhead lines
'''
# Import Modules
import arcpy
from arcpy import env

# Global Variables
canopyFC = arcpy.GetParameterAsText(0)#r'C:\Grad_School\Geog_569\Seattle City Light
Project\Data\Lidar\testWorkspace\ValidCanopyResults.gdb\ArterialFeederSurfaceHeight'
feederFC = arcpy.GetParameterAsText(1)#r'C:\Grad_School\Geog_569\Seattle City Light
Project\Data\SCLCapstoneProject.gdb\OH_Feeders'
outputLocation = arcpy.GetParameterAsText(3)#r'C:\Grad_School\Geog_569\Seattle City Light
Project\Data\Lidar\testWorkspace\ResultsCanopyPerFeeder.gdb'

bufferDistance = arcpy.GetParameterAsText(2)#'10'
nearCalculated = False
areaField = False
areaPerFeederField = False
totalCanopyField = False

env.workspace = outputLocation

# Methods
try:
   '''
   SECTION 1: Creating the buffer for the feeders.  This is used to calculate the area and takes
   the input parameter of the bufferDistance.  Here variation modeling could be done to complete this.
   '''
   # Create feeders buffer
   feedersBuffFCName = 'OH_Feeder{0}FtBuffer'.format(bufferDistance)
   feedersBuffFC = '{0}\\{1}'.format(outputLocation, feedersBuffFCName)
   #print '\nStarted Process...'
   arcpy.AddMessage('\nStarted Process...')
   #print 'Creating feeder buffer . . .'
   arcpy.AddMessage('Creating feeder buffer . . .')
```

```
    # If the buffered feature class doesn't already exist create it
    if arcpy.Exists(feedersBuffFC) == False:
        arcpy.Buffer_analysis(feederFC, feedersBuffFC, '{0} Feet'.format(bufferDistance))
    else:
        #print '\tBuffer Already Existed'
        arcpy.AddMessage('\tBuffer Already Existed')

    '''
    SECTION 3:  Get the selected canopy first by what road it is associated with, then whether it falls
within the
    feeders buffer
    '''
    #print 'Begginging to go through calculate canopy per feeder'
    arcpy.AddMessage('Begginging to go through calculate canopy per feeder . . .')

    # Clip the canopy featureclass to the buffered feeder feature class
    canopyClipFCName = 'FeederCanopy{0}FtBuff'.format(bufferDistance)
    canopyClipFC = r'{0}\\{1}'.format(outputLocation, canopyClipFCName)
    if arcpy.Exists(canopyClipFC) == False:
        #print 'Clipping Canopy . . .'
        arcpy.AddMessage('Clipping Canopy . . .')
        arcpy.Clip_analysis(canopyFC, feedersBuffFC, canopyClipFC)
    else:
        #print "Canopy clip feature class all ready exists"
        arcpy.AddMessage("Canopy clip feature class all ready exists")

    # Evaluate if near distance already calculated
    fields = arcpy.ListFields(canopyClipFC)
    for field in fields:
        if field.name == 'NEAR_FID':
            nearCalculated = True
        if field.name == 'AreaSqFt':
            areaField = True
        if field.name == 'aPerFeeder':
            areaPerFeederField == True


    # Use near analysis to calculate which feeder the canopy will go to
    if nearCalculated == False:
        #print 'Calculating near distance for clipped canopy . . .'
        arcpy.AddMessage('Calculating near distance for clipped canopy . . .')
        arcpy.Near_analysis(canopyClipFC, feederFC)
    else:
        #print 'Near distance already calculated'
        arcpy.AddMessage('Near distance already calculated')

    if areaField == False:
        #print 'Creating field to calculate area in square feet . . .'
```

```
    arcpy.AddMessage('Creating field to calculate area in square feet . . .')
    arcpy.AddField_management(canopyClipFC, "AreaSqFt", "DOUBLE")
  else:
    #print 'Clipped canopy feature square feet field all ready exists'
    arcpy.AddMessage('Clipped canopy feature square feet field all ready exists')

  if areaPerFeederField == False:
    #print 'Creating area per feeder field . . .'
    arcpy.AddMessage('Creating area per feeder field . . .')
    arcpy.AddField_management(canopyClipFC, "aPerFeeder", "Double")
  else:
    #print 'Area per feeder field already exists'
    arcpy.AddMessage('Area per feeder field already exists')

  # Sort the attribute table before you edit it
  canopyClipSortFCName = '{0}Sort'.format(canopyClipFCName)
  canopyClipSortFC = r'{0}\\{1}'.format(outputLocation, canopyClipSortFCName)

  if arcpy.Exists(canopyClipSortFC) == False:
    #print 'Sorting the clipped canopy layer . . .'
    arcpy.AddMessage('Sorting the clipped canopy layer . . .')
    arcpy.Sort_management(canopyClipFC, canopyClipSortFC, 'NEAR_FID')
  else:
    #print 'Sort feature class already exists . . .'
    arcpy.AddMessage('Sort feature class already exists . . .')

  #print 'Calculating number of records in canopy feeter class . . .'
  arcpy.AddMessage('Calculating number of records in canopy feeter class . . .')
  numRows = arcpy.GetCount_management(canopyClipSortFC)
  numberRowsInCanopyFC = int(numRows.getOutput(0))

  #print 'Calculating the area of canopy of clipped features & sum per feeder . . .'
  arcpy.AddMessage('Calculating the area of canopy of clipped features & sum per feeder . . .')
  feederAreaTemp = []
  feederArea = []
  feederAreaFields = ['NEAR_FID','Shape_Area']
  counter = 0
  count = 0
  areaPerFeeder = 0
  with arcpy.da.UpdateCursor(canopyClipSortFC, feederAreaFields) as cursor:
    for row in cursor:
      count+=1
      feederID = row[0]
      feederAreaSqFt = row[1]
      if count == numberRowsInCanopyFC:
        areaPerFeeder += feederAreaSqFt
        feederArea.append([feederID, areaPerFeeder])
      else:
```

```
            if len(feederAreaTemp) == 0:
               feederAreaTemp.append([feederID, feederAreaSqFt])
               areaPerFeeder+=feederAreaSqFt
            elif (len(feederAreaTemp) - 1 == 0) & (feederID == feederAreaTemp[counter - 1][0]):
               feederAreaTemp.append([feederID, feederAreaSqFt])
               areaPerFeeder+=feederAreaSqFt
            elif feederID == feederAreaTemp[counter-1][0]:
               feederAreaTemp.append([feederID, feederAreaSqFt])
               areaPerFeeder+=feederAreaSqFt
            else:
               feederArea.append([feederAreaTemp[counter-1][0], areaPerFeeder])
               areaPerFeeder = feederAreaSqFt
               feederAreaTemp = []
               feederAreaTemp.append([feederID, feederAreaSqFt])
               counter=0
         counter+=1

   # Have to have a canopy clip layer as well
   canopyClipFeatureLayer = 'CanopyClipFeatureLayer'
   if arcpy.Exists(canopyClipFeatureLayer) == False:
      #print 'Creating canopy clipped feature layer .  . .'
      arcpy.AddMessage('Creating canopy clipped feature layer .  . .')
      arcpy.MakeFeatureLayer_management(canopyClipFC, canopyClipFeatureLayer)
   else:
      arcpy.Delete_management(canopyClipFeatureLayer)
      arcpy.MakeFeatureLayer_management(canopyClipFC, canopyClipFeatureLayer)
      #print 'Canopy clip feature layer already exists. Deleted and created new one.'
      arcpy.AddMessage('Canopy clip feature layer already exists. Deleted and created new one.')

   # Populate the canopyClipFeature with its resulting total canopy area
   #print 'Calculating total area per feeder . . .'
   arcpy.AddMessage('Calculating total area per feeder . . .')
   for area in feederArea:
      fid = area[0]
      whereClause = 'NEAR_FID = {0}'.format(fid)
      arcpy.SelectLayerByAttribute_management(canopyClipFeatureLayer, "NEW_SELECTION",
whereClause)
      arcpy.CalculateField_management(canopyClipFeatureLayer, "aPerFeeder", area[1], "PYTHON")

   # Create a new feeders feature class to store the area results in
   resultCanopyFeederFCName = 'OH_Feeder_CanopyResults'
   resultCanopyFeederFC = '{0}\\{1}'.format(outputLocation, resultCanopyFeederFCName)

   if arcpy.Exists(resultCanopyFeederFC) == False:
      #print 'Creating result canopy feeder feature class . . .'
      arcpy.AddMessage('Creating result canopy feeder feature class . . .')
      arcpy.CopyFeatures_management(feederFC, resultCanopyFeederFC)
      #print 'Creating result square feet field . . .'
```

```
      arcpy.AddMessage('Creating result square feet field . . .')
      arcpy.AddField_management(resultCanopyFeederFC, "ResultCanopySQFt", "DOUBLE")
   else:
      #print 'Result canopy feature class already exists'
      arcpy.AddMessage('Result canopy feature class already exists')

   # Dissolve the canopy clip feature layer to use to join back to OH_Feeder_CanopyResult FC
   dissolveCanopyFCName = 'FeederCanopyResults_Dissolve'
   dissolveCanopyFC = '{0}\\{1}'.format(outputLocation, dissolveCanopyFCName)

   # The dissovle tool requires a feature class, so get the feature class from the canopy clip feature layer
   #print 'Getting canopy clip feature class from canopy clip feature layer . . .'
##    arcpy.AddMessage('Getting canopy clip feature class from canopy clip feature layer . . .')
##    desc = arcpy.Describe(canopyClipFeatureLayer)
##    arcpy.AddMessage('described canopyClipFeatureLayer')
##    canopyClipFC = desc.catalogPath
##    arcpy.AddMessage ('got path of canopyClipFeature Layer feature class')
   arcpy.AddMessage('This is where it failed earlier')

   if arcpy.Exists(dissolveCanopyFC) == False:
      #print 'Disolving canopy clip feature class . . .'
      arcpy.AddMessage('Disolving canopy clip feature class . . .')
      arcpy.Dissolve_management(canopyClipFC, dissolveCanopyFC, ["NEAR_FID", "aPerFeeder"])
   else:
      #print 'Dissolved canopy clip feature class aleady exists'
      arcpy.AddMessage('Dissolved canopy clip feature class aleady exists')

   # Join tables and calculate
   #print 'Joining table and calcuating area per feeder . . .'
   arcpy.AddMessage('Joining table and calcuating area per feeder . . .')
   arcpy.JoinField_management(resultCanopyFeederFC, "OBJECTID", dissolveCanopyFC, "NEAR_FID",
"aPerFeeder")

   # Calculate area field in result Canopy feature class
   #print 'Calculaing the area field in the result feature class . . .'
   arcpy.AddMessage('Calculaing the area field in the result feature class . . .')
   arcpy.CalculateField_management(resultCanopyFeederFC, "ResultCanopySQFt", "!aPerFeeder!",
"PYTHON")

   # Delete the joined field from the result
   #print 'Removing teomporary joined field from the result . . .'
   arcpy.AddMessage('Removing teomporary joined field from the result . . .')
   arcpy.DeleteField_management(resultCanopyFeederFC, "aPerFeeder")

##    resultCanopyFeederFC = r'C:\Grad_School\Geog_569\Seattle City Light
Project\Data\Lidar\resultHeight\ResultSurface.gdb\OH_Feeder_CanopyResults'
   # Calculate the final total canopy area per feeder
   resultFields = arcpy.ListFields(resultCanopyFeederFC)
```

```
for field in resultFields:
    if field.name == 'TotalCanopySqFt':
        totalCanopyField = True

if totalCanopyField == False:
    #print 'Creating total canopy field in result feature class . . .'
    arcpy.AddMessage('Creating total canopy field in result feature class . . .')
    arcpy.AddField_management(resultCanopyFeederFC, 'TotalCanopySqFt', 'DOUBLE')
else:
    #print 'Total canopy field already existed'
    arcpy.AddMessage('Total canopy field already existed')

# Sort the attribute table before you edit it
resultSortFCName = '{0}Sort'.format('OH_Feeder_CanopyResults')
resultSortFC = r'{0}\\{1}'.format(outputLocation, resultSortFCName)

if arcpy.Exists(resultSortFC) == False:
    #print 'Sording result layer . . .'
    arcpy.AddMessage('Sording result layer . . .')
    arcpy.Sort_management(resultCanopyFeederFC, resultSortFC, 'FEEDERID')
else:
    #print 'Sorted results feature class already existed'
    arcpy.AddMessage('Sorted results feature class already existed')

#print 'Calculating the number of records in the result feature class . . .'
arcpy.AddMessage('Calculating the number of records in the result feature class . . .')
numRecs = arcpy.GetCount_management(resultSortFC)
numRecsSort = int(numRecs.getOutput(0))

# Grouping the total result canopy area for each feeder
#print 'Grouping the total canopy per individual feeder . . .'
arcpy.AddMessage('Calculating the total canopy per individual feeder . . .')
feederTotalTemp = []
feederTotal = []
feederTotalFields = ['FEEDERID','ResultCanopySQFt']
fCount = 0
fCounter = 0
totalAreaPerFeeder = 0
with arcpy.da.SearchCursor(resultSortFC, feederTotalFields) as cursor:
    for row in cursor:
        fCount+=1
        feederID = row[0]
        feederAreaSqFt = row[1]
        if feederAreaSqFt is None:
            feederAreaSqFt = 0
        if fCount == numRecsSort:
            totalAreaPerFeeder += feederAreaSqFt
            feederTotal.append([feederID, areaPerFeeder])
```

```
            else:
                if len(feederTotalTemp) == 0:
                    feederTotalTemp.append([feederID, feederAreaSqFt])
                    totalAreaPerFeeder+=feederAreaSqFt
                elif len(feederTotalTemp) - 1 == 0:
                    feederTotalTemp.append([feederID, feederAreaSqFt])
                    totalAreaPerFeeder+=feederAreaSqFt
                elif feederID == feederTotalTemp[fCounter-1][0]:
                    feederTotalTemp.append([feederID, feederAreaSqFt])
                    totalAreaPerFeeder+=feederAreaSqFt
                else:
                    feederTotal.append([feederTotalTemp[len(feederTotalTemp)-1][0], totalAreaPerFeeder])
                    totalAreaPerFeeder = feederAreaSqFt
                    feederTotalTemp = []
                    feederTotalTemp.append([feederID, feederAreaSqFt])
                    fCounter=0
            fCounter+=1

    #print 'Calculating the total canopy for each feeder . . .'
    arcpy.AddMessage('Calculating the total canopy for each feeder . . .')
    for feeder in feederTotal:
        with arcpy.da.UpdateCursor(resultCanopyFeederFC, ["FEEDERID", "TotalCanopySqFt"]) as cursor2:
            for row in cursor2:
                if feeder[0] == row[0]:
                    row[1] = feeder[1]
                    cursor2.updateRow(row)

    # Dissolve the canopy clip feature layer to use to join back to OH_Feeder_CanopyResult FC
    dissolveResultFCName = 'TotalFeederCanopyResult'
    dissolveResultFC = '{0}\\{1}'.format(outputLocation, dissolveResultFCName)

    if arcpy.Exists(dissolveResultFC) == False:
        #print 'Disolving Canopy Resolve Feature Layer . . .'
        arcpy.AddMessage('Disolving Canopy Resolve Feature Layer . . .')
        arcpy.Dissolve_management(resultCanopyFeederFC, dissolveResultFC, ["FEEDERID",
"TotalCanopySqFt"])
    else:
        #print 'Dissolved result feature class already exists'
        arcpy.AddMessage('Dissolved result feature class already exists')

    #print "\nFINISHED"
    arcpy.AddMessage("\nFINISHED")

except Exception as e:
    print '{0}'.format(e)
    arcpy.AddMessage('{0}'.format(e))
```

Feature Dataset	Feature Class	Definition of Feature Class	Data Source	Spatial Reference	Data Model
BaseLayers					
	Building	Building locations throughout the SCL service area from 2009	Seattle City Light (SCL)	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	City	City boundaries in King County	King County GIS Portal	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	KingCounty	King County boundary	King County GIS Portal	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	SCLServiceAreaGeneral	Outer boundary of SCL service area	Derived (digitized) by analyst	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	ServiceAreaDetail	Detailed boundary of SCL service area (inner boundary included)	Derived (digitized) by analyst	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
CanopyFeederLayers					
	OH_Feeders	All the over head feeder lines in SCL service area	Seattle City Light (SCL)	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polyline
	SCLPercentCanopy11	Canopy density throughout SCL service area from 2011	Derived from United States Forest Service (USFS)	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	StreetNetwork	The transportation network (streets) for SCL service area	Seattle City Light (SCL)	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polyline
InequityIndexMCE_Inputs					
	ManufacturingCenter	Location of manufacturing centers	King County GIS Portal	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	ManufacturingIndustrialCenter	Manufacturing and Industrial areas	King County GIS Portal	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	Parks	Public parks in the greater Seattle area	King County GIS Portal	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	TreesPlanted	Samples of trees planted throughout region	Seattle Department of Transportation (SDOT)	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Point
	TreesRemoval	Total trees removed at specific location and time	Seattle City Light (SCL)	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Point
	WasteWaterTreatmentFacility	Waste water facility locations	King County GIS Portal	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Point
	Wetlands	Wetland areas	King County GIS Portal	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
InequityIndexMCE_Intermediate					
	FacilitySCL	Facilities within tracts	Derived from MCE workflow	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	Greenspace	Greenspace within tracts	Derived from MCE workflow	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	GreenspaceIntersect	Intersect of all areas for greenspace	Derived from MCE workflow	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	HospitalsSCL	Hospital locations in the greater Seattle area	Derived from MCE workflow	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Point
	IndustrialCenters	Industrial centers within selected tracts	Derived from MCE workflow	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	SCLTractsCanopySptJn	Tracts joined with the percent canopy layer	Derived from MCE workflow	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	TreesPlantedIntersect	Trees planted joined with tracts	Derived from MCE workflow	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	TreesRemovedIntersect	Trees removed joined with tracts	Derived from MCE workflow	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
InequityIndexMCE_Result					
	InequityIndexResult	Result layer with all scores	Derived from MCE workflow	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
SCLCalculateNetImpact_Results					
	FeederCanopy10FtBuff	Qualifying canopy within 10 ft. buffer of feeder line	Derived from python script tool	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	FeederCanopy10FtBuffSort	FeederCanopy10FtBuff sorted	Derived from python script tool	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	FeederCanopyResults_Dissolve	Qualifying canopy within 10 ft. buffer of feeder line dissolved by feeder line ID	Derived from python script tool	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	OH_Feeder_CanopyResults	Area (sq. ft.) of canopy within input clearance distance of each feeder multi-line part	Derived from python script tool	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polyline
	OH_Feeder_CanopyResultsSort	OH_Feeder_CanopyResults sorted	Derived from python script tool	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polyline
	OH_Feeder10FtBuffer	Feeder lines buffered by user input distance	Derived from python script tool	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	TotalFeederCanopyResult	Final result of total area (sq. ft.) of canopy within input clearance distance per feeder line	Derived from python script tool	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polyline
SCLCanopyPercent_Results					
	OH_FeederPercentCanopy	Result area (sq. ft.) of canopy possibly within clearance distance based on percent canopy layer	Derived by analyst	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polyline
	SCLPercentCanopy10FtBuff	Selected grid cells from SCLAreaPercentCanopy raster within clearance distance	Derived by analyst	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
SCLGetSelectedCanopyCover_Results					
	ArterialFeederSurfaceHeight	Final qualifying canopy area layer (NonArterialFeederSurfaceHeight appended to this layer)	Derived from python script tool	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	FeedersArterial_10FtBuff	Feeders associated with primary arterial streets buffered by input clearance distance	Derived from python script tool	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	FeedersNonArterial_10FtBuff	Feeders associated with all other types of streets buffered by input clearance distance	Derived from python script tool	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	FeedersSplit	Feeder lines split at vertices to associate with streets	Derived from python script tool	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polyline
	NonArterialFeederSurfaceHeig	Qualifying canopy area layer from non-primary arterial streets	Derived from python script tool	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
	ResultSurfaceHeight	Feature class resulting from subtracting the BaseHeight from the TopSurfaceHeight raster dataset	Derived from python script tool	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Polygon
Tables					
	Income10	Income by census tract from 2010		NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Table
	RacePopulationCensusTract10	Race and population by census tract from 2010		NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Table
Raster Datasets					
	BaseHeight	Base 6ft. Resolution raster from LIDAR 2000	Puget Sound LIDAR Consortium (PSLC)	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Raster
	SCLAreaPercentCanopy	Canopy density that intersects SCL feeders 2011	United States Forest Service (USFS)	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Raster
	SCLPercentCanopy	Canopy density for entire SCL service area 2011	United States Forest Service (USFS)	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Raster
	TopSurfaceHeight	Top surface (first return) raster from LIDAR 2000	Puget Sound LIDAR Consortium (PSLC)	NAD 1983 HARN StatePlane Washington North FIPS 4601 Fe	Raster

Figure 1 SES Map