Sewage Conveyance Systems and Impervious Groundcover in Philadelphia, PA

Sarah Jaraha

November 2018

Introduction

This analysis uses data from Philadelphia Water Department to visualize the relationship between Combined Sewer Overflow (CSO), Multiple Separate Storm Sewer Systems (MS4), and impervious groundcover in Philadelphia. ArcGIS, SQL Server, and Sublime were used to conduct the analysis.

Background

CSO is a sewage conveyance system in which stormwater runoff, domestic sewage, and industrial wastewater flow into a single pipe (Appendix A). Normally, the water in CSO pipes is safely transported to a sewage treatment plant. If there is a heavy rain or snow event, the amount of water in a CSO system can exceed the capacity of the sewage treatment plant. In these cases, CSO systems discharge the excess contents to local waterbodies, along with any pollutants they contain.

MS4 is a sewage conveyance system in which stormwater and sewage are collected separately (Appendix A). Sewage is collected in a sanitary sewer pipe and transported to a local treatment plant. Stormwater is collected in a stormwater sewer pipe, then discharged into a local waterbody. To manage the release of pollutants to local waterbodies, some MS4 users must obtain a permit before discharging.

Groundcover has a significant impact on the amount of stormwater that reaches a CSO or MS4 system. Impervious cover, like concrete and rooftops, diverts stormwater directly to the CSO or MS4 system. Pervious cover, like grass, absorbs stormwater and reduces the amount of stormwater that reaches CSO or MS4 systems (Appendix A).

Methods

The analysis was conducted using the following initial datasets from the Philadelphia Water Department: City_Boundary, CSO_Area, PARCELS, and Impervious_2015. The datasets come in the form of shapefiles. City_Boundary shows the Philadelphia city boundary. CSO_Area shows the portion of Philadelphia utilizing the CSO system. PARCELS shows the individual properties, also called parcels, in Philadelphia. Impervious_2015 categorizes the groundcover in Philadelphia (natural surface, parking lot, etc.). Table 1 provides a summary of the initial datasets.

Table 1: Initial Datasets

Layer Name	Description
City_Boundary	Philadelphia's city boundary
CSO_Area	Portion of Philadelphia utilizing the CSO system
PARCELS	Philadelphia parcels
Impervious_2015	Classification of groundcover in Philadelphia

The remaining datasets necessary to carry out the analysis can be drawn from the initial datasets using ArcPy and SQL Server (Appendix B). Datasets will be referred to as layers for the remainder of the report. Parcels falling between the CSO and MS4 areas, were placed into category containing the greater half of the parcel. The following layers were created: MS4_Area, CSO_Parcels_sorted, MS4_Parcels_sorted. The layers are described in Table 2.

Table 2: Layers Created Using City_Boundary, CSO_Area, and PARCELS

Layer Name	Description
MS4_Area	Portion of Philadelphia utilizing the MS4 system
CSO_Parcels_sorted	Parcels falling within the CSO area
MS4_Parcels_sorted	Parcels falling within the MS4 area

Once the CSO and MS4 parcels were sorted in the CSO_Parcels_sorted and MS4_Parcels_sorted layers, the Impervious_2015 layer was introduced to the analysis. First, a subset of Impervious_2015 was created which includes only impervious groundcover. This layer is called Impervious_2015_IA. Next, Impervious_2015_IA, CSO_Parcels_sorted, and MS4_parcels_sorted were manipulated in ArcPy and SQL Server (Appendix B) to create Impervious_2015_IA_intersect_CSOparcels and Impervious_2015_IA_intersect_MS4parcels. Table 3 provides a summary of the layers.

Table 3: Layers Created Using Impervious_2015, CSO_Parcels_sorted, MS4_parcels_sorted

Layer Name	Description
Impervious_2015_IA	Impervious groundcover in Philadelphia
Impervious_2015_IA_intersect_CSOparcels	Impervious groundcover within CSO parcels
Impervious_2015_IA_intersect_MS4parcels	Impervious groundcover within MS4 parcels

Results

The CSO_Parcels_sorted layer is summarized by billing class in Table 4. Dashes represent null values.

Table 4: CSO Parcels by Billing Class

Billing Class	Number of CSO Parcels	Percentage of Total CSO Parcels (%)
Apt	7,127	2
Cemetery	32	0
Condo	1,488	0
Exempt City	5,725	1
Exempt Community Garden	109	0
Exempt Notify	-	-
Exempt Other	42	0
Exempt Sideyard	1,333	0
Exempt Street	134	0
Non-Residential	66,654	15
Row	314,746	71
Single	7,089	2
Twin	35,814	8
TOTAL	440,293	100

Row is the most common billing class in the CSO area. Table 4 does not include CSO parcels that have not been assigned a billing class. Percentages are calculated from parcels with an assigned billing class.

The MS4_Parcels_sorted layer is summarized by billing class in Table 5.

Table 5: MS4 Parcels by Billing Class

Billing Class	Number of MS4 Parcels	Percentage of Total MS4 Parcels (%)
Apt	7,242	7
Cemetery	17	0
Condo	511	0
Exempt City	309	0
Exempt Community Garden	1	0
Exempt Notify	3	0
Exempt Other	11	0
Exempt Sideyard	390	0
Exempt Street	6	0
Non-Residential	7,144	7
Row	34,174	32
Single	21,129	20
Twin	34,930	33
TOTAL	105, 867	100

Twin is the most common billing class in the MS4 area, followed closely by row. Table 5 does not include MS4 parcels that have not been assigned a billing class. Percentages are calculated from parcels with an assigned billing class.

The Impervious_2015_IA_intersect_CSOparcels layer is summarized by billing class in Table 6. Dashes represent null values.

Table 6: Impervious Area (IA) Occupied by Each Billing Class in CSO Parcels

Billing Class	IA within CSO Parcels (ft^2)	Equivalent Number of Lincoln Financial Fields	Percentage of Total CSO IA (%)
Apt	10,591,436	16	1
Cemetery	696,222	1	0
Condo	11,636,348	17	1
Exempt City	3,720,602	5	0
Exempt Community Garden	90,068	0	0
Exempt Notify	-	-	-
Exempt Other	326,462	0	0
Exempt Sideyard	558,181	0	0
Exempt Street	655,796	1	0
Non-Residential	404,626,595	620	50
Row	300,575,411	461	37
Single	18,371,082	28	2
Twin	53,148,025	81	7
TOTAL	804,996,228	1230	100

The “Equivalent Number of Lincoln Financial Fields” column shows the number of Lincoln Financial Fields that is equivalent to the IA in each billing class. The non-residential billing class occupies the most IA. Table 6 does not include IA within CSO parcels that have not been assigned a billing class. Percentages are calculated from parcels with an assigned billing class.

The Impervious_2015_IA_intersect_MS4parcels layer is summarized by billing class in Table 7.

Table 7: Impervious Area (IA) Occupied by Each Billing Class in MS4 Parcels

Billing Class	IA within MS4 Parcels (ft^2)	Equivalent Number of Lincoln Financial Fields	Percentage of Total MS4 IA (%)
Apt	12,073,757	18	2
Cemetery	1,304,492	2	0
Condo	10,603,595	16	2
Exempt City	3,613,357	5	1
Exempt Community Garden	3,312	0	0
Exempt Notify	3,733	0	0
Exempt Other	131,834	0	0
Exempt Sideyard	257,135	0	0
Exempt Street	128,145	0	0
Non-Residential	342,268,593	525	65
Row	40,194,783	61	8
Single	62,706,443	96	12
Twin	55,543,704	85	11
TOTAL	528,832,883	808	100

The non-residential billing class occupies the most IA. Table 7 does not include IA within MS4 parcels that have not been assigned a billing class. Percentages are calculated from parcels with an assigned billing class.

Tables 4, 5, 6, and 7 are summarized in Table 8, below. Parcels that have not been assigned a billing class are included in the Table 8.

Table 8: Summary of Results

Calculated Field	Sewage Conveyance System	
	CSO	MS4
Number of Parcels	440,630	105,949
IA within Parcels (ft^2)	810,124,640	532,597,555

Discussion

Urban areas have more impervious area. This is reflected by the IA in CSO and MS4 parcels. The CSO system operates mostly in central Philadelphia and has a total IA of 810,124,640 ft^2 within parcels. The MS4 system covers the outskirts of Philadelphia, including Wissahickon Valley Park, Pennypack Park, and the Northeast Philadelphia Airport. The total IA within MS4 parcels is 532,597,555 ft^2.

The number of parcels in an area increases with the IA. The CSO area contains 440,630 parcels, while the MS4 area contains 105,949 parcels.

In the CSO area, the highest percentage of parcels are in the row billing class (71% of CSO parcels) and the highest percentage of IA is in the non-residential billing class (50% of IA). In the MS4 area, the highest percentage of parcels are in the twin billing class (33% of MS4 parcels) followed by row (32% of MS4 parcels), and the highest percentage of IA is in the non-residential billing class (65% of IA).

The inversion between the number of parcels and the IA may be due to the larger size of non-residential parcels (often warehouses or office buildings). Non-residential parcels occupy 10,120 ft^2/non-residential parcel, on average. Row parcels occupy 976 ft^2/row parcel, on average. Twin parcels occupy 1,536 ft^2/twin parcel, on average.

A future analysis may be conducted to include the parcels that do not have an assigned billing class. Rainfall and snowfall data may be incorporated to provide context.

Appendix A: Maps

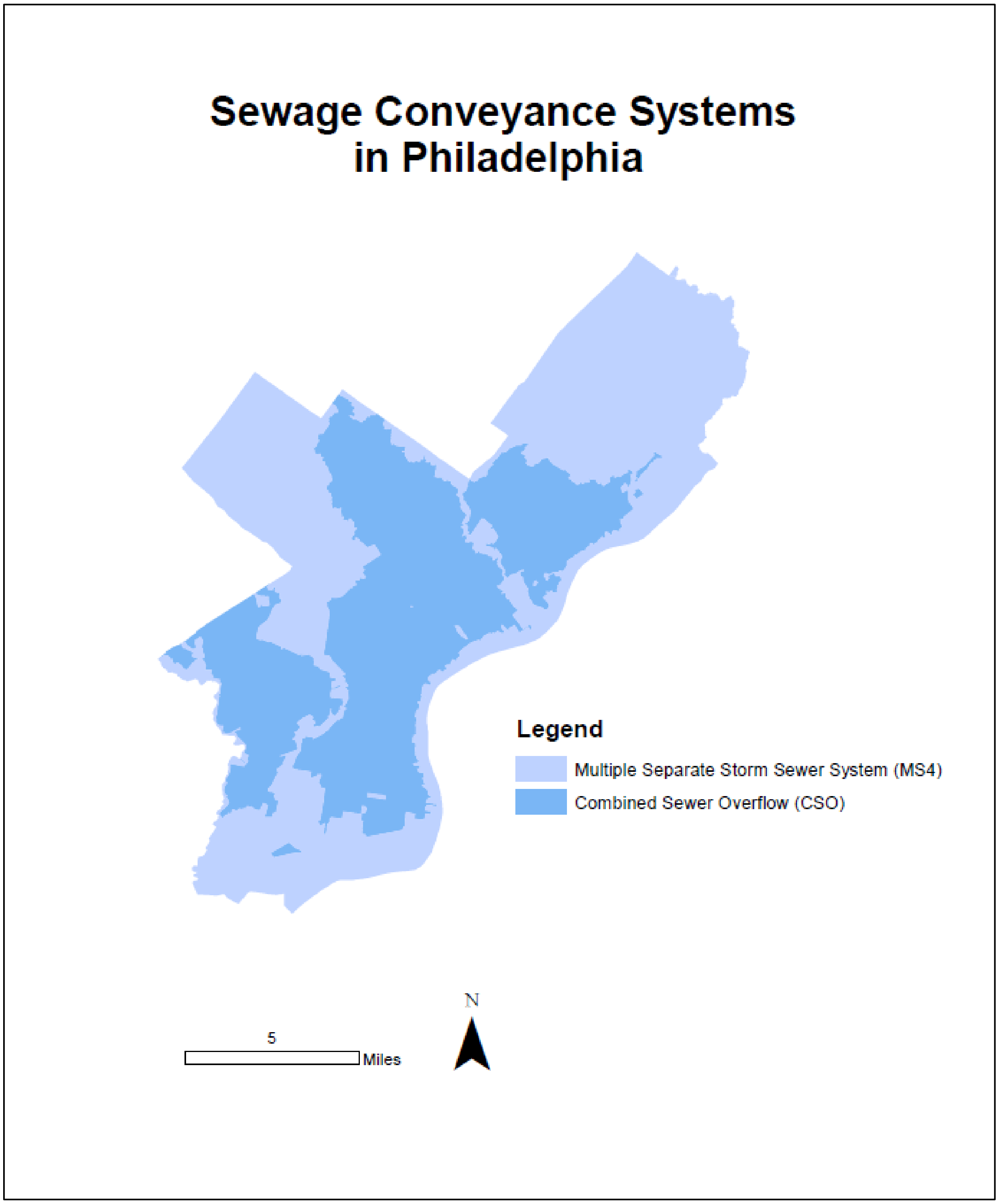
Sewage Conveyance Systems in Philadelphia
Legend
Multiple Separate Storm Sewer System (MS4)
Combined Sewer Overflow (CSO)
5
Miles
N

Figure 1: Map of CSO and MS4 sewage conveyance systems in Philadelphia

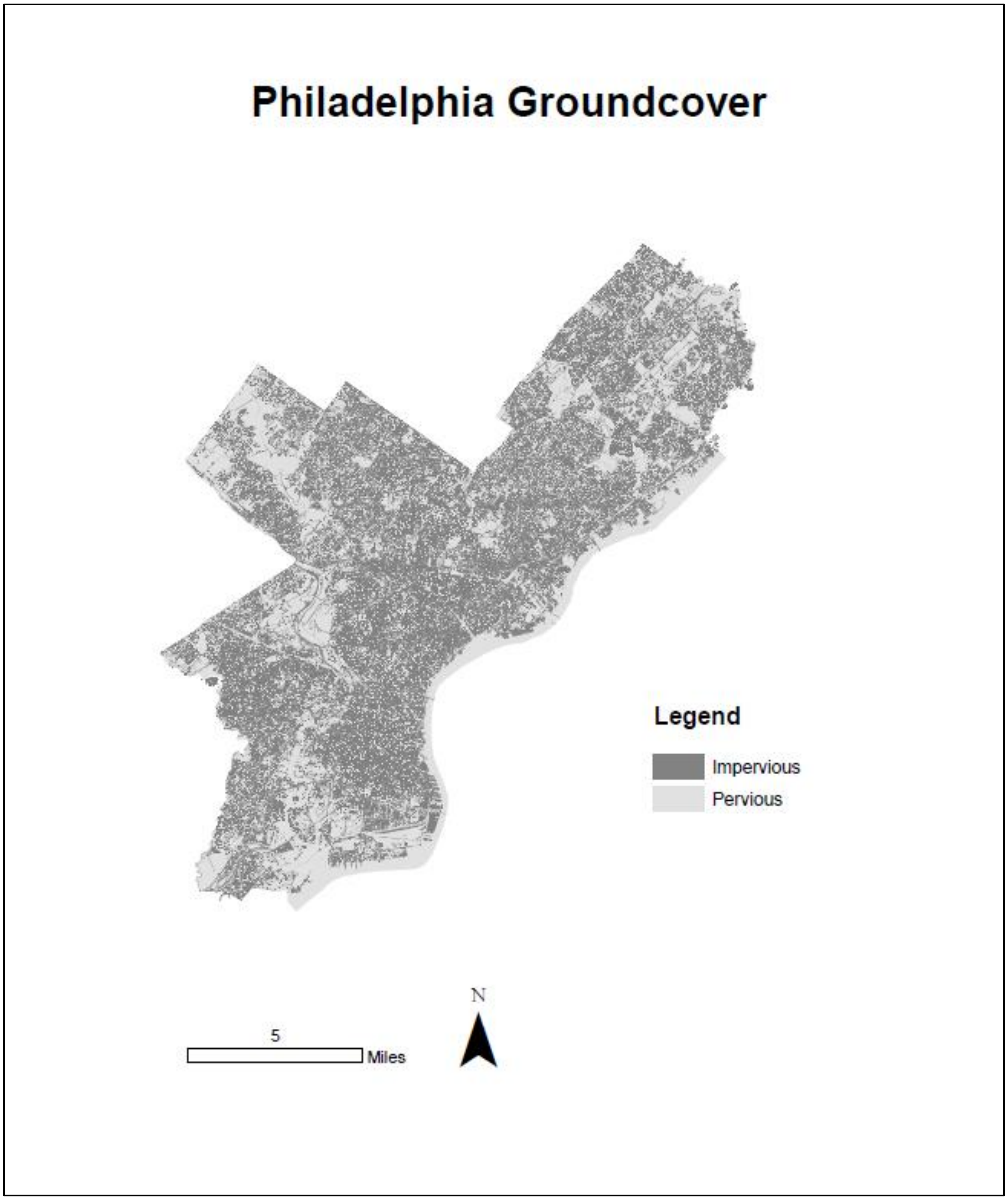

Figure 2: Map of impervious and pervious groundcover in Philadelphia

Appendix B: Code

languages: Python, SQL Server

```
# import packages

import arcpy
import pyodbc

# establish connection between ArcGIS and SQL Server

cnxn = pyodbc.connect('DRIVER={SQL Server Native Client
10.0};SERVER=PWDSTORMSQL;DATABASE=GISTEST1;UID=gisad;PWD=erv11new', autocommit=True)
##SERVER NAME HERE
cursor = cnxn.cursor()

# begin analysis

#######################################################################################
########################################################     MAKE INITIAL LAYERS    #####
#######################################################################################

#######################################################################################
######## MAKE FEATURE LAYERS: FL_City_Boundary , FL_CSO_Area

# DEF
data_in_boundary =
r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\City_Boundary'
data_in_area_cso = r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\CSO_Area'
data_in_parcels = r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\PARCELS'
data_out_boundary = 'City_Boundary_FL'
data_out_area_cso = 'CSO_Area_FL'
data_out_parcels = 'PARCELS_FL'

# DO
print 'Making Feature Layer: City_Boundary_FL'
arcpy.MakeFeatureLayer_management (data_in_boundary, data_out_boundary)

print 'Making Feature Layer: CSO_Area_FL'
arcpy.MakeFeatureLayer_management (data_in_area_cso, data_out_area_cso)

print 'Making Feature Layer: PARCELS_FL'
arcpy.MakeFeatureLayer_management(data_in_parcels, data_out_parcels)


#######################################################################################
######## CLIP: PARCELS_FL to CSO_Area_FL
######## MAKE: CSO_Parcels_arcpy

# DEF
data_out = r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\CSO_Parcels_arcpy'
```

```
# DO
print 'Executing Clip: CSO_Parcels_arcpy'
if arcpy.Exists(data_out) is True:
    arcpy.Delete_management(data_out)
    print (arcpy.GetMessages())
arcpy.Clip_analysis(data_out_parcels, data_out_area_cso, data_out)
print '\tClip Complete: CSO_Parcels_arcpy'

#######################################################################################
######## ERASE: CSO_Area_FL from City_Boundary_FL
######## MAKE: MS4_Area_arcpy

# DEF
data_out = r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\MS4_Area_arcpy'

# DO
print 'Executing Erase: MS4_Area_arcpy'
if arcpy.Exists(data_out) is True:
    arcpy.Delete_management(data_out)
    print (arcpy.GetMessages())
arcpy.Erase_analysis(data_out_boundary, data_out_area_cso, data_out)
print '\tErase Complete: MS4_Area_arcpy'

#######################################################################################
######## MAKE FEATURE LAYER: MS4_Area_arcpy_FL

# DEF
data_in_parcels_ms4 =
r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\MS4_AREA_arcpy'
data_out_parcels_ms4 = 'MS4_AREA_arcpy_FL'

# DO
print 'Making Feature Layer: MS4_AREA_arcpy_FL'
arcpy.MakeFeatureLayer_management(data_in_parcels_ms4, data_out_parcels_ms4)

#######################################################################################
######## CLIP: PARCELS_FL to MS4_Area_arcpy_FL
######## MAKE: MS4_Parcels_arcpy

# DEF
data_out = r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\MS4_Parcels_arcpy'

# DO
print 'Executing Clip: MS4_Parcels_arcpy'
if arcpy.Exists(data_out) is True:
    arcpy.Delete_management(data_out)
    print (arcpy.GetMessages())
arcpy.Clip_analysis(data_out_parcels, data_out_parcels_ms4, data_out)
print '\tClip Complete: MS4_Parcels_arcpy'

#######################################################################################
####################################################    COPY LAYERS TO GISTEST1   #####
```

```
#########################################################################################

#########################################################################################
######## COPY: MS4_Parcels_arcpy
#######  CSO_Data.gdb --> GISTEST1

# DEF
data_in = r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\MS4_Parcels_arcpy'
data_out =
r'C:\Users\sarah.jaraha\AppData\Roaming\ESRI\Desktop10.5\ArcCatalog\GISAD@PWDSTORMSQL_G
ISTEST1.sde\MS4_Parcels_arcpy'

# DO
print 'Copying: MS4_Parcels_arcpy --> GISTEST1'
if arcpy.Exists(data_out) is True:
    arcpy.Delete_management(data_out)
    print (arcpy.GetMessages())
arcpy.Copy_management(data_in, data_out)
print '\tCopy Complete: MS4_Parcels_arcpy --> GISTEST1'

#########################################################################################
######## COPY: CSO_Parcels_arcpy
#######  CSO_Data.gdb --> GISTEST1

# DEF
data_in = r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\CSO_Parcels_arcpy'
data_out =
r'C:\Users\sarah.jaraha\AppData\Roaming\ESRI\Desktop10.5\ArcCatalog\GISAD@PWDSTORMSQL_G
ISTEST1.sde\CSO_Parcels_arcpy'

# DO
print 'Copying: CSO_Parcels_arcpy --> GISTEST1'
if arcpy.Exists(data_out) is True:
    arcpy.Delete_management(data_out)
    print (arcpy.GetMessages())
arcpy.Copy_management(data_in, data_out)
print '\tCopy Complete: CSO_Parcels_arcpy --> GISTEST1'

#########################################################################################
####################################      MAKE TABLE OF SORTED BORDER PARCELS    ######
#########################################################################################

#########################################################################################
######## MAKE TABLE: CSO_MS4_sort

# SQL
query = """
exec drop_existing_table 'CSO_MS4_sort'

CREATE TABLE CSO_MS4_sort
(parcelid varchar(20),
    decision varchar(5)) ;
```

```
INSERT INTO CSO_MS4_sort
SELECT
    c.PARCELID,
    CASE
        WHEN (c.shape.STArea()/(m.shape.STArea()+ c.shape.STArea()))*100 > 50 THEN
'CSO'
        WHEN (c.shape.STArea()/(m.shape.STArea()+ c.shape.STArea()))*100 < 50 THEN
'MS4'
        ELSE '50/50'
    END
FROM CSO_PARCELS_ARCPY as c
JOIN MS4_PARCELS_ARCPY as m
ON c.parcelid = m.parcelid
; """


# DO
print 'Making Table: CSO_MS4_sort'
cursor.execute(query)
print '\tTable Complete: CSO_MS4_sort'

#########################################################################################
##########################   MAKE SORTED FEATURE CLASSES OF CSO and MS4 PARCELS   #####
#########################################################################################

#########################################################################################
######### SELECT: CSO_Parcels_arcpy_select

# DEF
data_in = r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\PARCELS'
where = 'PARCELID IN (SELECT PARCELID FROM CSO_PARCELS_ARCPY) AND PARCELID NOT IN
(SELECT parcelid FROM CSO_MS4_sort WHERE decision = \'MS4\')'
data_out =
r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\CSO_Parcels_arcpy_select'

# DO
print 'Executing Select : CSO_Parcels_arcpy_select'
if arcpy.Exists(data_out) is True:
    arcpy.Delete_management(data_out)
    print (arcpy.GetMessages())
arcpy.Select_analysis(data_in, data_out, where)
print '\tSelect Complete: CSO_Parcels_arcpy_select'

#########################################################################################
########### COPY FEATURES: CSO_Parcels_arcpy_sorted

# DEF
data_in = data_out
data_out =
r'C:\Users\sarah.jaraha\AppData\Roaming\ESRI\Desktop10.5\ArcCatalog\GISAD@PWDSTORMSQL_G
ISTEST1.sde\CSO_Parcels_arcpy_sorted'

# DO
```

```
print 'Copying Features From: CSO_Parcels_arcpy_select'
if arcpy.Exists(data_out) is True:
    arcpy.Delete_management(data_out)
    print (arcpy.GetMessages())
arcpy.CopyFeatures_management(data_in, data_out)
print '\tFeatures Copied As: CSO_Parcels_arcpy_sorted'

#######################################################################################
######### SELECT: MS4_Parcels_arcpy_select

# DEF
data_in = r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\PARCELS'
where = 'PARCELID IN (SELECT PARCELID FROM MS4_PARCELS_ARCPY) AND PARCELID NOT IN
(SELECT parcelid FROM CSO_MS4_sort WHERE decision = \'CSO\')'
data_out =
r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\MS4_Parcels_arcpy_select'

# DO
print 'Executing Select : MS4_Parcels_arcpy_select'
if arcpy.Exists(data_out) is True:
    arcpy.Delete_management(data_out)
    print (arcpy.GetMessages())
arcpy.Select_analysis(data_in, data_out, where)
print '\tSelect Complete: MS4_Parcels_arcpy_select'

#######################################################################################
########### COPY FEATURES: MS4_Parcels_arcpy_sorted

# DEF
data_in = data_out
data_out =
r'C:\Users\sarah.jaraha\AppData\Roaming\ESRI\Desktop10.5\ArcCatalog\GISAD@PWDSTORMSQL_G
ISTEST1.sde\MS4_Parcels_arcpy_sorted'

# DO
print 'Copying Features From: MS4_Parcels_arcpy_select'
if arcpy.Exists(data_out) is True:
    arcpy.Delete_management(data_out)
    print (arcpy.GetMessages())
arcpy.CopyFeatures_management(data_in, data_out)
print '\tFeatures Copied As: MS4_Parcels_arcpy_sorted'

#######################################################################################
############################    SUMMARIZE CSO and MS4 PARCELS BY BILLING CLASS    #####
#######################################################################################

#######################################################################################
######## MAKE TABLE: billing_class_summary_parcels

#SQL
query = """
--drop table
exec drop_existing_table 'billing_class_summary_parcels';
```

```
--create table
CREATE TABLE billing_class_summary_parcels
(Billing_Class nvarchar(25),
Number_of_CSO_Parcels int,
Percentage_of_Total_CSO_Parcels float,
Number_of_MS4_Parcels int,
Percentage_of_Total_MS4_Parcels float);

--insert: billing_class
INSERT INTO billing_class_summary_parcels(billing_class)
SELECT Billing_Class
FROM PARCELS
GROUP BY BILLING_CLASS;

--update: Number_of_CSO_Parcels
UPDATE billing_class_summary_parcels
SET Number_of_CSO_Parcels = CSO_count
FROM (
    SELECT billing_class, COUNT(*)as CSO_count
    FROM CSO_PARCELS_ARCPY
    WHERE PARCELID in
        (SELECT PARCELID
        FROM CSO_PARCELS_ARCPY
        EXCEPT
        SELECT parcelid
        FROM CSO_MS4_sort
        WHERE decision = 'MS4')
    GROUP BY billing_class) as CSO_tally
WHERE billing_class_summary_parcels.billing_class = CSO_tally.billing_class;

--update: CSO_percentage
UPDATE billing_class_summary_parcels
SET Percentage_of_Total_CSO_Parcels = percentage
FROM
    (SELECT Billing_Class, round((CAST(Number_of_CSO_Parcels AS float)/(SELECT
SUM(Number_of_CSO_Parcels) FROM billing_class_summary_Parcels)),2)*100 as percentage
    FROM billing_class_summary_parcels)as csop
WHERE billing_class_summary_parcels.Billing_Class = csop.Billing_Class;

--update: Number_of_MS4_Parcels
UPDATE billing_class_summary_parcels
SET Number_of_MS4_Parcels = MS4_count
FROM (
    SELECT billing_class, COUNT(*)as MS4_count
    FROM MS4_PARCELS_ARCPY
    WHERE PARCELID in
        (SELECT PARCELID
        FROM MS4_PARCELS_ARCPY
        EXCEPT
        SELECT parcelid
        FROM CSO_MS4_sort
        WHERE decision = 'CSO')
    GROUP BY billing_class) as MS4_tally
WHERE billing_class_summary_parcels.billing_class = MS4_tally.billing_class;

--update: MS4_percentage
UPDATE billing_class_summary_parcels
SET Percentage_of_Total_MS4_Parcels = percentage
FROM
    (SELECT Billing_Class, round((CAST(Number_of_MS4_Parcels AS float)/(SELECT
SUM(Number_of_MS4_Parcels) FROM billing_class_summary_Parcels)),2)*100 as percentage
    FROM billing_class_summary_parcels)as MS4p
WHERE billing_class_summary_parcels.Billing_Class = MS4p.Billing_Class;

"""

# DO
print 'Creating Table: billing_class_summary_parcels'
cursor.execute(query)
print '\tTable Complete: billing_class_summary_parcels'

#######################################################################################
##############################    MAKE FEATURE CLASS OF IMPERVIOUS AREA ONLY    #######
#######################################################################################

#######################################################################################
########## MAKE FEATURE LAYER: IMPERVIOUS_2015_IA_FL

# DEF
data_in = r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\IMPERVIOUS_2015'
where = "FCODE IN (
1001,1011,1021,1201,1211,12211811,1821,1831,1841,2201,2211,2221,2231,2241,2251,0,1000,1
010,1020,1200,1210,1220,1810,1820,1830,1840,2200,2210,2220,2230,2240,2250 )"
data_out_FL = 'IMPERVIOUS_2015_IA_FL'

# DO
print 'Making Feature Layer: IMPERVIOUS_2015_IA_FL'
arcpy.MakeFeatureLayer_management(data_in,data_out_FL, where)

#######################################################################################
########## COPY FEATURES: IMPERVIOUS_2015_IA

# DEF
data_out =
r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\IMPERVIOUS_2015_IA'

# DO
print 'Copying Features From: IMPERVIOUS_2015_IA_FL'
if arcpy.Exists(data_out) is True:
    arcpy.Delete_management(data_out)
    print (arcpy.GetMessages())
arcpy.CopyFeatures_management(data_out_FL, data_out)
print '\tFeatures Copied As: IMPERVIOUS_2015_IA'

#######################################################################################
```

```
###################################    IMPERVIOUS AREA IN CSO and MS4 PARCELS    ######
#######################################################################################

#######################################################################################
######### Intersect: IMPERVIOUS_2015_IA_intersect_CSOparcels

# DEF
intersect_1 =
r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\IMPERVIOUS_2015_IA'
intersect_2 =
r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\CSO_Parcels_arcpy_sorted'
intersection = [intersect_1, intersect_2]
data_out =
r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\IMPERVIOUS_2015_IA_intersect_C
SOparcels'

# DO
print 'Executing Intersect: IMPERVIOUS_2015_IA_intersect_CSOparcels'
if arcpy.Exists(data_out) is True:
    arcpy.Delete_management(data_out)
    print (arcpy.GetMessages())
arcpy.Intersect_analysis(intersection, data_out)
print '\tIntersect Complete: IMPERVIOUS_2015_IA_intersect_CSOparcels'

#######################################################################################
########### Intersect: IMPERVIOUS_2015_IA_intersect_MS4parcels

# DEF
intersect_1 =
r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\IMPERVIOUS_2015_IA'
intersect_2 =
r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\MS4_Parcels_arcpy_sorted'
intersection = [intersect_1, intersect_2]
data_out =
r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\IMPERVIOUS_2015_IA_intersect_M
S4parcels'

# DO
print 'Executing Intersect: IMPERVIOUS_2015_IA_intersect_MS4parcels'
if arcpy.Exists(data_out) is True:
    arcpy.Delete_management(data_out)
    print (arcpy.GetMessages())
arcpy.Intersect_analysis(intersection, data_out)
print '\tIntersect Complete: IMPERVIOUS_2015_IA_intersect_MS4parcels'


#######################################################################################
######################################################   COPY LAYERS TO GISTEST1   #####
#######################################################################################

#######################################################################################
########### COPY: IMPERVIOUS_2015_IA_intersect_CSOparcels
########### CSO_Data.gdb--> GISTEST1
```

```
# DEF
data_in =
r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\IMPERVIOUS_2015_IA_intersect_C
SOparcels'
data_out =
r'C:\Users\sarah.jaraha\AppData\Roaming\ESRI\Desktop10.5\ArcCatalog\GISAD@PWDSTORMSQL_G
ISTEST1.sde\IMPERVIOUS_2015_IA_intersect_CSOparcels'

# DO
print'Making Copy: IMPERVIOUS_2015_IA_intersect_CSOparcels --> GISTEST1'
if arcpy.Exists(data_out) is True:
    arcpy.Delete_management(data_out)
    print (arcpy.GetMessages())
arcpy.Copy_management(data_in, data_out)
print '\tCopy Complete: IMPERVIOUS_2015_IA_intersect_CSOparcels --> GISTEST1'

#########################################################################################
########### COPY: IMPERVIOUS_2015_IA_intersect_MS4parcels
########### CSO_Data.gdb--> GISTEST1

data_in =
r'C:\Users\sarah.jaraha\Desktop\CSO_Project\CSO_Data.gdb\IMPERVIOUS_2015_IA_intersect_M
S4parcels'
data_out =
r'C:\Users\sarah.jaraha\AppData\Roaming\ESRI\Desktop10.5\ArcCatalog\GISAD@PWDSTORMSQL_G
ISTEST1.sde\IMPERVIOUS_2015_IA_intersect_MS4parcels'

print 'Making Copy: IMPERVIOUS_2015_IA_intersect_MS4parcels --> GISTEST1'
if arcpy.Exists(data_out) is True:
    arcpy.Delete_management(data_out)
    print (arcpy.GetMessages())
arcpy.Copy_management(data_in, data_out)
print '\tCopy Complete: IMPERVIOUS_2015_IA_intersect_MS4parcels --> GISTEST1'

#########################################################################################
#################################   SUMMARIZE IMPERVIOUS AREA BY BILLING CLASS    #####
#########################################################################################

# SQL
query = """
--Drop Existing Table
exec drop_existing_table 'billing_class_summary_IA';

-- Create Table: billing_class_summary_IA
CREATE TABLE billing_class_summary_IA
(Billing_Class varchar(25),
IA_within_CSO_parcels_sqft int,
Equivalent_Number_of_LFF_CSO int,
Percentage_of_Total_CSO_IA float,
IA_within_MS4_parcels_sqft int,
Equivalent_Number_of_LFF_MS4 int,
Percentage_of_Total_MS4_IA float);
```

```
-- Insert Column: Billing_Class
INSERT INTO billing_class_summary_IA (Billing_Class)
SELECT BILLING_CLASS
FROM PARCELS
GROUP BY BILLING_CLASS;

-- Update Column: IA_within_CSO_parcels_sqft
UPDATE billing_class_summary_IA
SET IA_within_CSO_parcels_sqft = CSO_IA
FROM
    (SELECT BILLING_CLASS, CAST(SUM(Shape.STArea()) AS int) as CSO_IA
    FROM IMPERVIOUS_2015_IA_INTERSECT_CSOPARCELS
    GROUP BY BILLING_CLASS)as cso_ia
WHERE billing_class_summary_IA.BILLING_CLASS = cso_ia.BILLING_CLASS;

-- Update Column: Equivalent_number_of_LFF_CSO
UPDATE billing_class_summary_IA
SET Equivalent_number_of_LFF_CSO = Lincoln_Financial_Fields
FROM
    (SELECT billing_class, ((IA_within_CSO_parcels_sqft) / 651750) as
Lincoln_Financial_Fields
    FROM billing_class_summary_IA)as lff_cso
WHERE billing_class_summary_IA.BILLING_CLASS = lff_cso.BILLING_CLASS;

-- Update Column: Percentage_of_Total_CSO_IA
UPDATE billing_class_summary_IA
SET Percentage_of_Total_CSO_IA = percentage
FROM
    (SELECT round(CAST(IA_within_CSO_parcels_sqft AS FLOAT)/(SELECT
SUM(IA_within_CSO_parcels_sqft) FROM billing_class_summary_IA),2)*100 as percentage,
billing_class
    FROM billing_class_summary_IA) as calc
WHERE calc.BILLING_CLASS = billing_class_summary_IA.BILLING_CLASS;

-- Update Column: IA_within_MS4_parcels_sqft
UPDATE billing_class_summary_IA
SET IA_within_MS4_parcels_sqft = MS4_IA
FROM
    (SELECT BILLING_CLASS, CAST(SUM(Shape.STArea()) AS int) as MS4_IA
    FROM IMPERVIOUS_2015_IA_INTERSECT_MS4PARCELS
    GROUP BY BILLING_CLASS)as ms4_ia
WHERE billing_class_summary_IA.BILLING_CLASS = ms4_ia.BILLING_CLASS;

-- Update Column: Equivalent_number_of_LFF_MS4
UPDATE billing_class_summary_IA
SET Equivalent_number_of_LFF_MS4 = Lincoln_Financial_Fields
FROM
    (SELECT billing_class, ((IA_within_MS4_parcels_sqft) / 651750) as
Lincoln_Financial_Fields
    FROM billing_class_summary_IA)as lff_ms4
WHERE billing_class_summary_IA.BILLING_CLASS = lff_ms4.BILLING_CLASS;
```

```
-- Update Column: Percentage_of_Total_MS4_IA
UPDATE billing_class_summary_IA
SET Percentage_of_Total_MS4_IA = percentage
FROM
    (SELECT round(CAST(IA_within_MS4_parcels_sqft AS FLOAT)/(SELECT
SUM(IA_within_MS4_parcels_sqft) FROM billing_class_summary_IA),2)*100 as percentage,
billing_class
    FROM billing_class_summary_IA) as calc
WHERE calc.BILLING_CLASS = billing_class_summary_IA.BILLING_CLASS;


"""

# DO
print 'Creating Table: billing_class_summary_IA'
cursor.execute(query)
print '\tTable Complete: billing_class_summary_IA'
```