The PCI+ Python Model: Selecting Road Projects That Optimize Pavement Condition Plus Other Civic Priorities

Kevin Gustavson, Ph.D., Information Technology
Craig Schaefer, Engineering Services

SCAUG Regional Conference – April 2019

CITY OF Tulsa
A New Kind of Energy™

Overall Goal of the PCI+ Model

Create a largely automated model for street prioritization that:

- **Maintains Pavement Condition Index (PCI)** of **65** citywide
  - Arterials → **67**
  - Non-arterials → **63**
- Layers in **other priorities,** and
- Provides a mechanism to **weight those priorities**.

Pavement Condition Index (0 – 100)

Numerical way to represent the condition of pavement

100	Newly constructed
70 – 99	Generally only routine maintenance needed
40 – 70	Rehabilitation (i.e., mill and overlay)
0 – 40	Reconstruction needed (depending on type)

Lowest PCI values → 5x more costly to repair

Reconstruction → reset to 100

Rehabilitation → could be less than 100

Three Main Pavement Types

Concrete - “PCC”

This Photo by Unknown Author is licensed under CC BY-SA

Most Expensive
Longest lasting (~40 yr)

Asphalt – “AC”

This Photo by Unknown Author is licensed under CC BY-SA

Less Expensive
Shorter life (~30 yr)

(The model treats asphalt over asphalt the same as just asphalt)

Asphalt Overlay
Asphalt over Concrete –”APC”

This Photo by Unknown Author is licensed under CC BY-SA

Much Less Expensive
Shortest life (~10 yr)

Engineering Consultant – Evaluation & ICON Model

- Surveys streets every 5-6 years
  - Measures PCI using core samples, inspection
  - Can project PCI values using typical decay rates.

ICON Model: Picks road projects to maintain desired PCI.

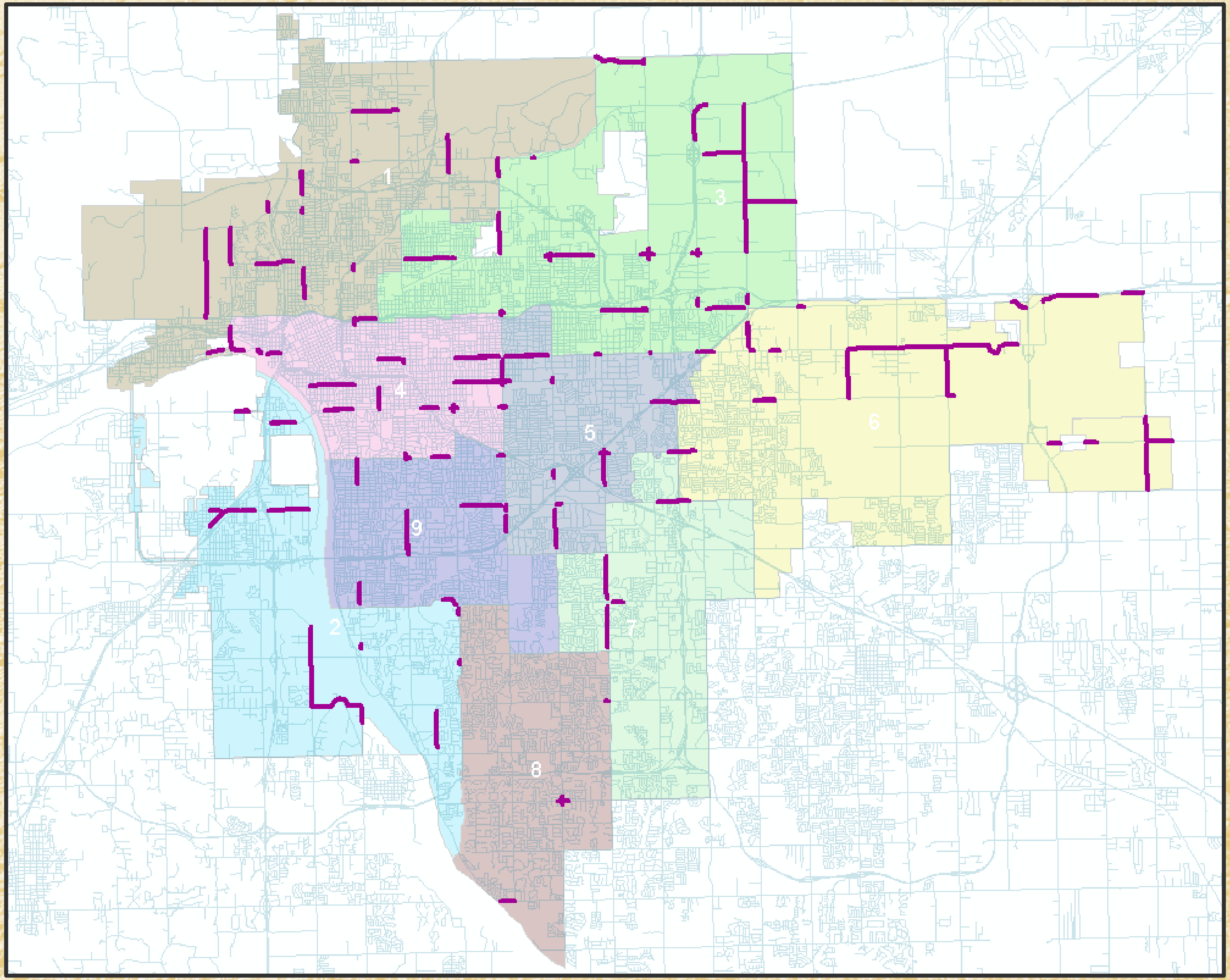
Indicates:

- Improvement method (rehabilitation, reconstruction, etc.)
- Cost
- Year for project

ICON Model Selections

Patchwork:

- Small portions of intersections or corridors
- Often only one side of the street
- Most are **not** reasonable construction projects

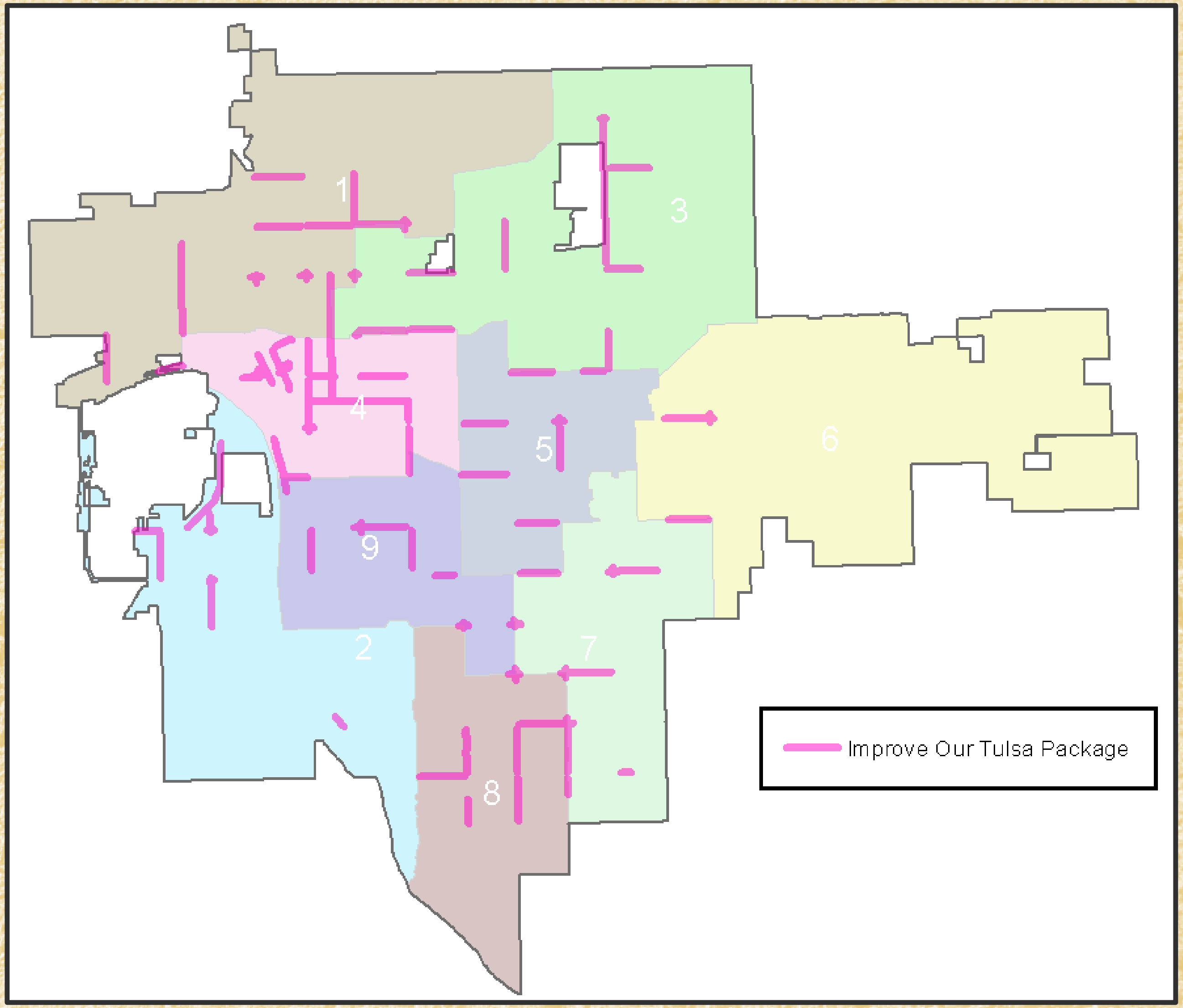


Improvements That Maintain a PCI of 67 - Arterial

Past Improvement Packages

Engineers would work intensively (short time-frame) to:

- Work segments into complete projects
  - Intersections
  - Corridors
- Consider other civic priorities (pipe replacement, etc)



Urban Data Pioneers (UDP)

Program started by the Bynum Administration.

Mission → Improve the use of data throughout the City of Tulsa.

Teams of City Employees and Community Members work on problems together

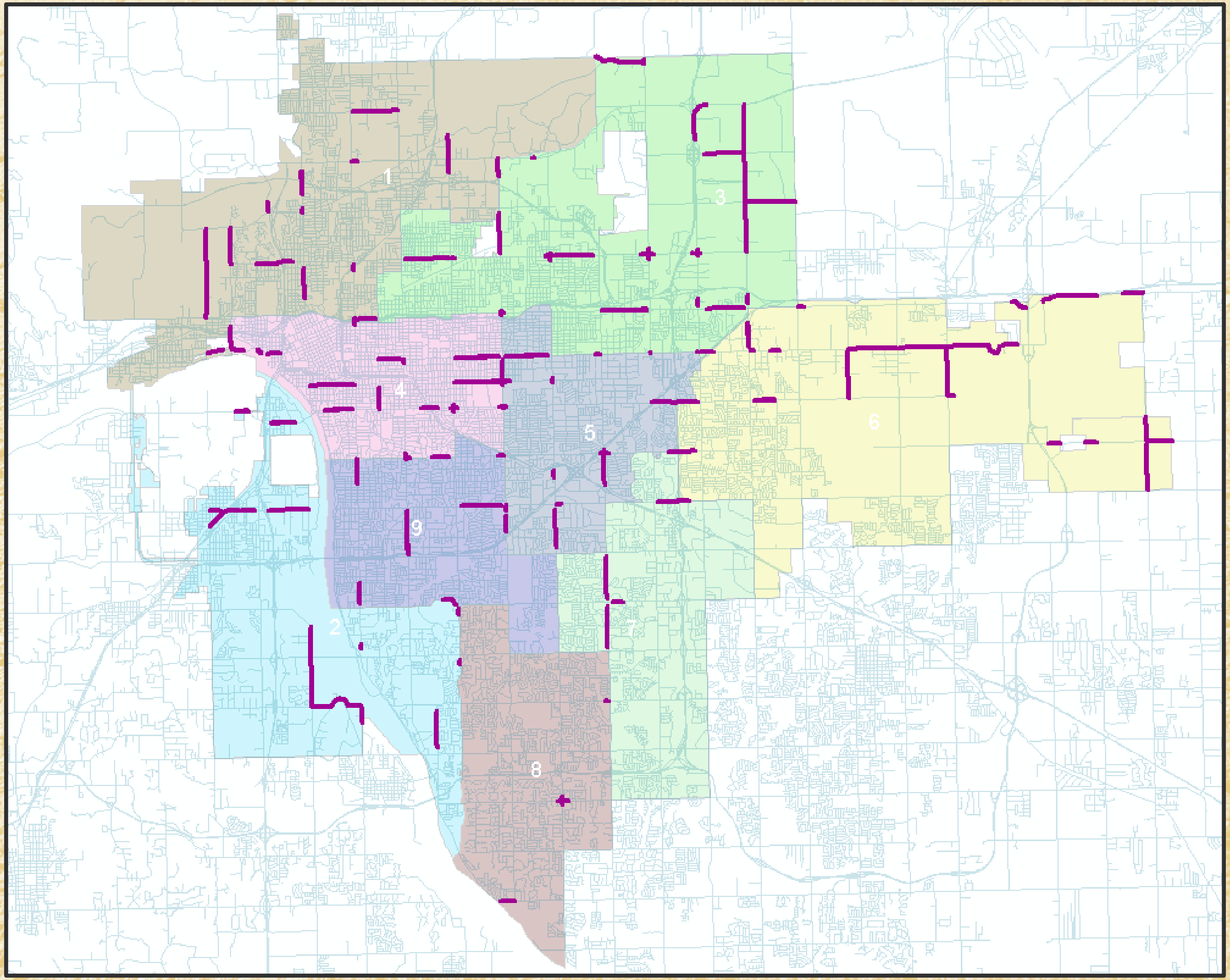
PCI+ One of the first UDP projects, started in 2017

→ improve data analysis in selecting roads

Tulsa won the "Engaged Cities Award" largely because of the Urban Data Pioneers program

PCI+ Model Objectives

- Create workable projects from ICON data
- Simultaneously attain other civic goals
- Allow prioritization of civic goals by Administration
- Spend appropriate percentage of budget per district
- ***Meet the PCI goal***



Improvements That Maintain a PCI of 67 - Arterial

First Modeling Strategy

Shuffle pavement areas and PCI values within each district.

- Modeled after process Engineers used in the past.
- Not enough wiggle room → FAILED

Second Modeling Strategy

Focus on the money → SUCCESS

- Shuffle PCI values and areas City-wide
- Estimate costs for roads as if all done in 2024
- Let dollar value determine a district's fair share of projects.

Road selection is the final step in the model. Looking ahead to see where to begin...

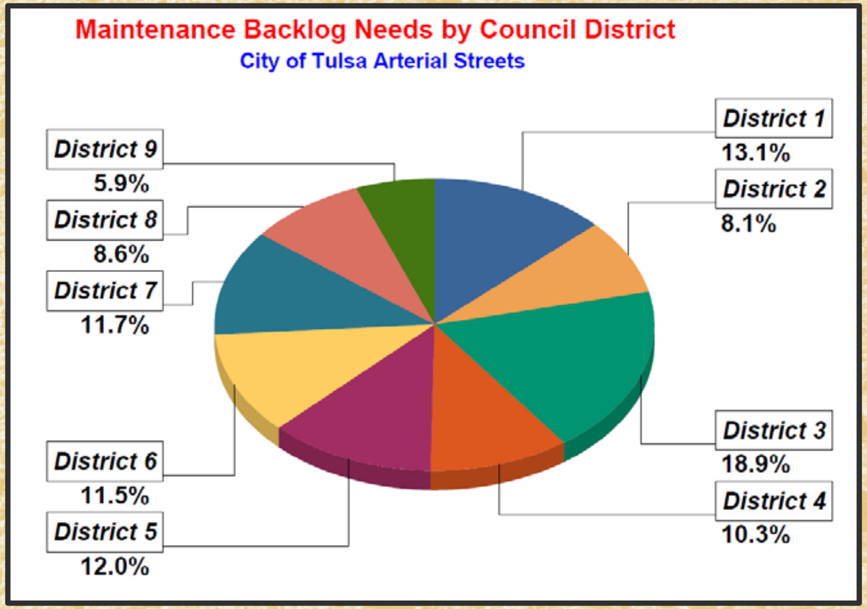
The PCI+ Model

Part I: Make the Base Road feature class

Work with data provided by consultant to:

- Create workable intersection and corridor projects
- Include all needed data to select projects that
  - Maintain the city PCI goal

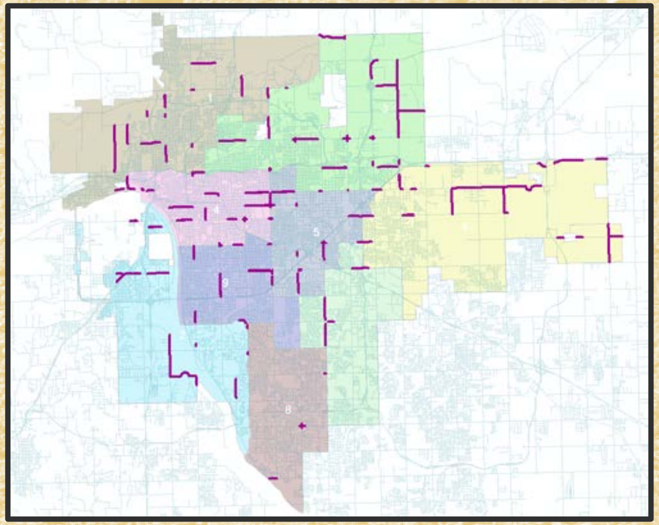
    (pavement area, type, and PCI value)
  - Spend the appropriate amount of money per district (costs)



Consultant provides % need per district (cost)

Engineering Consultant Provides 3 Shapefiles

- **Current** – Street segments with projected PCI values plus much more (pavement type, street width, etc.)
- **Scenario** – Same segments as above with:
  - ICON model selections indicated
    (based on available budget and target PCI)
  - Cost of recommended actions (2021-2026).

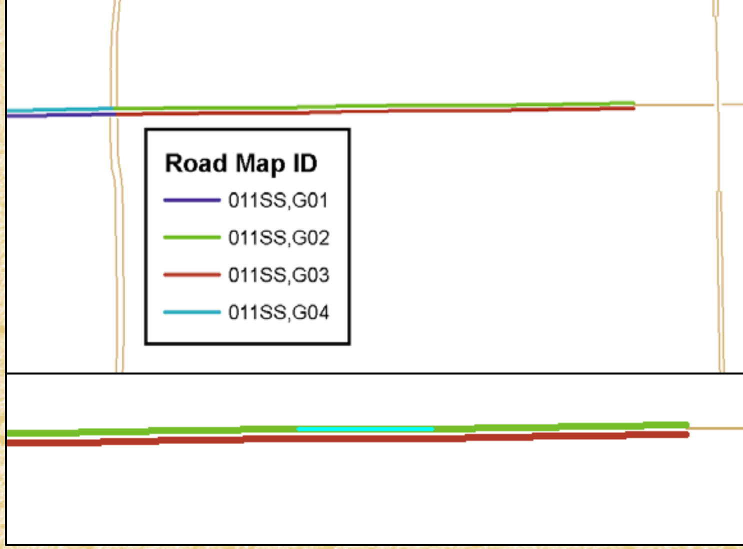


By request:

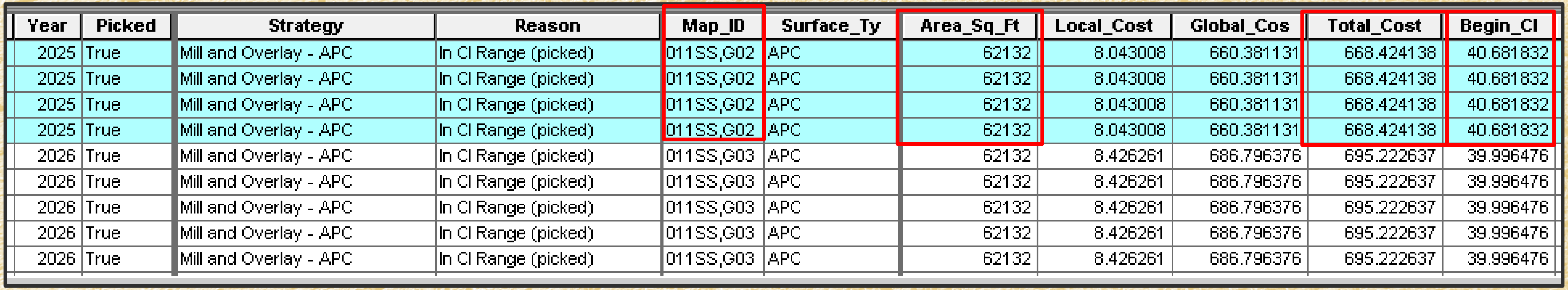
- **2024 Cost** and treatment for every segment

  (ICON model with unlimited budget spent entirely in 2024)
  - For cost of any road selected by PCI+ model

Get to Know the Data



- No Unique Identifier…
  - several segments make up one Map_ID (same PCI, same total area, same costs)
    → need to be dissolved

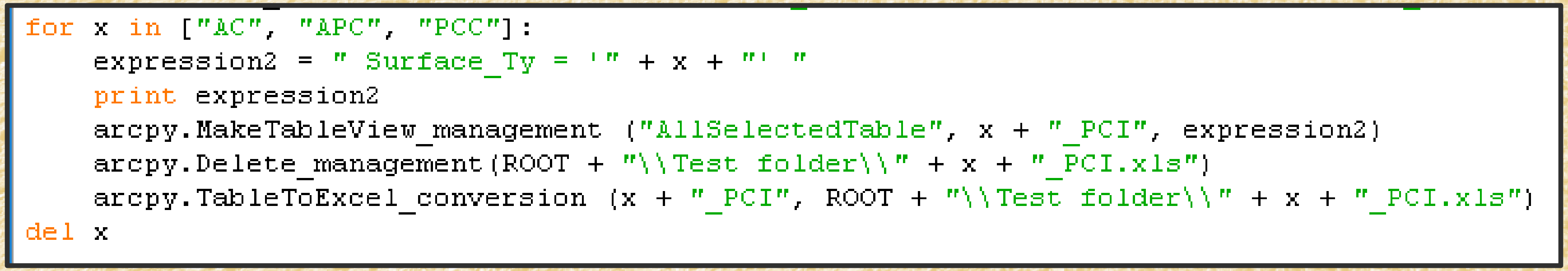


Year	Picked	Strategy	Reason	Map_ID	Surface_Ty	Area_Sq_Ft	Local_Cost	Global_Cos	Total_Cost	Begin_CI
2025	True	Mill and Overlay - APC	In CI Range (picked)	011SS,G02	APC	62132	8.043008	660.381131	668.424138	40.681832
2025	True	Mill and Overlay - APC	In CI Range (picked)	011SS,G02	APC	62132	8.043008	660.381131	668.424138	40.681832
2025	True	Mill and Overlay - APC	In CI Range (picked)	011SS,G02	APC	62132	8.043008	660.381131	668.424138	40.681832
2025	True	Mill and Overlay - APC	In CI Range (picked)	011SS,G02	APC	62132	8.043008	660.381131	668.424138	40.681832
2026	True	Mill and Overlay - APC	In CI Range (picked)	011SS,G03	APC	62132	8.426261	686.796376	695.222637	39.996476
2026	True	Mill and Overlay - APC	In CI Range (picked)	011SS,G03	APC	62132	8.426261	686.796376	695.222637	39.996476
2026	True	Mill and Overlay - APC	In CI Range (picked)	011SS,G03	APC	62132	8.426261	686.796376	695.222637	39.996476
2026	True	Mill and Overlay - APC	In CI Range (picked)	011SS,G03	APC	62132	8.426261	686.796376	695.222637	39.996476
2026	True	Mill and Overlay - APC	In CI Range (picked)	011SS,G03	APC	62132	8.426261	686.796376	695.222637	39.996476

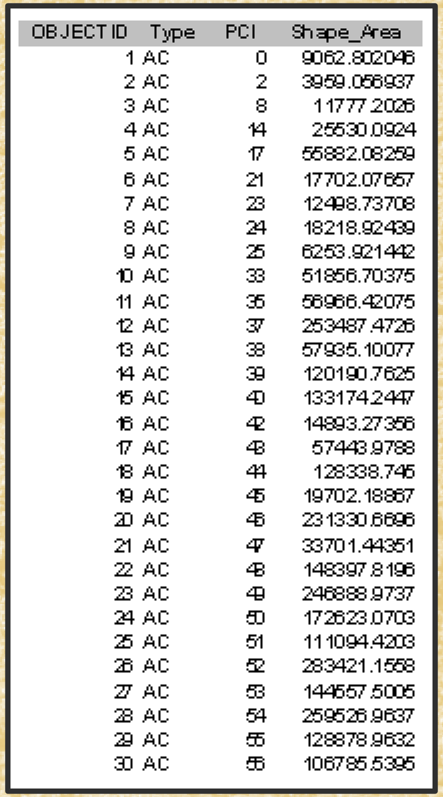
- Cost in thousands
- Lopping off the last character of Map_ID allows dissolving into:
  - reasonable corridors (ending in a digit)
  - intersections (ending in a letter)

Create PCI Tables for Each Pavement Type

- Select the ICON Picked = ‘True’ records
- Dissolve by Map_ID, Surface_Ty, PCI, & Area
- Dissolve again by Surface_Ty and PCI…sum the areas
- Create tables for AC, APC, and PCC pavements:
    - Loop through each pavement type:
        - Make a table view for pavement type
        - Delete the old table output
        - Save off a new excel table for each pavement type



```
for x in ["AC", "APC", "PCC"]:
    expression2 = " Surface_Ty = '" + x + "' "
    print expression2
    arcpy.MakeTableView_management ("AllSelectedTable", x + "_PCI", expression2)
    arcpy.Delete_management(ROOT + "\\Test folder\\" + x + "_PCI.xls")
    arcpy.TableToExcel_conversion (x + "_PCI", ROOT + "\\Test folder\\" + x + "_PCI.xls")
del x
```

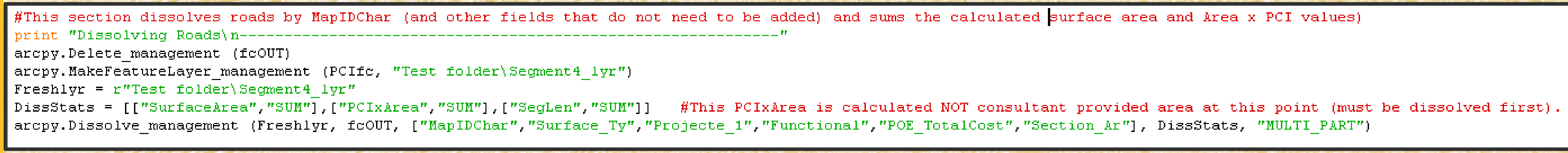


OBJECTID	Type	PCI	Shape_Area
1	AC	0	9062.802046
2	AC	2	3959.056937
3	AC	8	11777.2026
4	AC	14	25530.0924
5	AC	17	55882.08259
6	AC	21	17702.07657
7	AC	23	12498.73708
8	AC	24	18218.92439
9	AC	25	6253.921442
10	AC	33	51856.70375
11	AC	35	56966.42075
12	AC	37	253487.4726
13	AC	38	57935.10077
14	AC	39	120190.7625
15	AC	40	133174.2447
16	AC	42	14893.27356
17	AC	43	57443.9788
18	AC	44	128338.745
19	AC	45	19702.18867
20	AC	46	231330.6696
21	AC	47	33701.44351
22	AC	48	148397.8196
23	AC	49	246888.9737
24	AC	50	172623.0703
25	AC	51	111094.4203
26	AC	52	283421.1558
27	AC	53	144657.5005
28	AC	54	259526.9637
29	AC	55	128878.9632
30	AC	56	106785.5395

Create the Base Roads feature class

- Create feature classes from the shapefiles (from consultant)
- Add new fields to one and pull needed data from the others
  - Uses UpdateCursor with SearchCursor
  - Selects an identical feature in the other feature class
  - Pulls in & manipulates needed data

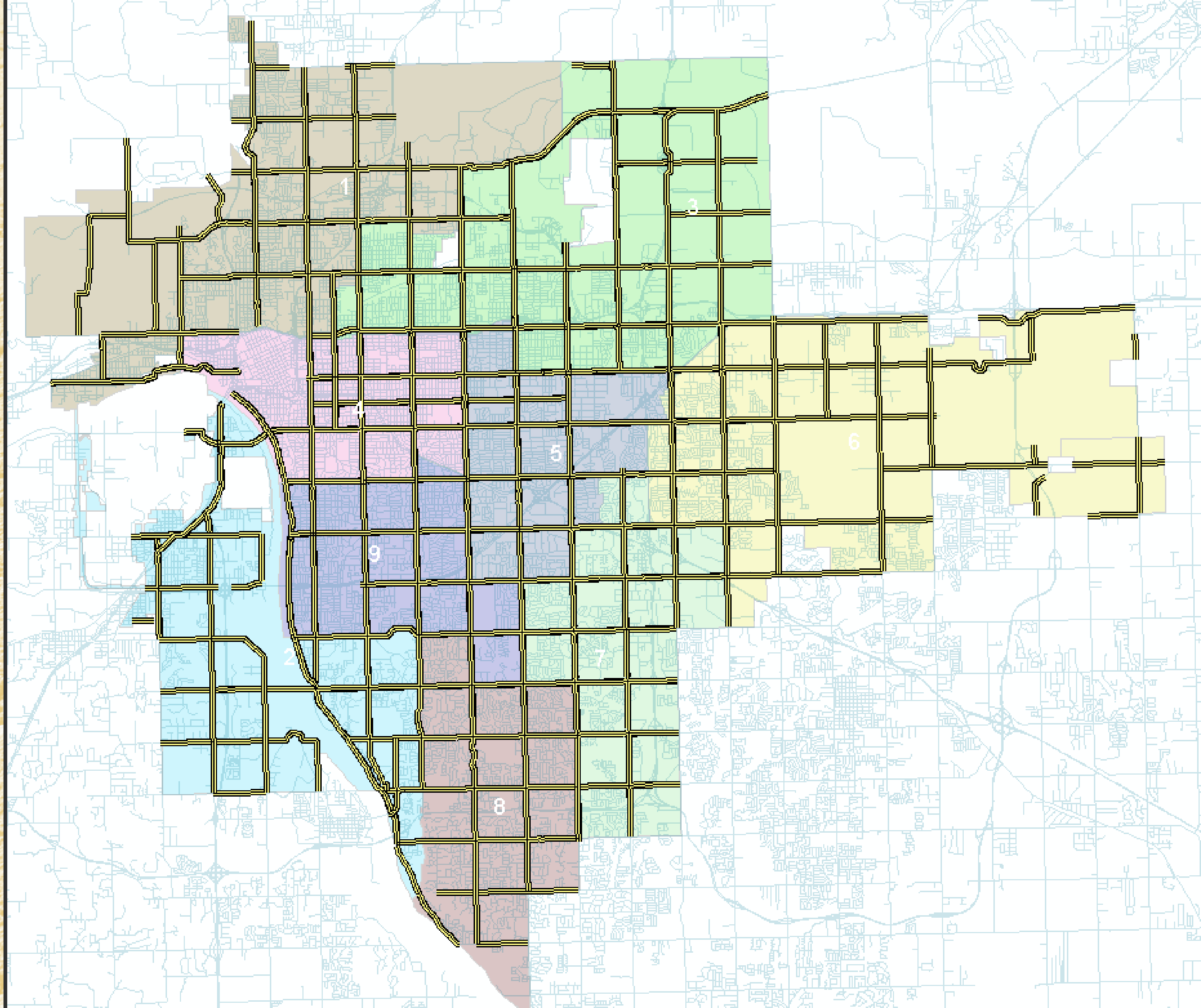
  Main fields added:
    - PCI x Area (Useful when dissolving areas…dividing again by area gives avg PCI value)
    - 2024 Cost → of construction
    - Segment length (for “manual” ratio policy in next script)
- Dissolve to Map_ID level



```
#This section dissolves roads by MapIDChar (and other fields that do not need to be added) and sums the calculated surface area and Area x PCI values)
print "Dissolving Roads\n------------------------------------------------------------"
arcpy.Delete_management (fcOUT)
arcpy.MakeFeatureLayer_management (PCIfc, "Test folder\Segment4_lyr")
Freshlyr = r"Test folder\Segment4_lyr"
DissStats = [["SurfaceArea","SUM"],["PCIxArea","SUM"],["SegLen","SUM"]]    #This PCIxArea is calculated NOT consultant provided area at this point (must be dissolved first).
arcpy.Dissolve_management (Freshlyr, fcOUT, ["MapIDChar","Surface_Ty","Projecte_1","Functional","POE_TotalCost","Section_Ar"], DissStats, "MULTI_PART")
```

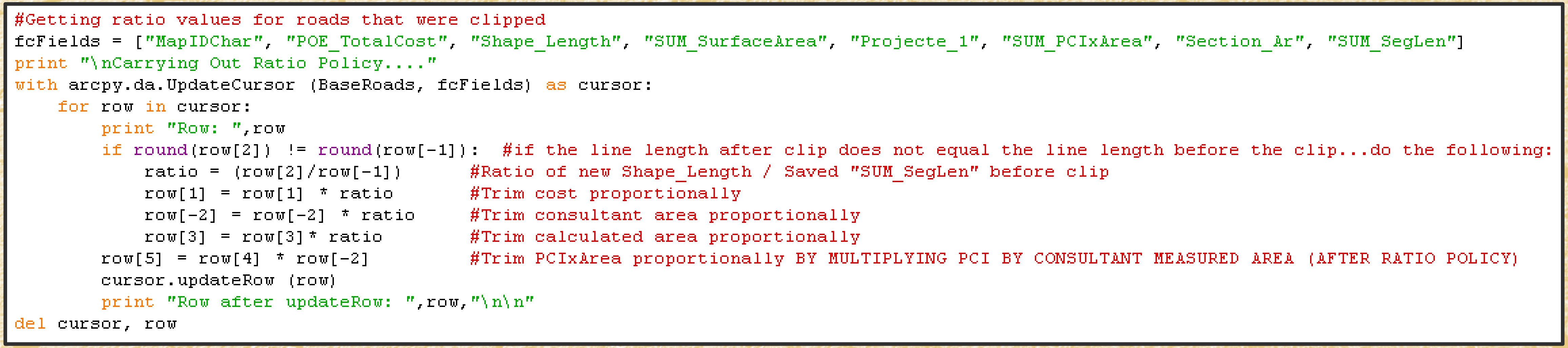
Initial Feature Class

- Part of some roads outside city limits
- Roads along city borders are not city responsibility



Cut Parts of Roads Outside City Limits

- Clip the road feature class by city limits.
- Ratio policy would not work in script (field_info.setSplitRule)
  - even exporting script from Model Builder to use (did not work)
  - could not figure out why…so….

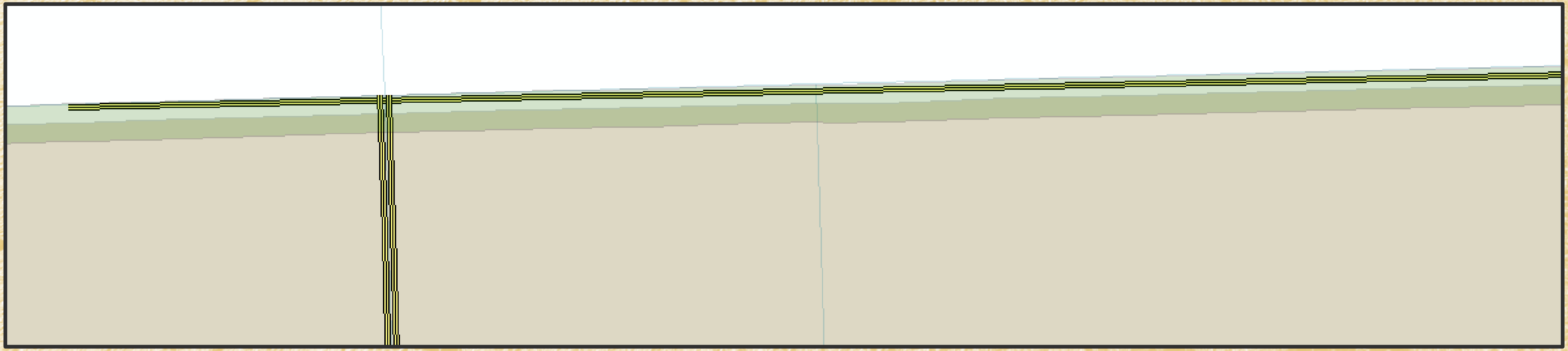


```
#Getting ratio values for roads that were clipped
fcFields = ["MapIDChar", "POE_TotalCost", "Shape_Length", "SUM_SurfaceArea", "Projecte_1", "SUM_PCIxArea", "Section_Ar", "SUM_SegLen"]
print "\nCarrying Out Ratio Policy...."
with arcpy.da.UpdateCursor (BaseRoads, fcFields) as cursor:
    for row in cursor:
        print "Row: ",row
        if round(row[2]) != round(row[-1]):  #if the line length after clip does not equal the line length before the clip...do the following:
            ratio = (row[2]/row[-1])      #Ratio of new Shape_Length / Saved "SUM_SegLen" before clip
            row[1] = row[1] * ratio       #Trim cost proportionally
            row[-2] = row[-2] * ratio     #Trim consultant area proportionally
            row[3] = row[3]* ratio        #Trim calculated area proportionally
        row[5] = row[4] * row[-2]         #Trim PCIxArea proportionally BY MULTIPLYING PCI BY CONSULTANT MEASURED AREA (AFTER RATIO POLICY)
        cursor.updateRow (row)
        print "Row after updateRow: ",row,"\n\n"
del cursor, row
```

- Python ratio policy: goes through each record, checks for segment length change, if so, multiplies data by length ratio.

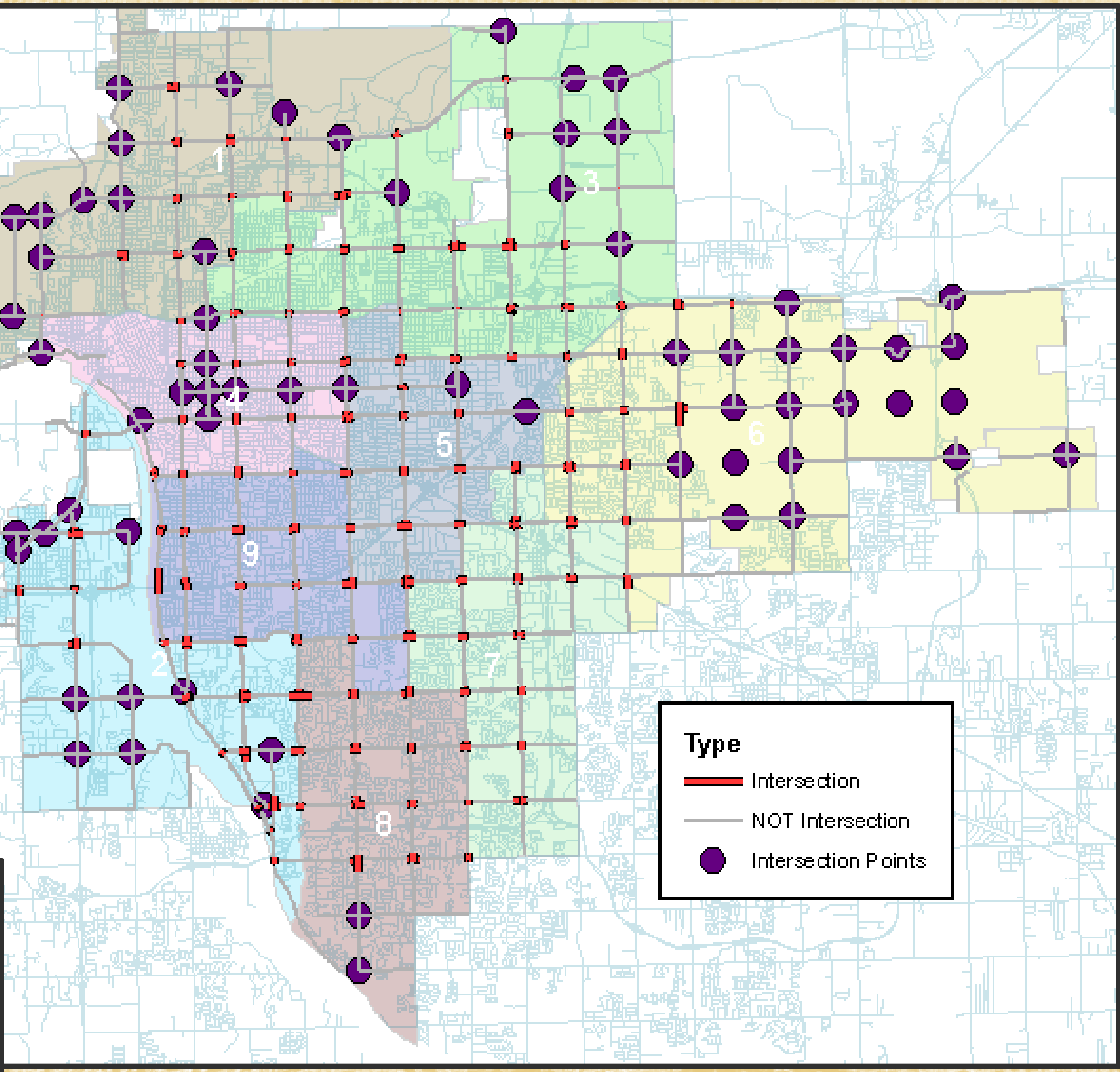
Cut Roads Along City Boundary

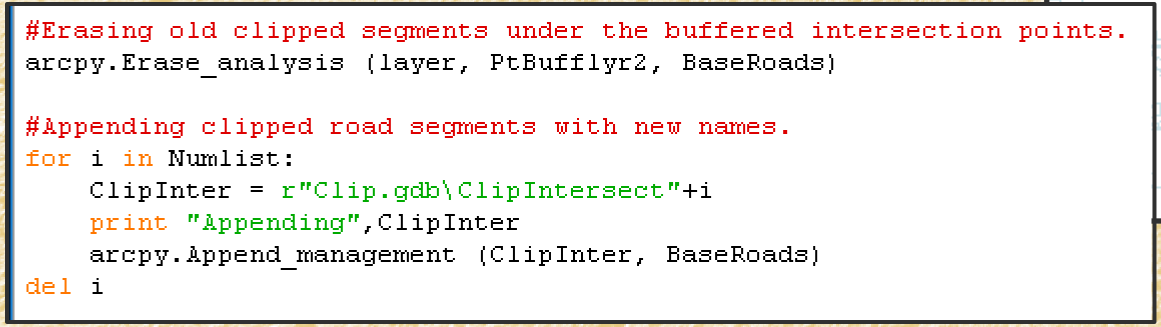
- Buffer the city boundary by 50 feet (geometries not perfect)
- Select all of the features that “HAVE_THEIR_CENTER_IN” the buffer.
- Delete them all.



Create Intersections Where Needed

- Where intersection geometry absent: buffer points 275 ft.
- Clip roads with buffered points → ratio policy to get proportional area, cost, etc. (Arc tool works this time!)
- Renames ID and type for new intersection pieces
- “Erases” old and “appends” new clipped segments



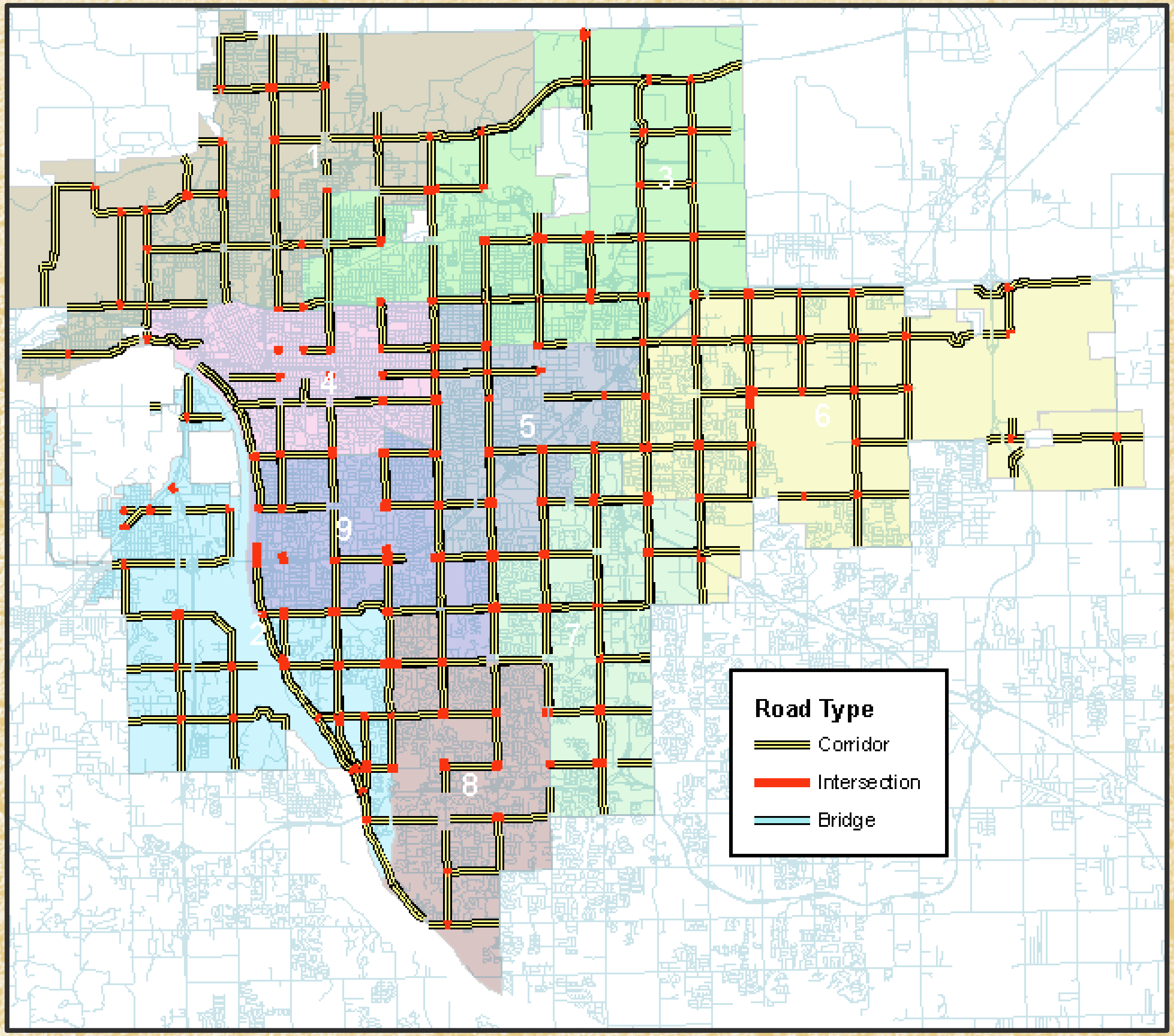


```
#Erasing old clipped segments under the buffered intersection points.
arcpy.Erase_analysis (layer, PtBufflyr2, BaseRoads)

#Appending clipped road segments with new names.
for i in Numlist:
    ClipInter = r"Clip.gdb\ClipIntersect"+i
    print "Appending",ClipInter
    arcpy.Append_management (ClipInter, BaseRoads)
del i
```

Remove “Predetermine” and Dataless Road Segments

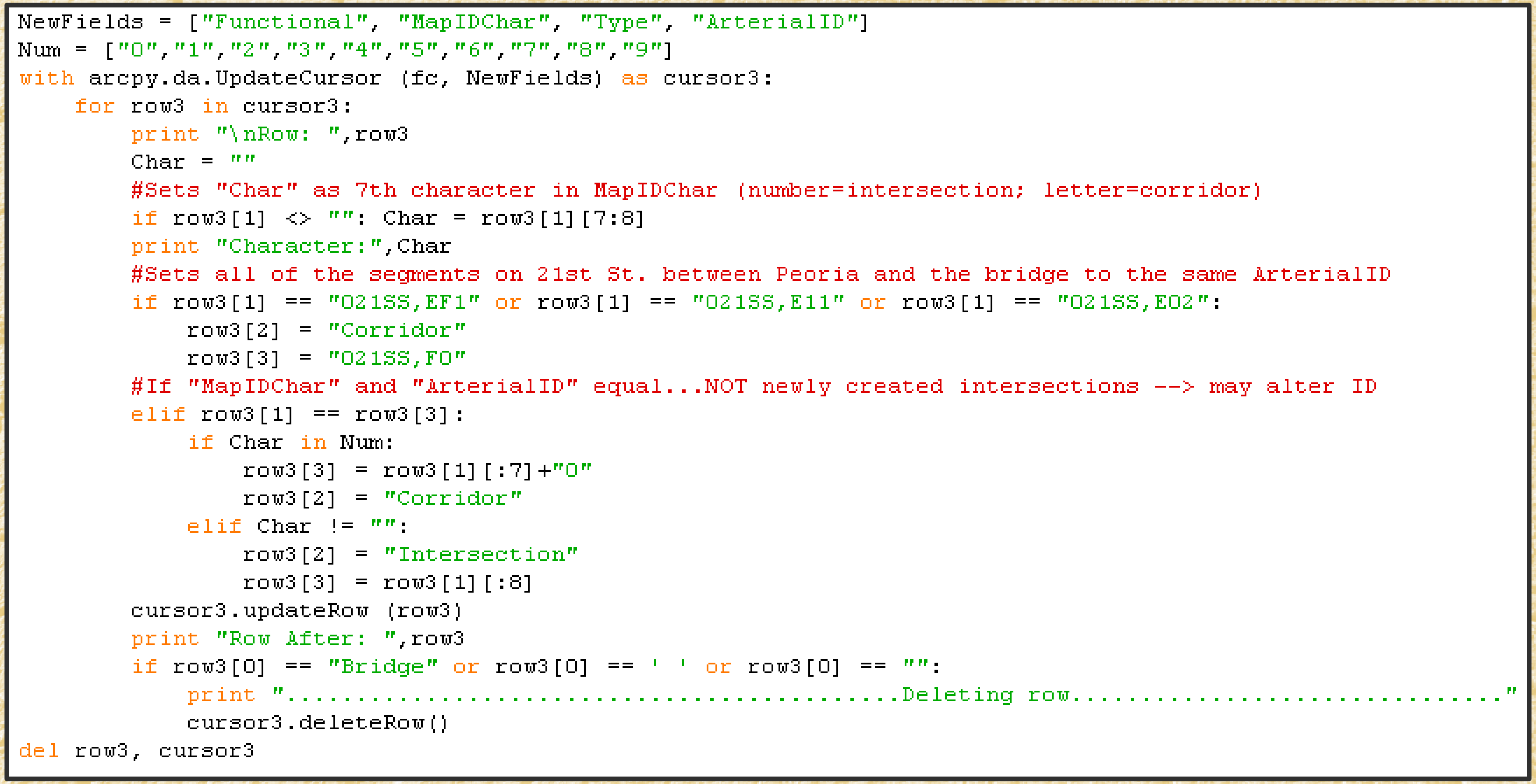
- Delete “Predetermine” roads
  - Ones already funded for upcoming improvements

- Delete roads with no data
  - Fringe roads included in original dataset



Manipulate ID field; Save in New “Arterial_ID” field

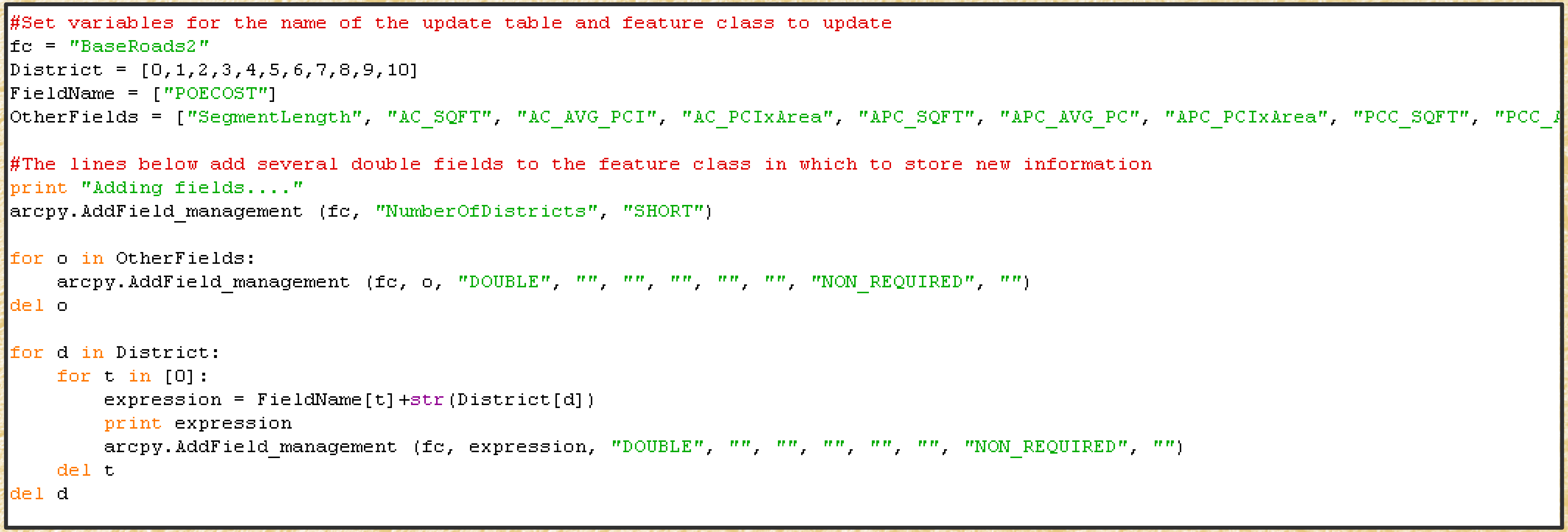
- Adds new fields and fills them
  - “ArterialID”: Shortened ID field → allows dissolve into corridors & intersections
  - “Type”: Intersection vs. Corridor (based on 7th character in “MapIDChar”)



```
NewFields = ["Functional", "MapIDChar", "Type", "ArterialID"]
Num = ["0","1","2","3","4","5","6","7","8","9"]
with arcpy.da.UpdateCursor (fc, NewFields) as cursor3:
    for row3 in cursor3:
        print "\nRow: ",row3
        Char = ""
        #Sets "Char" as 7th character in MapIDChar (number=intersection; letter=corridor)
        if row3[1] <> "": Char = row3[1][7:8]
        print "Character:",Char
        #Sets all of the segments on 21st St. between Peoria and the bridge to the same ArterialID
        if row3[1] == "021SS,EF1" or row3[1] == "021SS,E11" or row3[1] == "021SS,E02":
            row3[2] = "Corridor"
            row3[3] = "021SS,F0"
        #If "MapIDChar" and "ArterialID" equal...NOT newly created intersections --> may alter ID
        elif row3[1] == row3[3]:
            if Char in Num:
                row3[3] = row3[1][:7]+"0"
                row3[2] = "Corridor"
            elif Char != "":
                row3[2] = "Intersection"
                row3[3] = row3[1][:8]
        cursor3.updateRow (row3)
        print "Row After: ",row3
        if row3[0] == "Bridge" or row3[0] == ' ' or row3[0] == "":
            print "............................................Deleting row..............................."
            cursor3.deleteRow()
del row3, cursor3
```

Create 21 Fields (Easier than Model Builder)

- Sets up needed fields for upcoming data manipulation
  - Cost fields for each district 1 – 9 [“POECOST1”, “POECOST2”, …]
    - “POECOST0” is total cost of all districts
    - “POECOST10” is cost for downtown (a part of district 4)
  - 3 fields for each type of pavement (SQFT, AVG_PCI, PCIxArea)



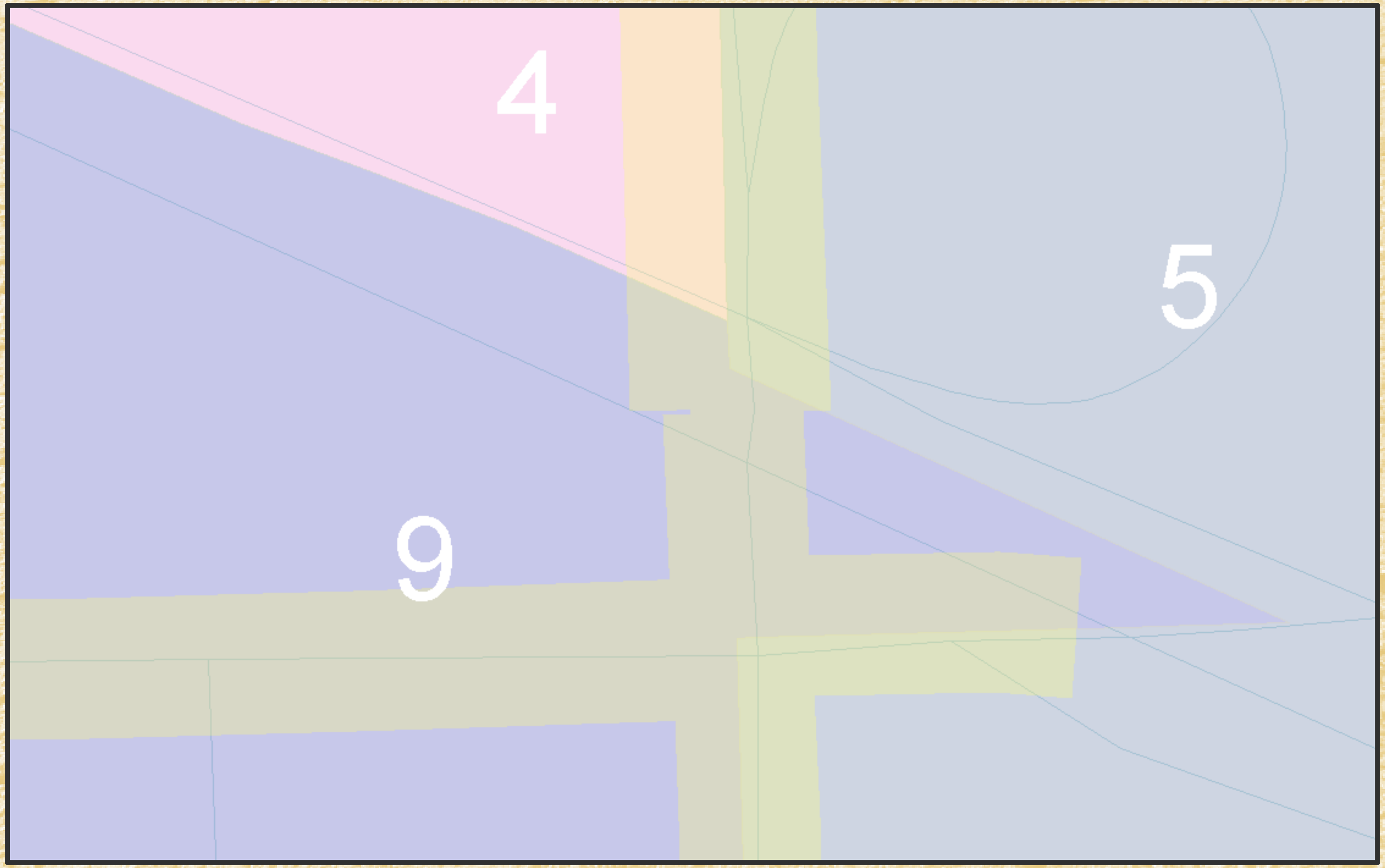
```
#Set variables for the name of the update table and feature class to update
fc = "BaseRoads2"
District = [0,1,2,3,4,5,6,7,8,9,10]
FieldName = ["POECOST"]
OtherFields = ["SegmentLength", "AC_SQFT", "AC_AVG_PCI", "AC_PCIxArea", "APC_SQFT", "APC_AVG_PC", "APC_PCIxArea", "PCC_SQFT", "PCC_A

#The lines below add several double fields to the feature class in which to store new information
print "Adding fields...."
arcpy.AddField_management (fc, "NumberOfDistricts", "SHORT")

for o in OtherFields:
    arcpy.AddField_management (fc, o, "DOUBLE", "", "", "", "", "", "NON_REQUIRED", "")
del o

for d in District:
    for t in [0]:
        expression = FieldName[t]+str(District[d])
        print expression
        arcpy.AddField_management (fc, expression, "DOUBLE", "", "", "", "", "", "NON_REQUIRED", "")
    del t
del d
```

Add Data By District

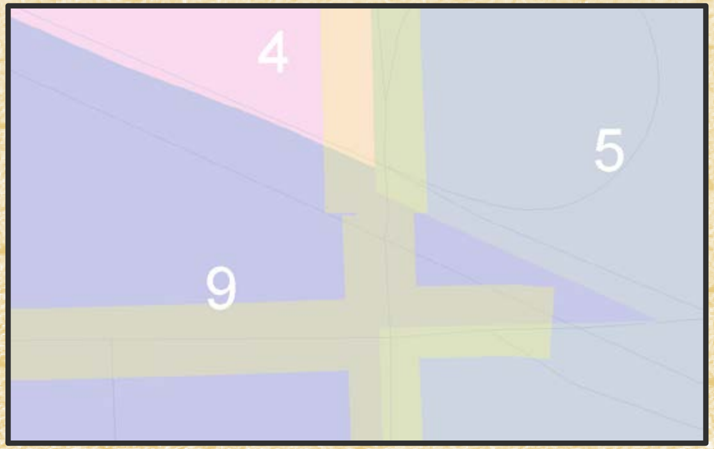
- Some roads straddle district boundaries
- Need appropriate proportion of cost for each district

→Buffer the road layer

→Now a polygon



Add Data By District



- Go through district geometry one by one
- Clip road layer with district polygon

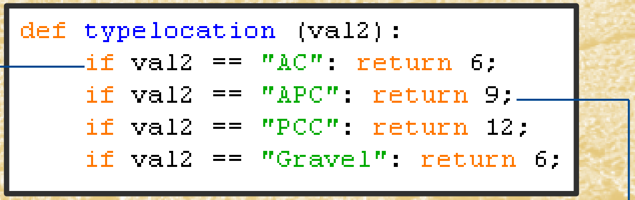
  → use ratio policy (it works again!)
- Fill in the 21 fields appropriately:

  District 3 Clip - only parts of roads from District 3 (x = 3)

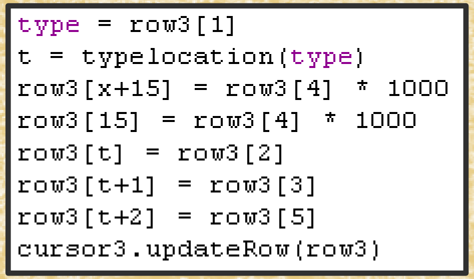
  - POECOST0 - add to total budget
  - POECOST3 - add to District 3 budget (3 + 15)
  - If the segment is “Surface_Ty” of “AC”:

    →Fill in the AC fields (List position start: t = 6)

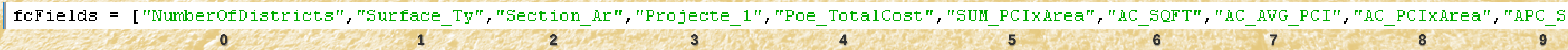
    - AC_SQFT (t = 6)
    - AC_AVG_PCI (t+1 = 7)
    - AC_PCIxArea (t+2 = 8)



```
def typelocation (val2):
    if val2 == "AC": return 6;
    if val2 == "APC": return 9;
    if val2 == "PCC": return 12;
    if val2 == "Gravel": return 6;
```



```
type = row3[1]
t = typelocation(type)
row3[x+15] = row3[4] * 1000
row3[15] = row3[4] * 1000
row3[t] = row3[2]
row3[t+1] = row3[3]
row3[t+2] = row3[5]
cursor3.updateRow(row3)
```

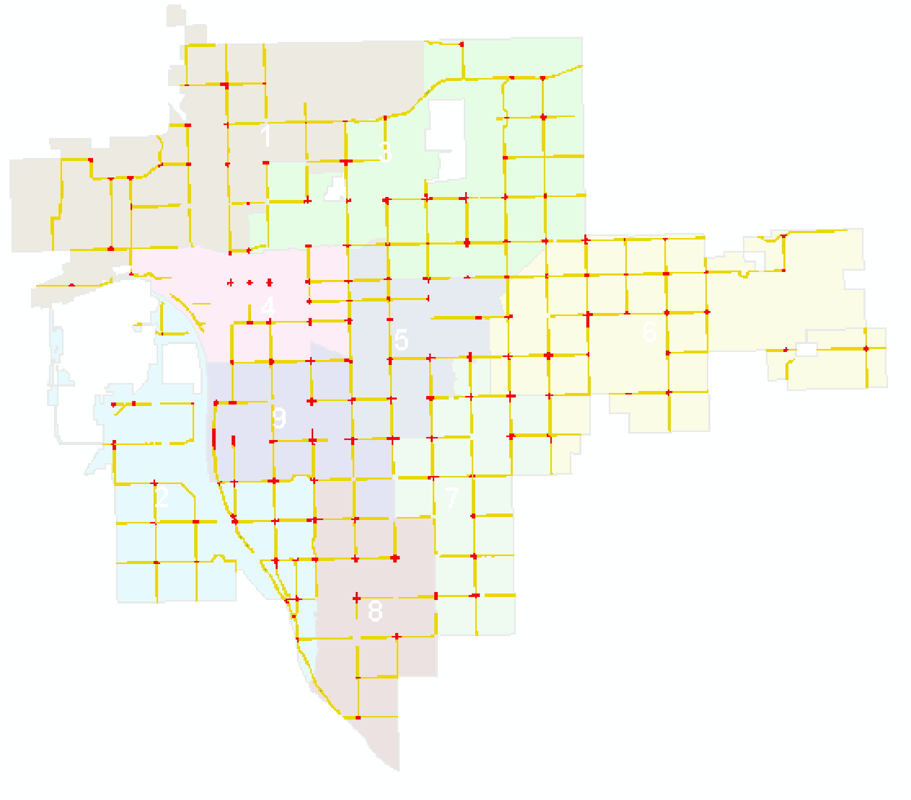


```
fcFields = ["NumberOfDistricts","Surface_Ty","Section_Ar","Projecte_1","Poe_TotalCost","SUM_PCIxArea","AC_SQFT","AC_AVG_PCI","AC_PCIxArea","APC_S
```

0 1 2 3 4 5 6 7 8 9

Append and Dissolve

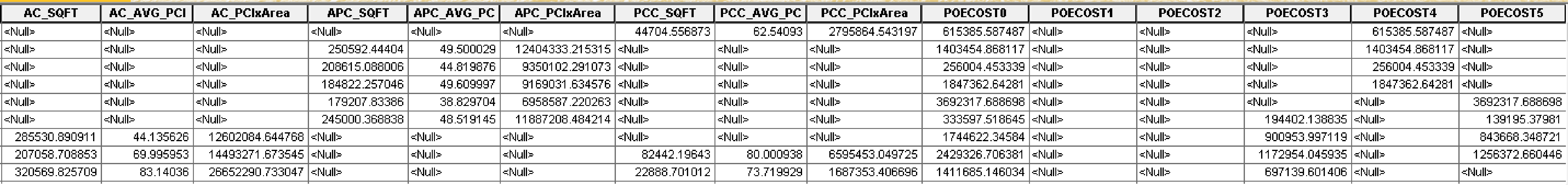
- Append 9 districts back together
- Dissolve on “Arterial_ID”
  – Sum all 21 fields



Now, corridors and intersections:

→in single pieces

→contain all of the cost and PCI data needed



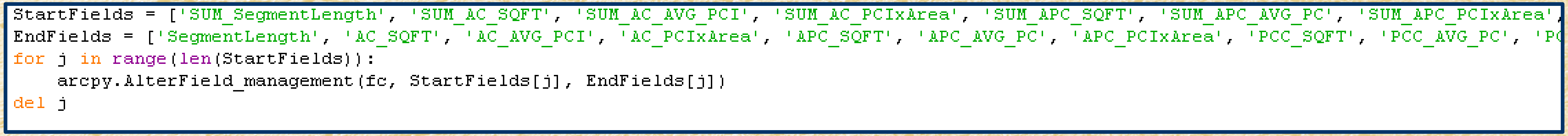
AC_SQFT	AC_AVG_PCI	AC_PCIxArea	APC_SQFT	APC_AVG_PC	APC_PCIxArea	PCC_SQFT	PCC_AVG_PC	PCC_PCIxArea	POECOST0	POECOST1	POECOST2	POECOST3	POECOST4	POECOST5
<Null>	<Null>	<Null>	<Null>	<Null>	<Null>	44704.556873	62.54093	2795864.543197	615385.587487	<Null>	<Null>	<Null>	615385.587487	<Null>
<Null>	<Null>	<Null>	250592.44404	49.500029	12404333.215315	<Null>	<Null>	<Null>	1403454.868117	<Null>	<Null>	<Null>	1403454.868117	<Null>
<Null>	<Null>	<Null>	208615.088006	44.819876	9350102.291073	<Null>	<Null>	<Null>	256004.453339	<Null>	<Null>	<Null>	256004.453339	<Null>
<Null>	<Null>	<Null>	184822.257046	49.609997	9169031.634576	<Null>	<Null>	<Null>	1847362.64281	<Null>	<Null>	<Null>	1847362.64281	<Null>
<Null>	<Null>	<Null>	179207.83386	38.829704	6958587.220263	<Null>	<Null>	<Null>	3692317.688698	<Null>	<Null>	<Null>	<Null>	3692317.688698
<Null>	<Null>	<Null>	245000.368838	48.519145	11887208.484214	<Null>	<Null>	<Null>	333597.518645	<Null>	<Null>	194402.138835	<Null>	139195.37981
285530.890911	44.135626	12602084.644768	<Null>	<Null>	<Null>	<Null>	<Null>	<Null>	1744622.34584	<Null>	<Null>	900953.997119	<Null>	843668.348721
207058.708853	69.995953	14493271.673545	<Null>	<Null>	<Null>	82442.19643	80.000938	6595453.049725	2429326.706381	<Null>	<Null>	1172954.045935	<Null>	1256372.660446
320569.825709	83.14036	26652290.733047	<Null>	<Null>	<Null>	22888.701012	73.719929	1687353.406696	1411685.146034	<Null>	<Null>	697139.601406	<Null>	<Null>

→ Another script converts all of the Null values to zero.

Alter Field Names and Recalculate Average PCI

- Get Rid of “SUM_” in the summed fields.

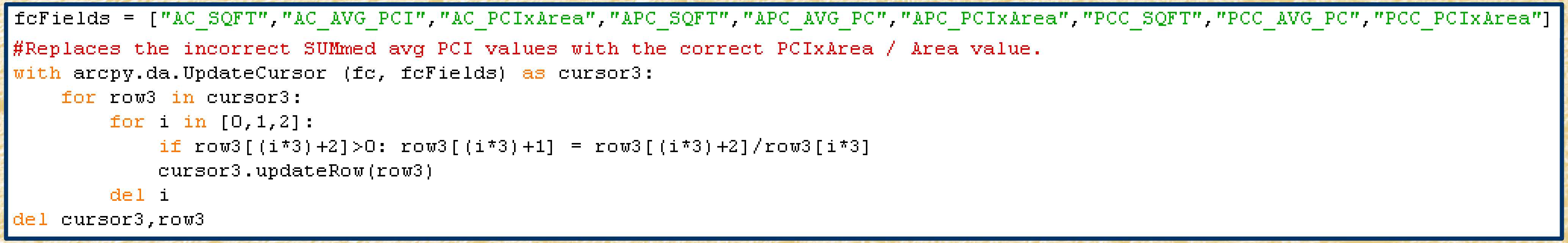
  → Quick with Python



```
StartFields = ['SUM_SegmentLength', 'SUM_AC_SQFT', 'SUM_AC_AVG_PCI', 'SUM_AC_PCIxArea', 'SUM_APC_SQFT', 'SUM_APC_AVG_PC', 'SUM_APC_PCIxArea',
EndFields = ['SegmentLength', 'AC_SQFT', 'AC_AVG_PCI', 'AC_PCIxArea', 'APC_SQFT', 'APC_AVG_PC', 'APC_PCIxArea', 'PCC_SQFT', 'PCC_AVG_PC', 'P
for j in range(len(StartFields)):
    arcpy.AlterField_management(fc, StartFields[j], EndFields[j])
del j
```

- PCI cannot be summed

  → Needs to be recalculated from summed (PCIxArea)/Area



```
fcFields = ["AC_SQFT","AC_AVG_PCI","AC_PCIxArea","APC_SQFT","APC_AVG_PC","APC_PCIxArea","PCC_SQFT","PCC_AVG_PC","PCC_PCIxArea"]
#Replaces the incorrect SUMmed avg PCI values with the correct PCIxArea / Area value.
with arcpy.da.UpdateCursor (fc, fcFields) as cursor3:
    for row3 in cursor3:
        for i in [0,1,2]:
            if row3[(i*3)+2]>0: row3[(i*3)+1] = row3[(i*3)+2]/row3[i*3]
            cursor3.updateRow(row3)
        del i
del cursor3,row3
```

The PCI+ Model

Part II: Acquire Information for Scoring

Spatial analysis to determine additional benefits of road projects

Within the right-of-way (ROW) of each road segment:

- counts (collisions, traffic volume, ADA ramps)
- lengths (of pipe, sidewalk, etc.)
- percentage of areas (land use)

Typical Procedure

- Add new fields
- Select road features…one by one (Update Cursor)
- Point features: “SelectByLocation” features “WITHIN” selection
- Lines / polygons: “Clip” parts by selected road feature.
- Count the selections or clipped pieces
- If count > 0:
  - Sort through the collected features (Search Cursor)
  - Manipulate the data
- Store needed data in the new fields

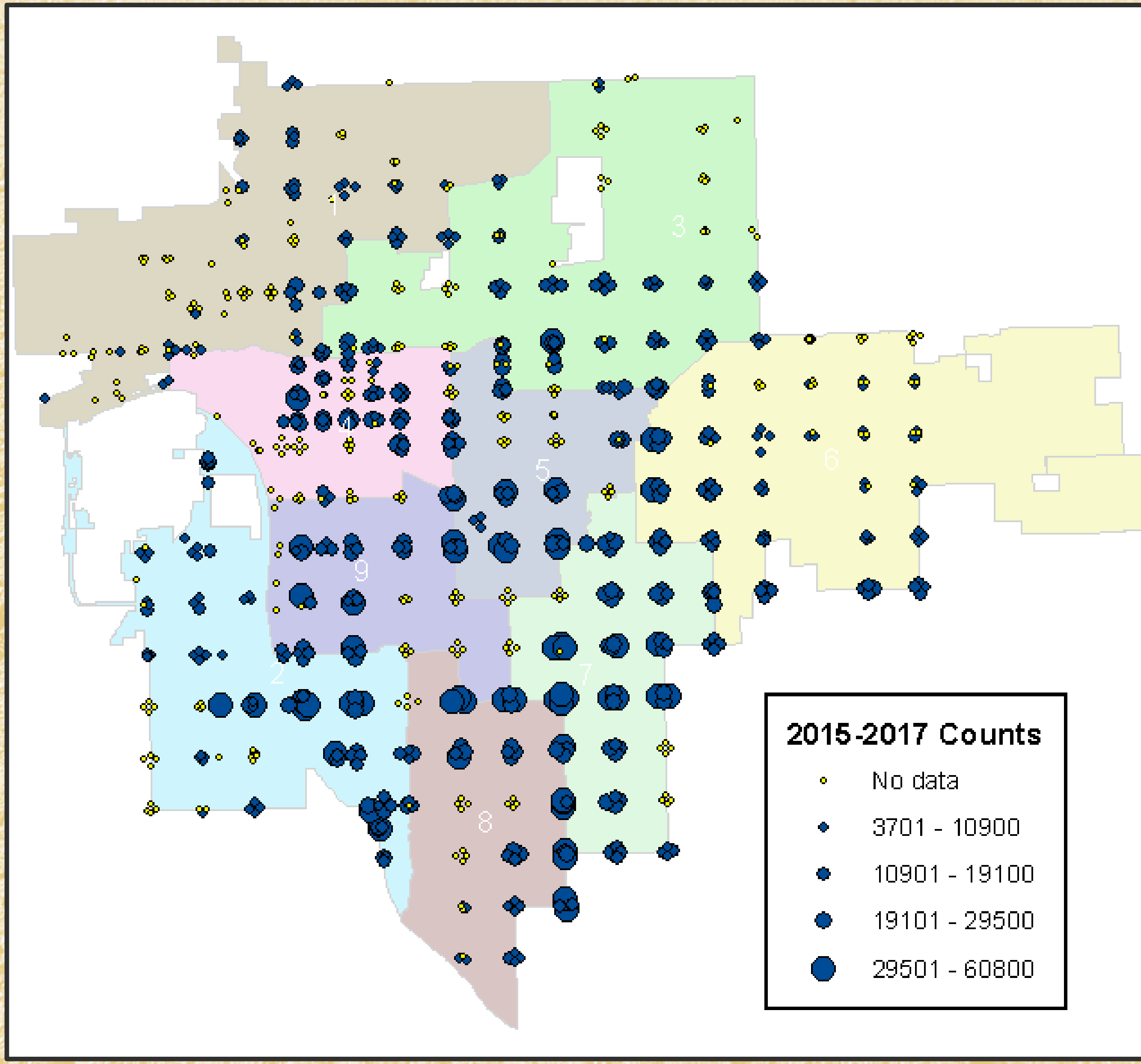
Average Daily Traffic

Higher Traffic Counts = more impact on the traveling public

For each road segment: Records the **Average** value of traffic counts in both:

- 2014
- 2015-2017

High count used for scoring



No data – typically construction

Collisions

Changes in a street's geometry could improve safety

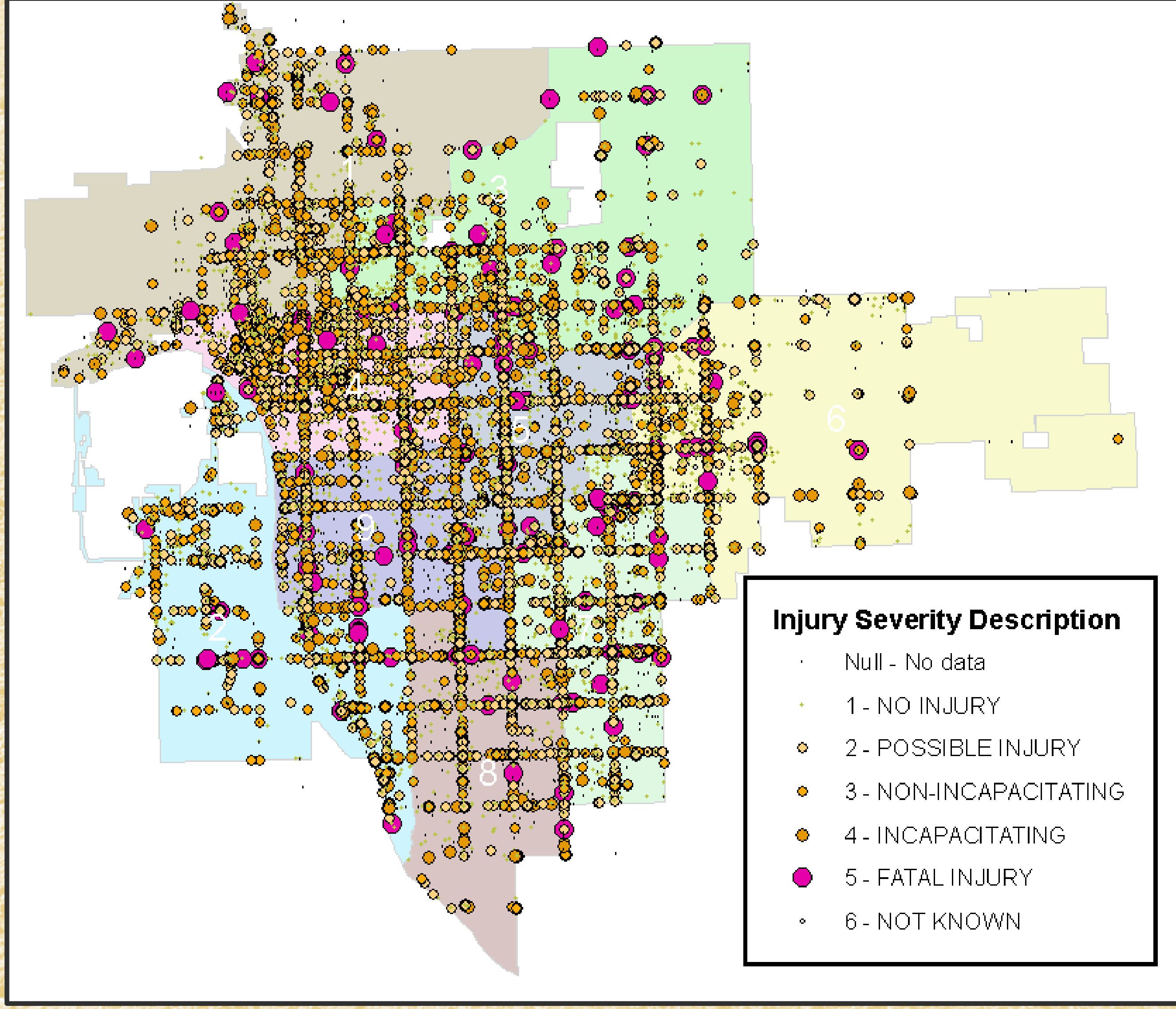
For each road segment: Counts only the **Injury** and **Fatal** accidents over 3 year period.

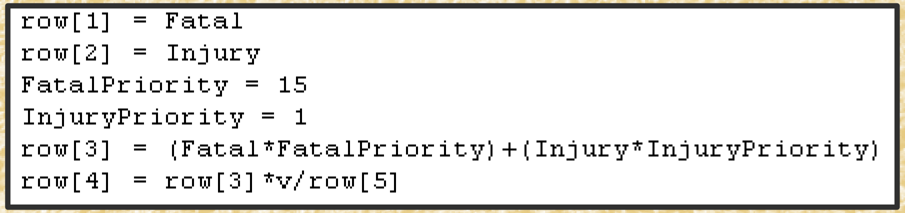
Fatal accidents weighted more heavily.

Normalize:

normal corridor length / Current segment length

Normal corridor length (v) = 1 mile – typical intersection





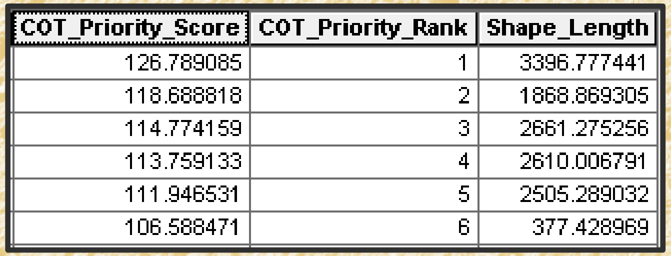
```
row[1]  = Fatal
row[2]  = Injury
FatalPriority = 15
InjuryPriority = 1
row[3]  = (Fatal*FatalPriority)+(Injury*InjuryPriority)
row[4]  = row[3]*v/row[5]
```

Sidewalk Gaps

Upgrade streets without sidewalks

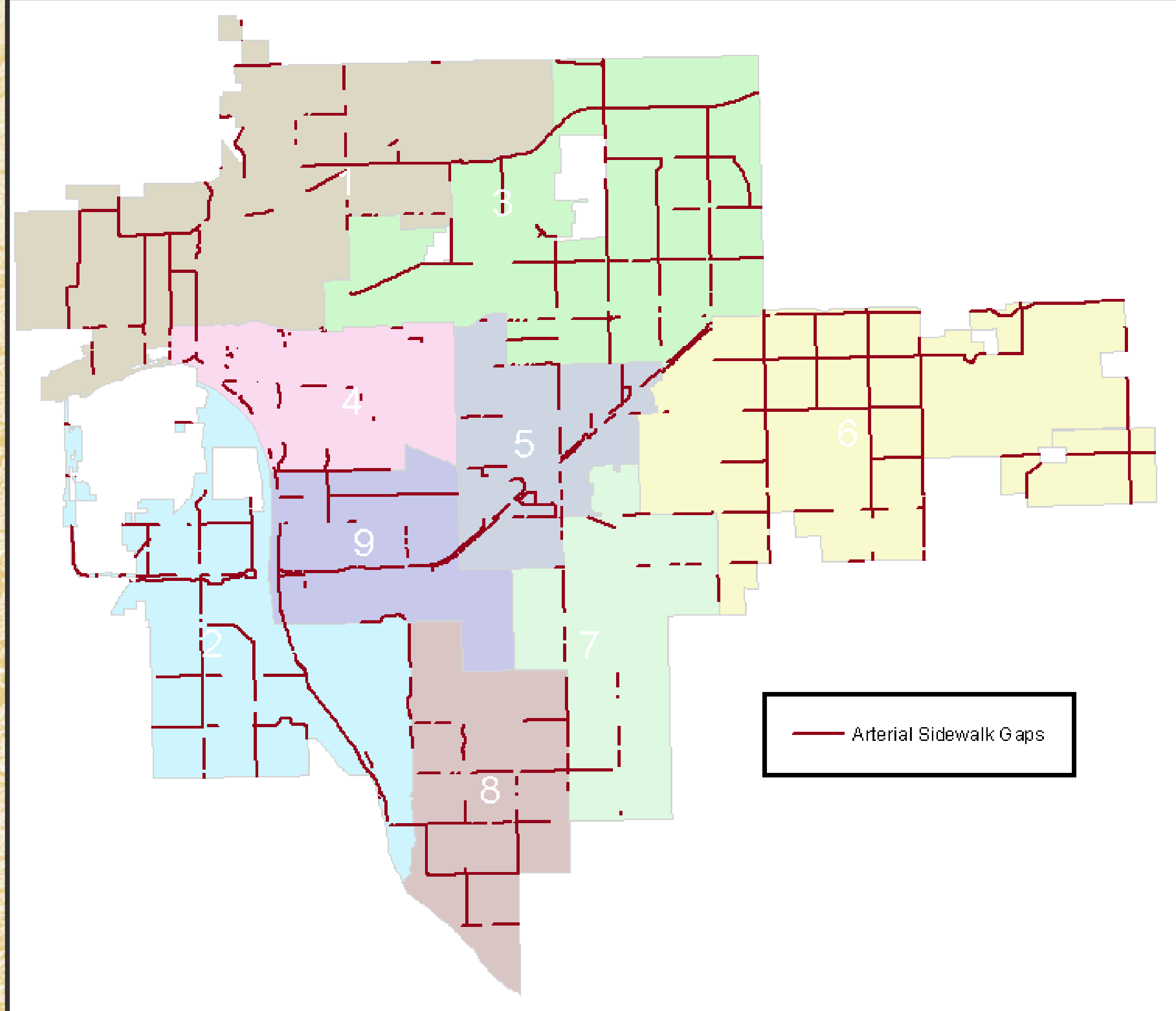
For each road segment: Find the total length of sidewalk gap in ROW using “Clip”.

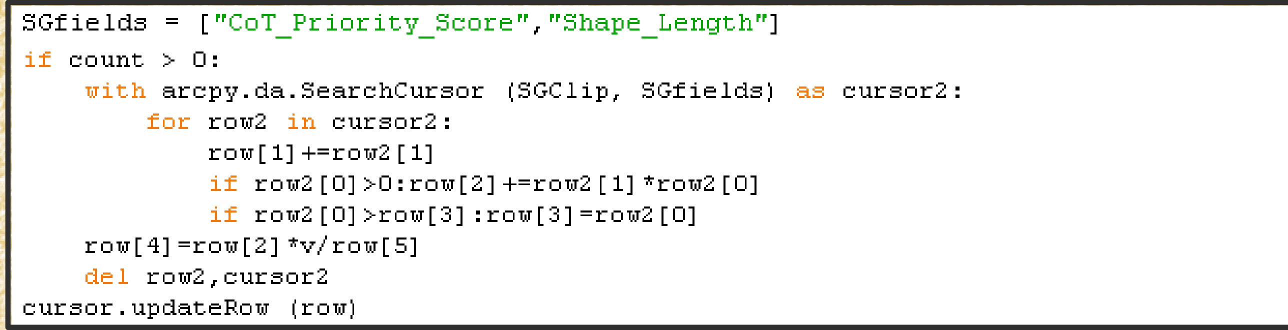
Store sum of length x priority (priority indicated in a feature class field, “COT_Priority_Score”)



COT_Priority_Score	COT_Priority_Rank	Shape_Length
126.789085	1	3396.777441
118.688818	2	1868.869305
114.774159	3	2661.275256
113.759133	4	2610.006791
111.946531	5	2505.289032
106.588471	6	377.428969

Normalize





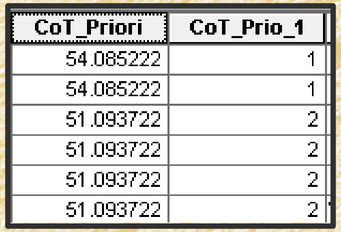
```
SGfields = ["CoT_Priority_Score","Shape_Length"]
if count > 0:
    with arcpy.da.SearchCursor (SGClip, SGfields) as cursor2:
        for row2 in cursor2:
            row[1]+=row2[1]
            if row2[0]>0:row[2]+=row2[1]*row2[0]
            if row2[0]>row[3]:row[3]=row2[0]
    row[4]=row[2]*v/row[5]
    del row2,cursor2
cursor.updateRow (row)
```

Go Plan

Opportunities to add or improve bicycle routes

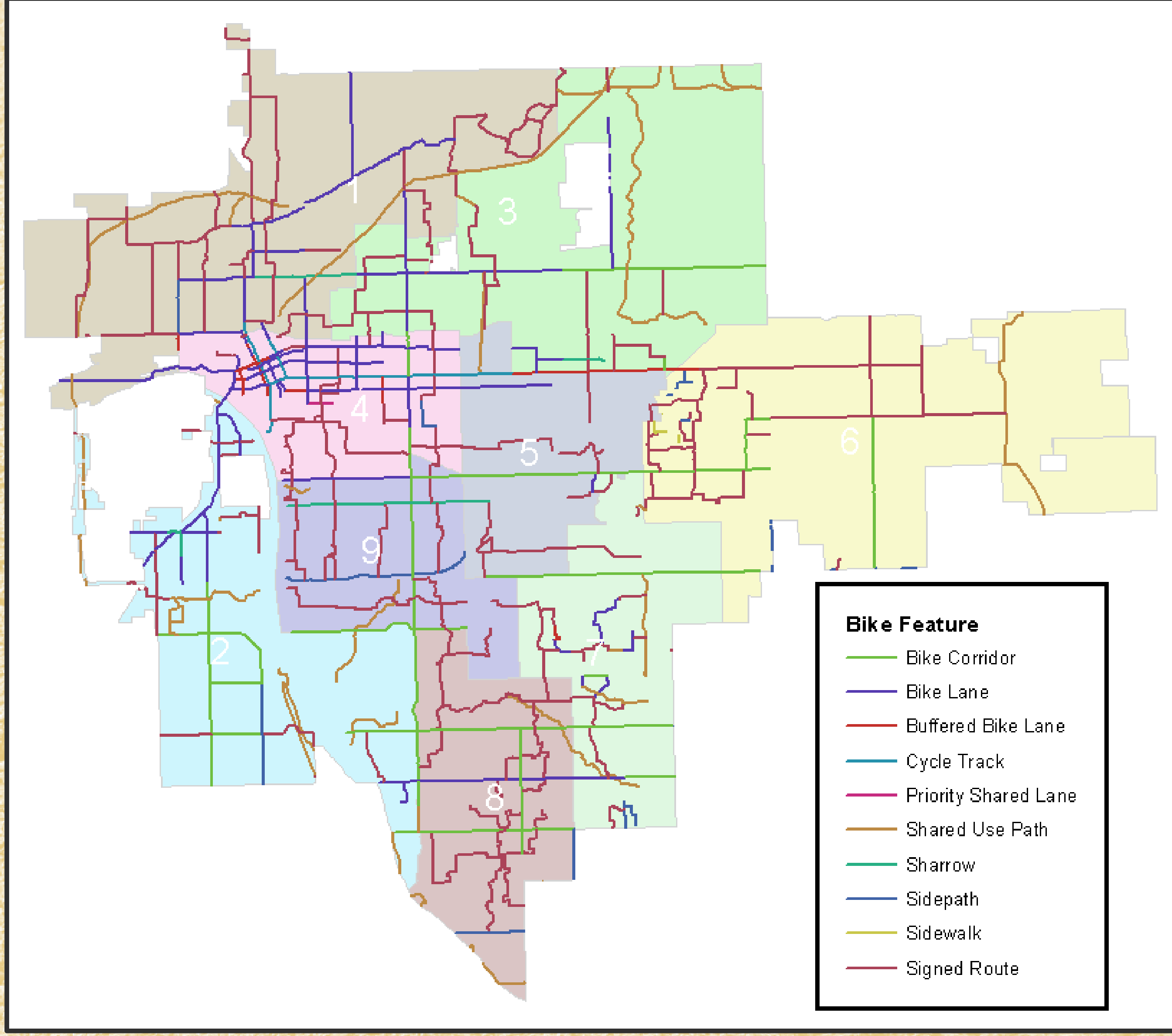
For each road segment:
Find the total length of trail in ROW using “Clip”.

Store sum of length x priority (priority indicated in a feature class field, “CoT_Priori”)



CoT_Priori	CoT_Prio_1
54.085222	1
54.085222	1
51.093722	2
51.093722	2
51.093722	2
51.093722	2

Normalize



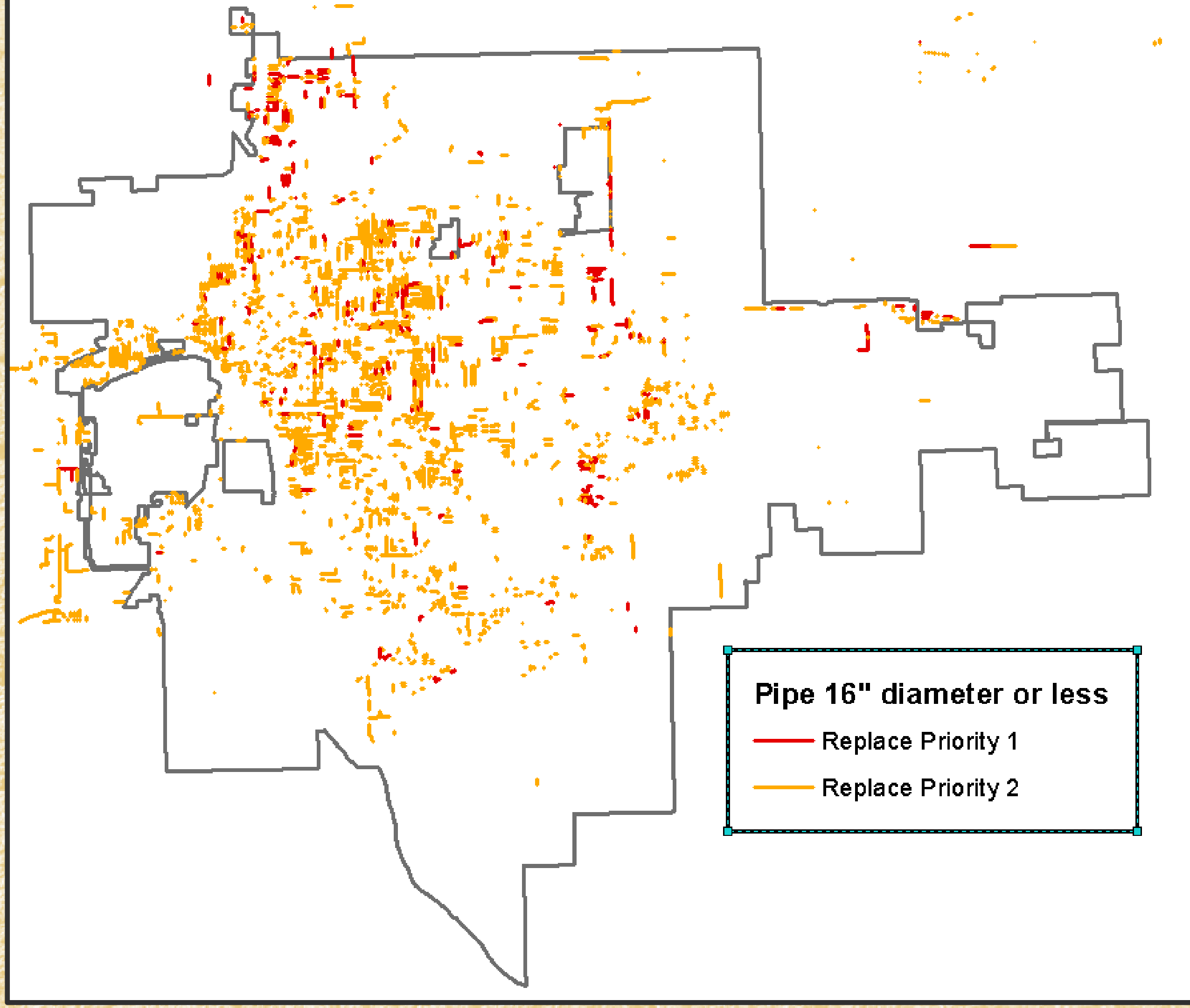
Water Pipe Replacement

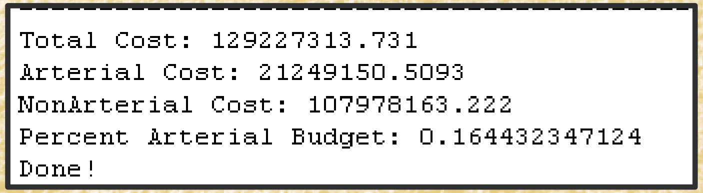
Replacing priority water lines would benefit residents and reduce cost

INFOMASTER program identifies pipe most likely to fail (5 priority levels).

Objective: Replace priority 1 & 2 cast iron pipe, 16 in diameter or smaller with set budget.

A script finds percentage in Arterial ROW → to set available budget for Arterial pipe replacement.





```
Total Cost: 129227313.731
Arterial Cost: 21249150.5093
NonArterial Cost: 107978163.222
Percent Arterial Budget: 0.164432347124
Done!
```

Water Pipe Replacement

Replacing priority water lines would benefit residents and reduce cost

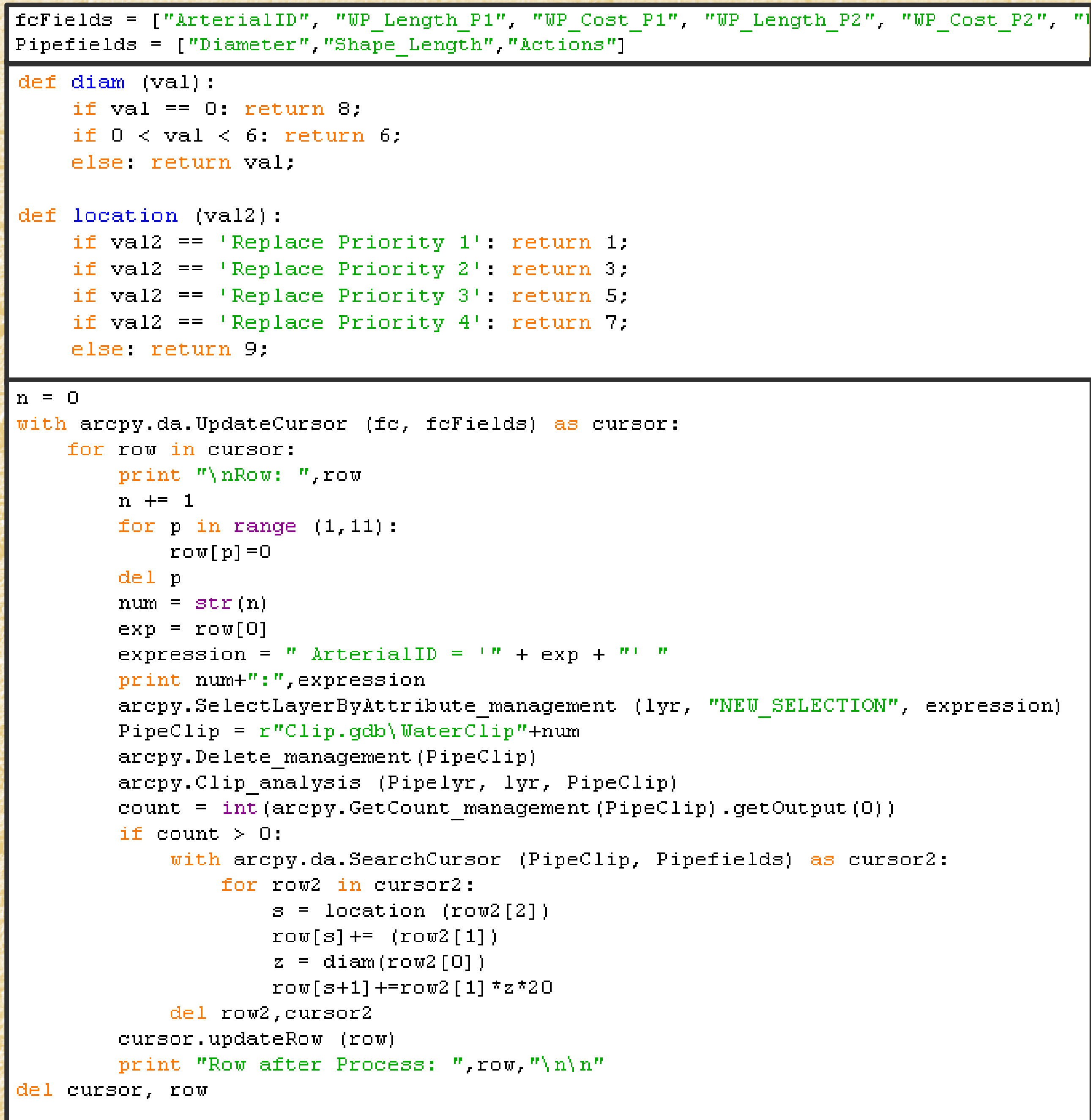
For each road segment:

- Clip these pipes by ROW
- Sum length by Priority (1-5)
- Estimate replacement cost.

Score

- Priority1 length x 3
- Priority2 length x 1
- Priority(3-5) length x 0

Normalize.



```
fcFields = ["ArterialID", "WP_Length_P1", "WP_Cost_P1", "WP_Length_P2", "WP_Cost_P2", "
Pipefields = ["Diameter","Shape_Length","Actions"]
```

```
def diam (val):
    if val == 0: return 8;
    if 0 < val < 6: return 6;
    else: return val;

def location (val2):
    if val2 == 'Replace Priority 1': return 1;
    if val2 == 'Replace Priority 2': return 3;
    if val2 == 'Replace Priority 3': return 5;
    if val2 == 'Replace Priority 4': return 7;
    else: return 9;
```

```
n = 0
with arcpy.da.UpdateCursor (fc, fcFields) as cursor:
    for row in cursor:
        print "\nRow: ",row
        n += 1
        for p in range (1,11):
            row[p]=0
        del p
        num = str(n)
        exp = row[0]
        expression = " ArterialID = '" + exp + "'  "
        print num+":",expression
        arcpy.SelectLayerByAttribute_management (lyr, "NEW_SELECTION", expression)
        PipeClip = r"Clip.gdb\WaterClip"+num
        arcpy.Delete_management(PipeClip)
        arcpy.Clip_analysis (Pipelyr, lyr, PipeClip)
        count = int(arcpy.GetCount_management(PipeClip).getOutput(0))
        if count > 0:
            with arcpy.da.SearchCursor (PipeClip, Pipefields) as cursor2:
                for row2 in cursor2:
                    s = location (row2[2])
                    row[s]+= (row2[1])
                    z = diam(row2[0])
                    row[s+1]+=row2[1]*z*20
            del row2,cursor2
        cursor.updateRow (row)
        print "Row after Process: ",row,"\n\n"
del cursor, row
```

ADA Accessibility

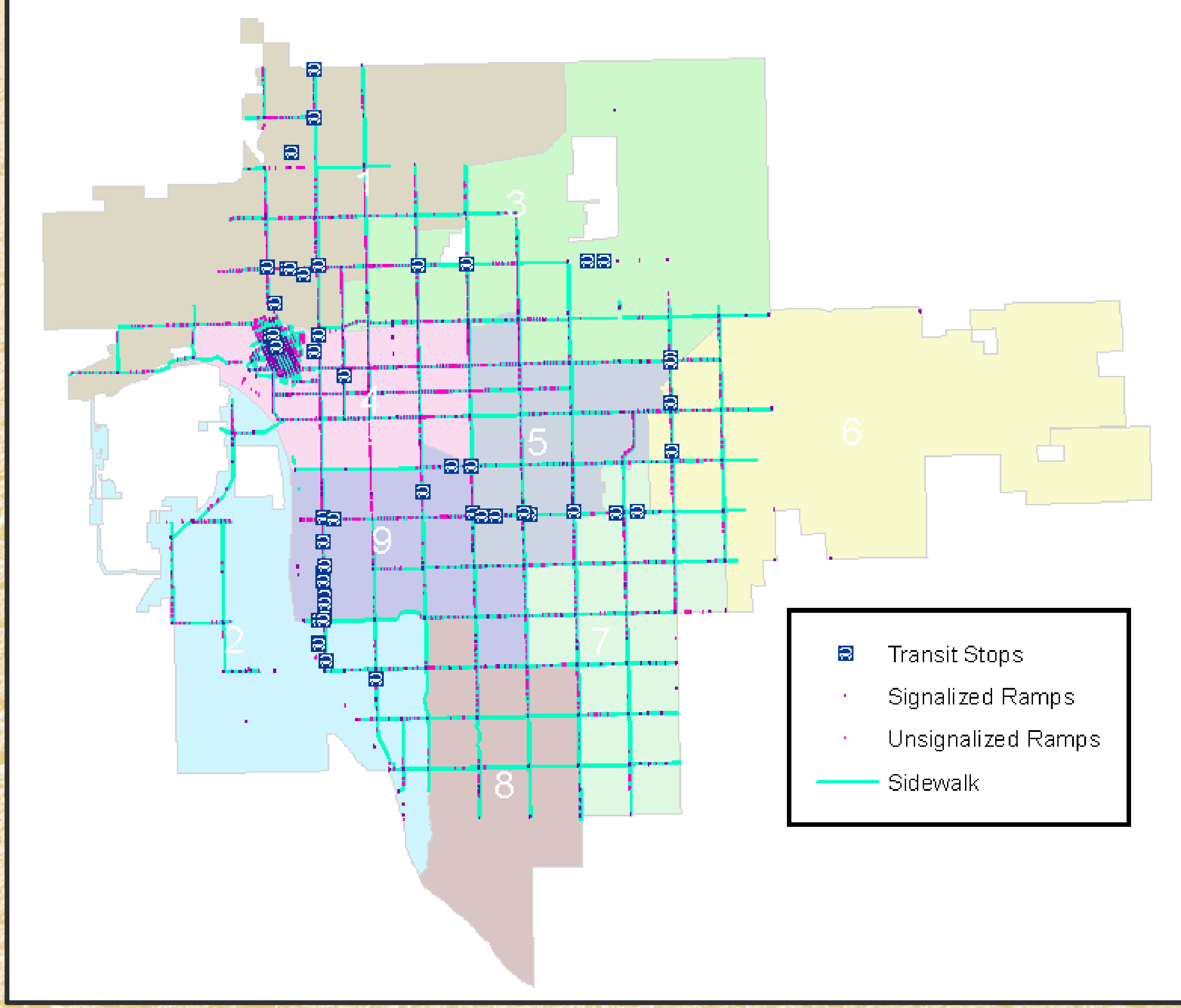
Areas without ADA Accessibility would be brought into compliance

Three components:

- Transit Stops
- Ramps
  - Signalized Ramps
  - Unsignalized Ramps
- Sidewalks

Three priorities:

- Low
- Medium
- High



ADA Accessibility

Areas without ADA Accessibility would be brought into compliance

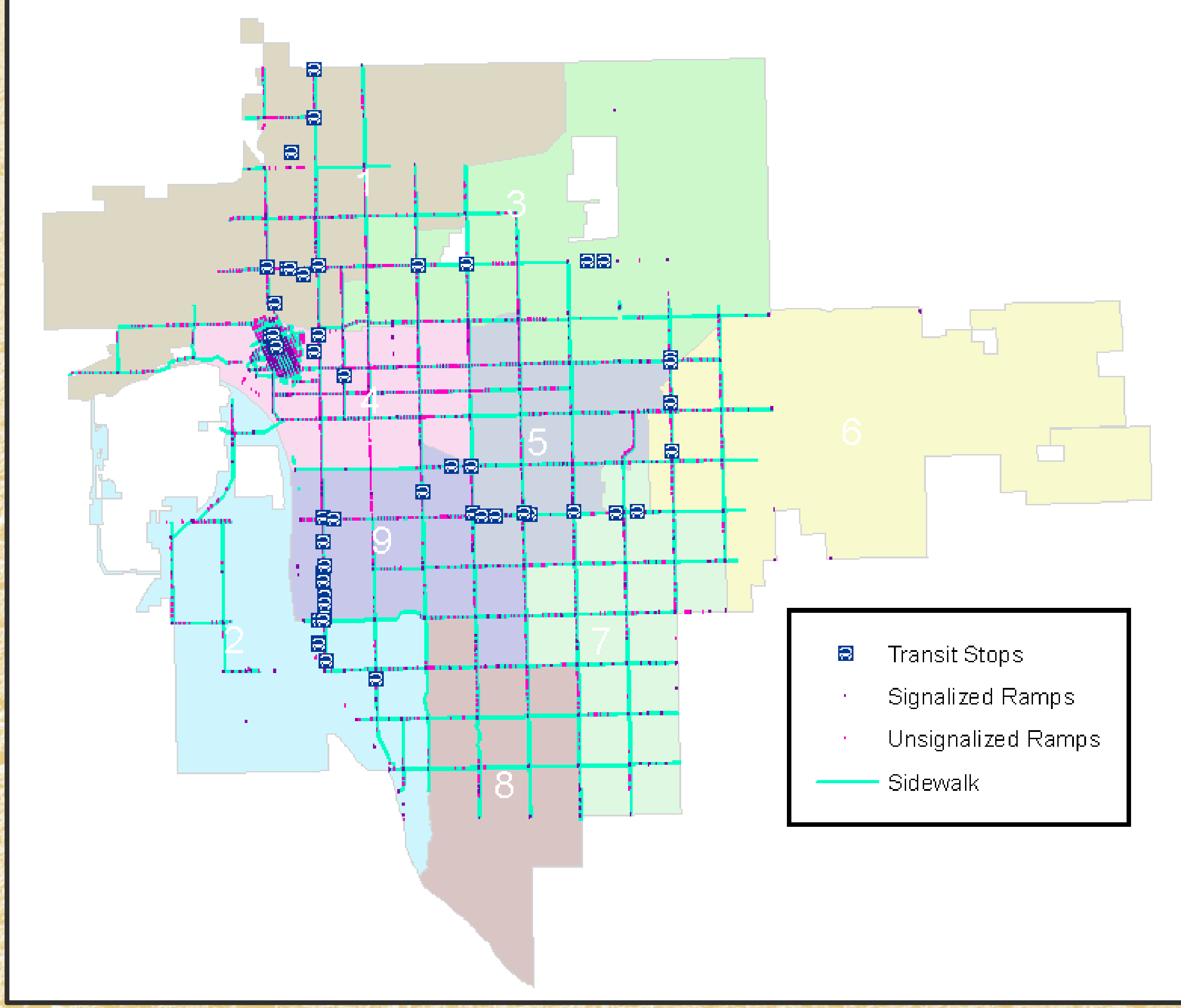
For each road segment: Sum features or lengths

Multiply by Priority score

- Low = 0
- Med = 2
- High = 3

Normalize

Total score: Transit 60%, Ramps 20%, Sidewalks 20%



Ramp Score:

```
row[7] = (UnsigL*0) + (UnsigM*2) + (UnsigH*3)+ (SigL*0) + (SigM*2) + (SigH*3)
```

Land Use (LU)

Improving areas with certain land uses could lead to population growth

For each road segment: "Clip" the LU polygons in the ROW.

Store values in list by LU type.

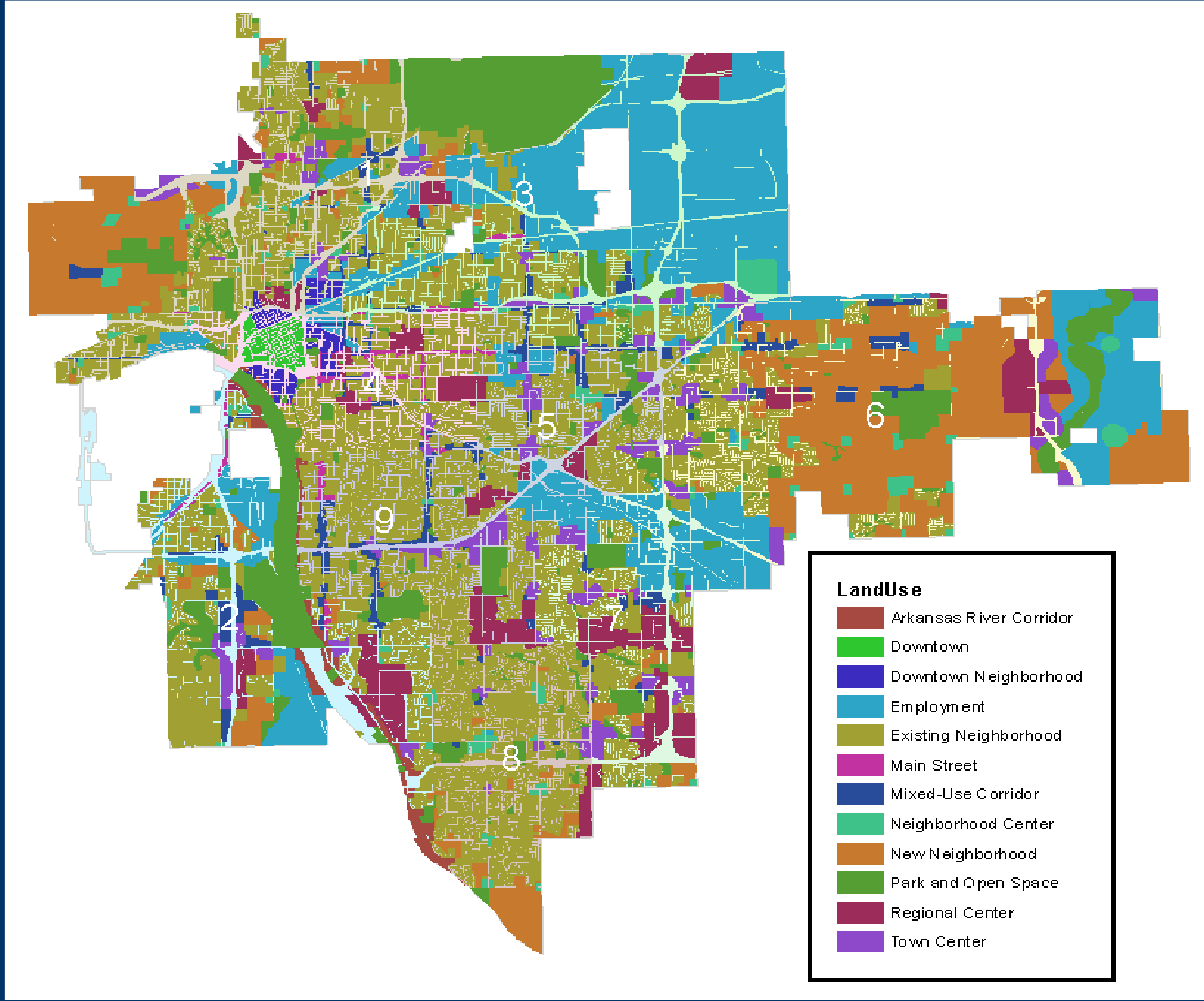
- 0 position – Total area
- 1-12 – LU area by category

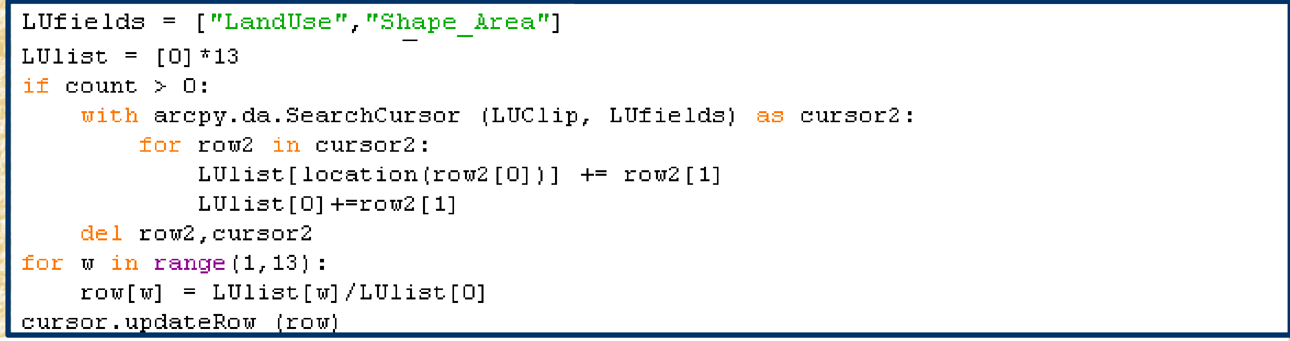
Then, store for each LU category:

- LU area / Total area (from list)

Score with weights by LU type:

- Downtown / Main Street: 20
- New Neighborhood: 0 etc…




```
LUfields = ["LandUse","Shape_Area"]
LUlist = [0]*13
if count > 0:
    with arcpy.da.SearchCursor (LUClip, LUfields) as cursor2:
        for row2 in cursor2:
            LUlist[location(row2[0])] += row2[1]
            LUlist[0]+=row2[1]
    del row2,cursor2
for w in range(1,13):
    row[w] = LUlist[w]/LUlist[0]
cursor.updateRow (row)
```

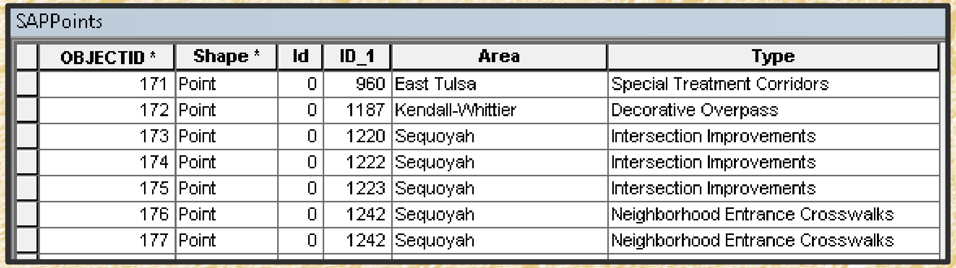
```
fcFields = ["ArterialID", "LU_Ark", "LU_DnTn", "LU_DnTnHood", "LU_Employ", "LU_ExHood", "LU_MainSt", "LU_MixUse", "LU_NeighCtr", "LU_NewHood", "LU_ParkOpen", "LU_RegCtr", "LU_TnCtr"]
```

Small Area Plans

Recommended upgrades can be implemented

Planning Department gave us:

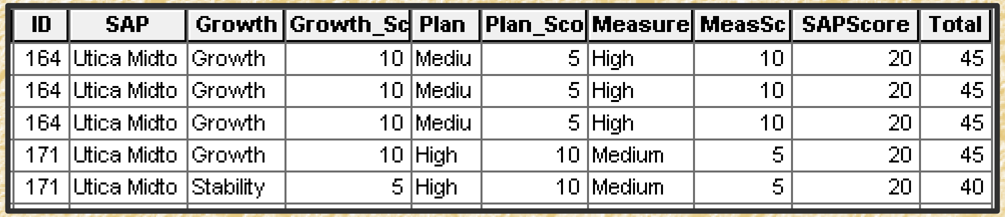
- 2 feature classes
  - SAP Points
  - SAP Lines



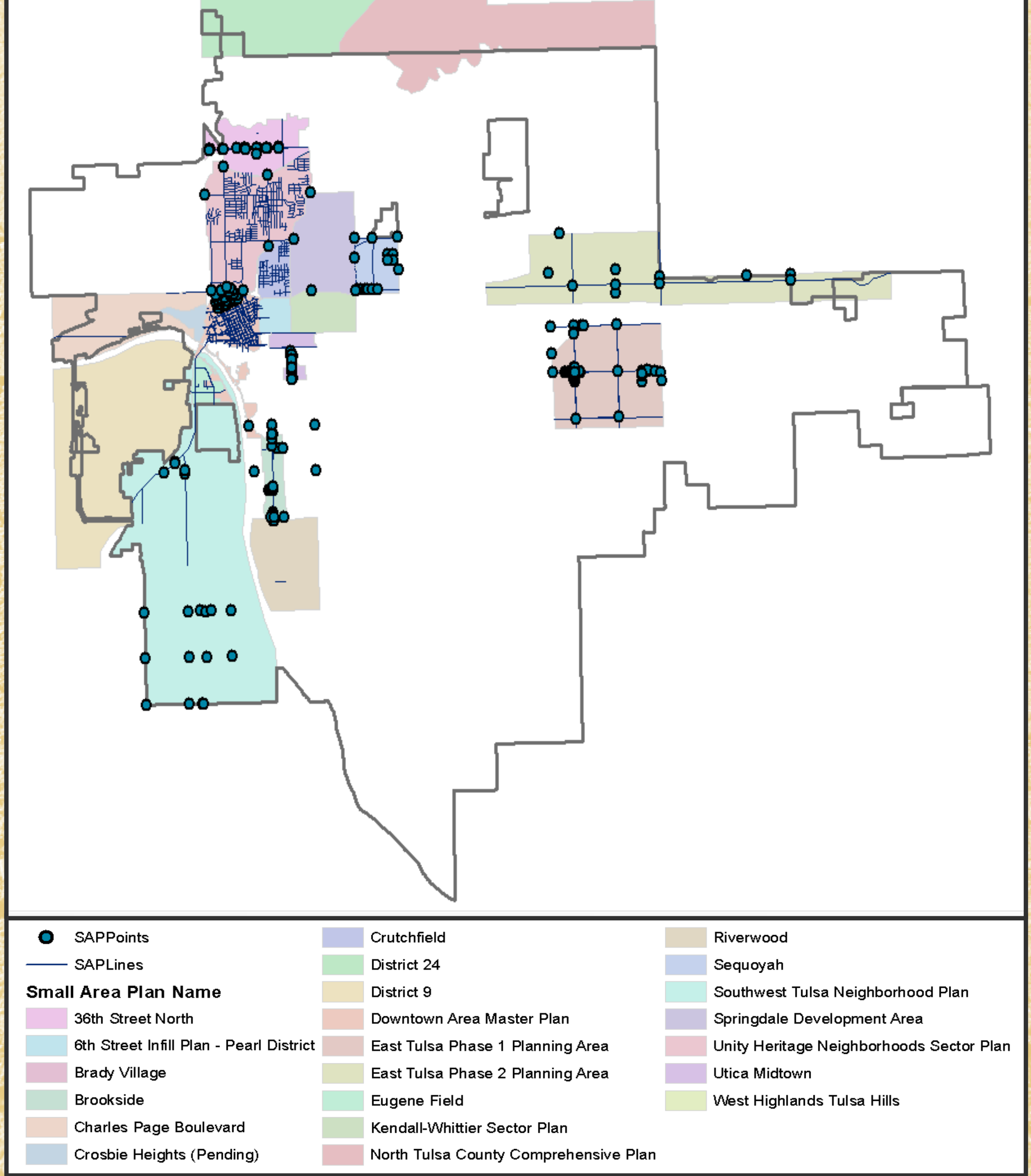
SAPPoints

OBJECTID *	Shape *	Id	ID_1	Area	Type
171	Point	0	960	East Tulsa	Special Treatment Corridors
172	Point	0	1187	Kendall-Whittier	Decorative Overpass
173	Point	0	1220	Sequoyah	Intersection Improvements
174	Point	0	1222	Sequoyah	Intersection Improvements
175	Point	0	1223	Sequoyah	Intersection Improvements
176	Point	0	1242	Sequoyah	Neighborhood Entrance Crosswalks
177	Point	0	1242	Sequoyah	Neighborhood Entrance Crosswalks

- Spreadsheet with scores
  - “ID_1” linked to scores



ID	SAP	Growth	Growth_Sc	Plan	Plan_Sco	Measure	MeasSc	SAPScore	Total
164	Utica Midto	Growth	10	Mediu	5	High	10	20	45
164	Utica Midto	Growth	10	Mediu	5	High	10	20	45
164	Utica Midto	Growth	10	Mediu	5	High	10	20	45
171	Utica Midto	Growth	10	High	10	Medium	5	20	45
171	Utica Midto	Stability	5	High	10	Medium	5	20	40


SAPPoints
SAPLines
Small Area Plan Name
36th Street North
6th Street Infill Plan - Pearl District
Brady Village
Brookside
Charles Page Boulevard
Crosbie Heights (Pending)
Crutchfield
District 24
District 9
Downtown Area Master Plan
East Tulsa Phase 1 Planning Area
East Tulsa Phase 2 Planning Area
Eugene Field
Kendall-Whittier Sector Plan
North Tulsa County Comprehensive Plan
Riverwood
Sequoyah
Southwest Tulsa Neighborhood Plan
Springdale Development Area
Unity Heritage Neighborhoods Sector Plan
Utica Midtown
West Highlands Tulsa Hills

Small Area Plans

Recommended upgrades can be implemented

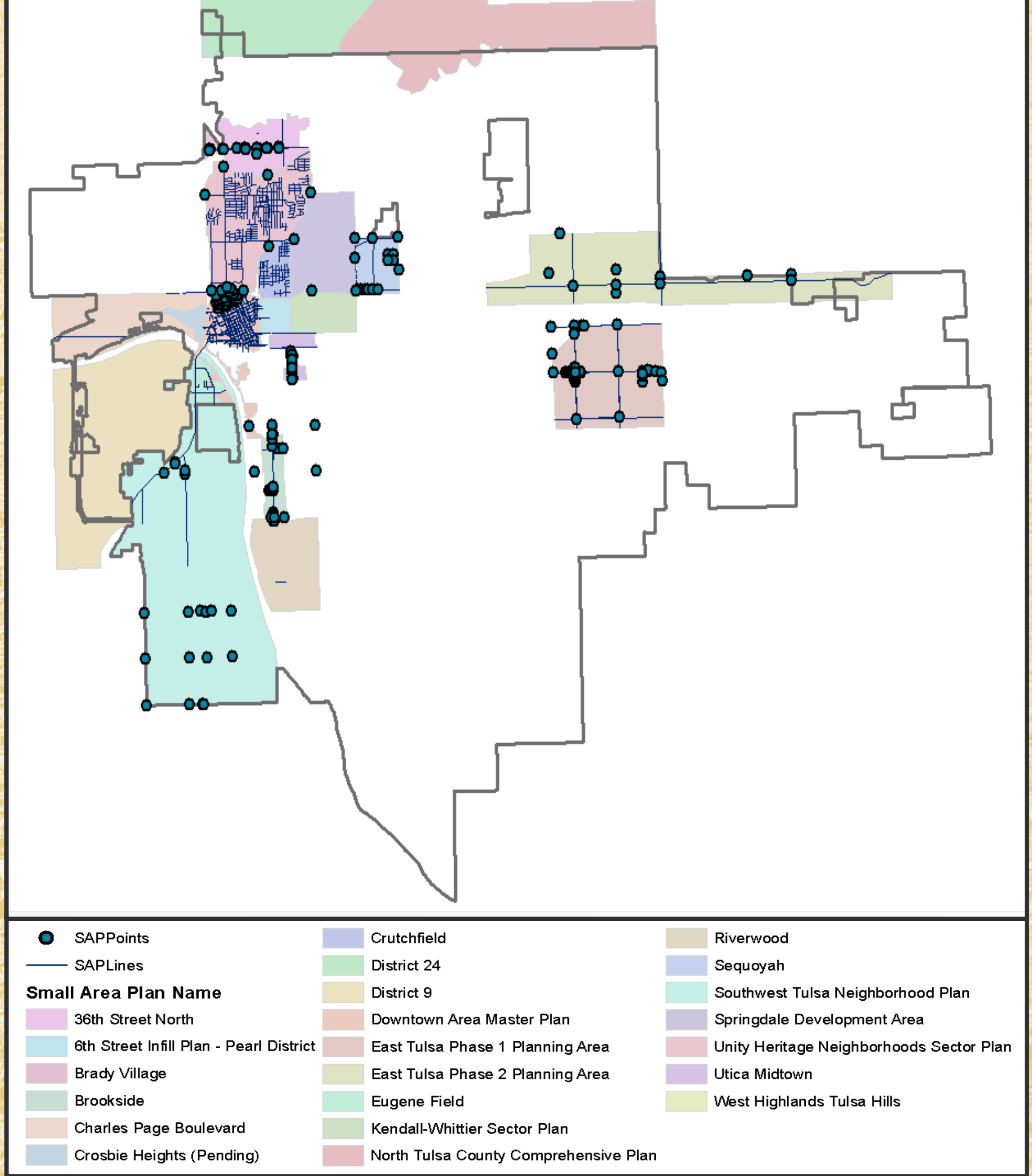
Read ID-Score key-value pairs into a Python dictionary

```
with arcpy.da.SearchCursor (PlTable, TableFields) as cursor5:
    for row5 in cursor5:
        SAPdict[row5[0]]= row5[1]-20
del row5,cursor5
```

For each road segment:

- Find intersecting features
- Use ID as “key” to return “value” as they are summed

```
arcpy.SelectLayerByLocation_management (ptlyr, "INTERSECT", lyr)
count2 = int(arcpy.GetCount_management(ptlyr).getOutput(0))
if count2 > 0:
    with arcpy.da.SearchCursor (ptlyr, SAPfields) as cursor2:
        for row2 in cursor2:
            row[-2] += SAPdict[row2[0]]
    del row2,cursor2
```

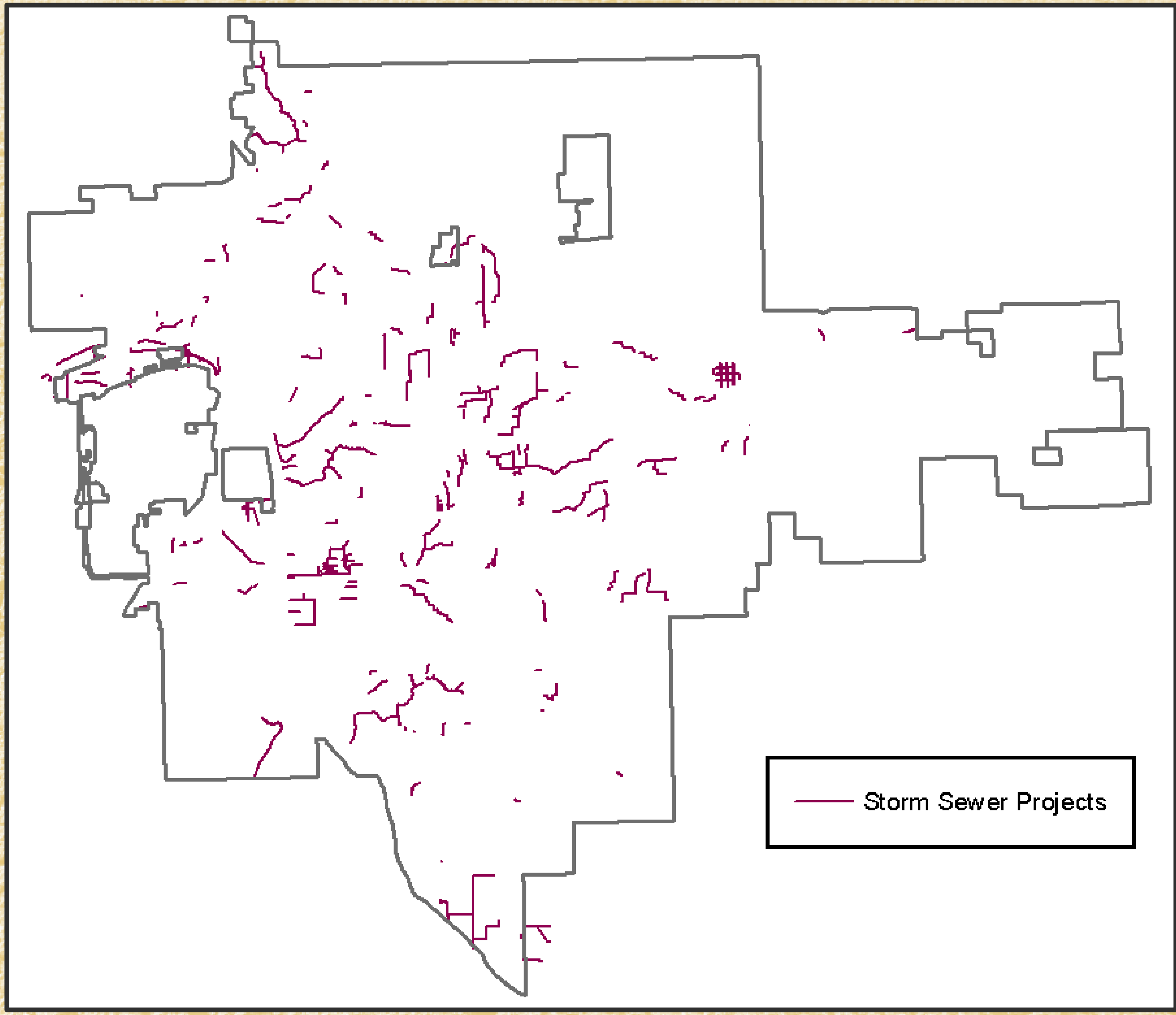

● SAPPoints
SAPLines
Small Area Plan Name
36th Street North
6th Street Infill Plan - Pearl District
Brady Village
Brookside
Charles Page Boulevard
Crosbie Heights (Pending)
Crutchfield
District 24
District 9
Downtown Area Master Plan
East Tulsa Phase 1 Planning Area
East Tulsa Phase 2 Planning Area
Eugene Field
Kendall-Whittier Sector Plan
North Tulsa County Comprehensive Plan
Riverwood
Sequoyah
Southwest Tulsa Neighborhood Plan
Springdale Development Area
Unity Heritage Neighborhoods Sector Plan
Utica Midtown
West Highlands Tulsa Hills

Master Drainage Plan

Negative score because funds are not available to make all improvements

For each road segment:

- Tally all features that intersect ROW
- Negative score for any intersecting segment (-200)
  - Not out of running…just lower priority.
  - Projects typically run way beyond the roads.



The PCI+ Model

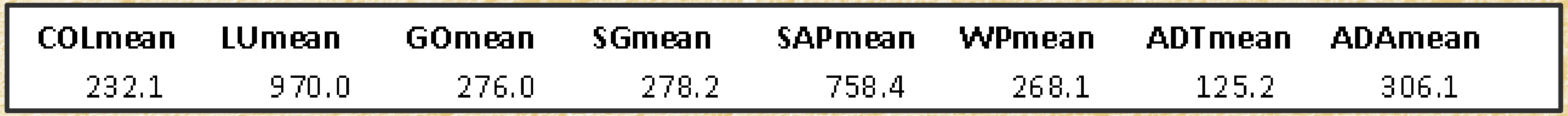
Part III: Scoring Roads

A script adds raw scores for each category

→ weights within categories satisfied

(f.ex. - P1 pipe more heavily than P2 pipes)

BUT, total points per category → vastly different (pipes vs. SAP)



COLmean	LUmean	GOmean	SGmean	SAPmean	WPmean	ADTmean	ADAmean
232.1	970.0	276.0	278.2	758.4	268.1	125.2	306.1

Do not want to sum or weight unequal categories.

To compare categories fairly, the scores need to be curved……

Curving Road Scores

Used a linear curve.

$$f(x) = y_0 + \left(\frac{y_1 - y_0}{x_1 - x_0}\right)(x - x_0).$$

(x_1,y_1) is (0.0000000000000001, 0.0000000000000001) to avoid divide by zero with (0,0)

(x_0,y_0) is (current category mean, maximum category mean → LUmean)

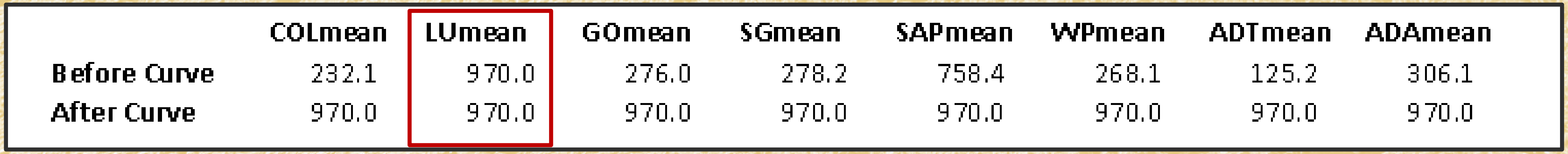
x is the uncurved score in question.

Interpretation: Zero is still zero*, adjust mean to the highest mean of all categories

→ All the other scores are adjusted linearly.

→ No changes to the category with the maximum mean.

→ All categories end up with the same mean value (same number of total points)



	COLmean	LUmean	GOmean	SGmean	SAPmean	WPmean	ADTmean	ADAmean
Before Curve	232.1	970.0	276.0	278.2	758.4	268.1	125.2	306.1
After Curve	970.0	970.0	970.0	970.0	970.0	970.0	970.0	970.0

*The script actually slightly adjusts incoming zero scores to be zero. Again, because of divide by zero issue.

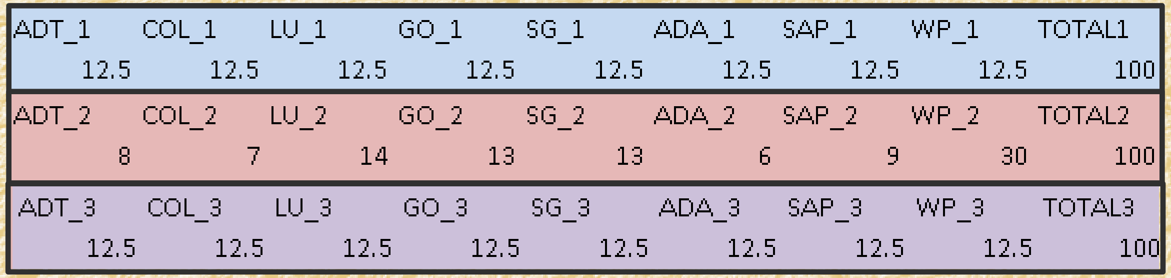
Weighting Road Scores by Category / District

Each Category can be weighted differently in each district

…potentially up to administrative officials

Traffic Counts	Traffic Crashes	GO Plan Facilities	Sidewalk Gaps	Land Use	Small Area Plans	Water Pipes	Disabilities Act
20%	15%	10%	20%	5%	10%	10%	10%

Input Excel file with variable scoring per district:



District 1:

ADT_1	COL_1	LU_1	GO_1	SG_1	ADA_1	SAP_1	WP_1	TOTAL1
12.5	12.5	12.5	12.5	12.5	12.5	12.5	12.5	100

District 2:

ADT_2	COL_2	LU_2	GO_2	SG_2	ADA_2	SAP_2	WP_2	TOTAL2
8	7	14	13	13	6	9	30	100

etc…

ADT_3	COL_3	LU_3	GO_3	SG_3	ADA_3	SAP_3	WP_3	TOTAL3
12.5	12.5	12.5	12.5	12.5	12.5	12.5	12.5	100

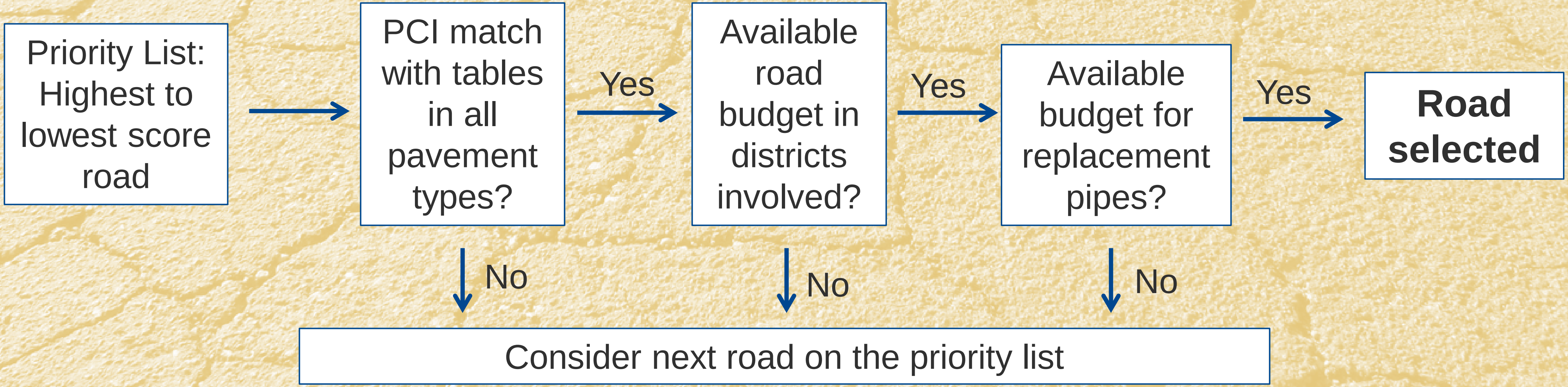
In the end → Balanced all categories in all districts.

→ Summed 12.5% of scores from each category.

The PCI+ Model

Part IV: Selecting Roads

Final Script: Roads selected in rank order if they meet all the necessary criteria.



Priority List: Highest to lowest score road

PCI match with tables in all pavement types?

Yes

Available road budget in districts involved?

Yes

Available budget for replacement pipes?

Yes

Road selected

No

No

No

Consider next road on the priority list

The selection script has ability to:

- Adjust PCI value range suitable for match with table data (used +/- 1).
- Adjust the budget wiggle room for each district (used 1-2%).
- Consider pavement type (AC, PCC, APC) in particular order (to improve model performance).
- Control the number and method of iterations through PCI tables to find a match.
- Pre-select roads (give certain roads first crack at the tables)
- Remove roads (keep out roads deemed unsuitable)

Two Search Methods:

First Pass selecting roads:

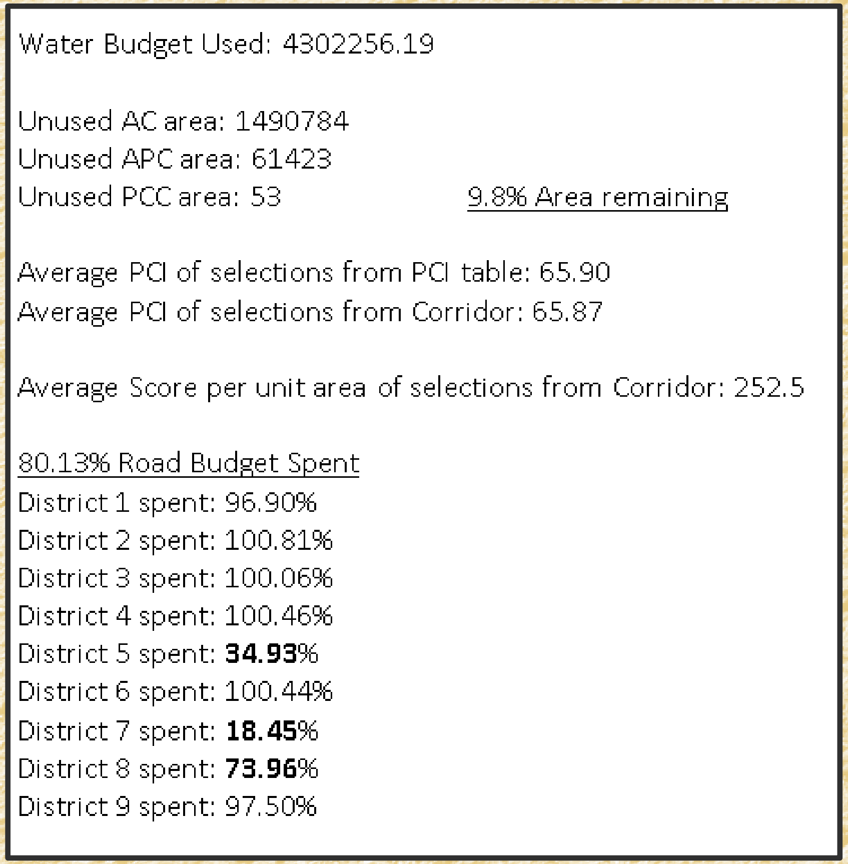
- Searches through 3 PCI tables line by line (AC, APC, PCC)
  - accumulating PCI values to match those in the road feature class
- Carefully matches PCI values, keeping it tight (+/- 1)
- Complex searching mechanism…does not give up easily
  - Repeats search through PCI tables in different ways for each road
- Typically 80% area or more selected

Remaining Passes:

- Calculates remaining Area and PCIxArea in PCI tables.
- Recalculates after every selection
- Allows greater and greater freedom with each pass
- If type close to empty → may drive remaining PCI outside 0-100 range

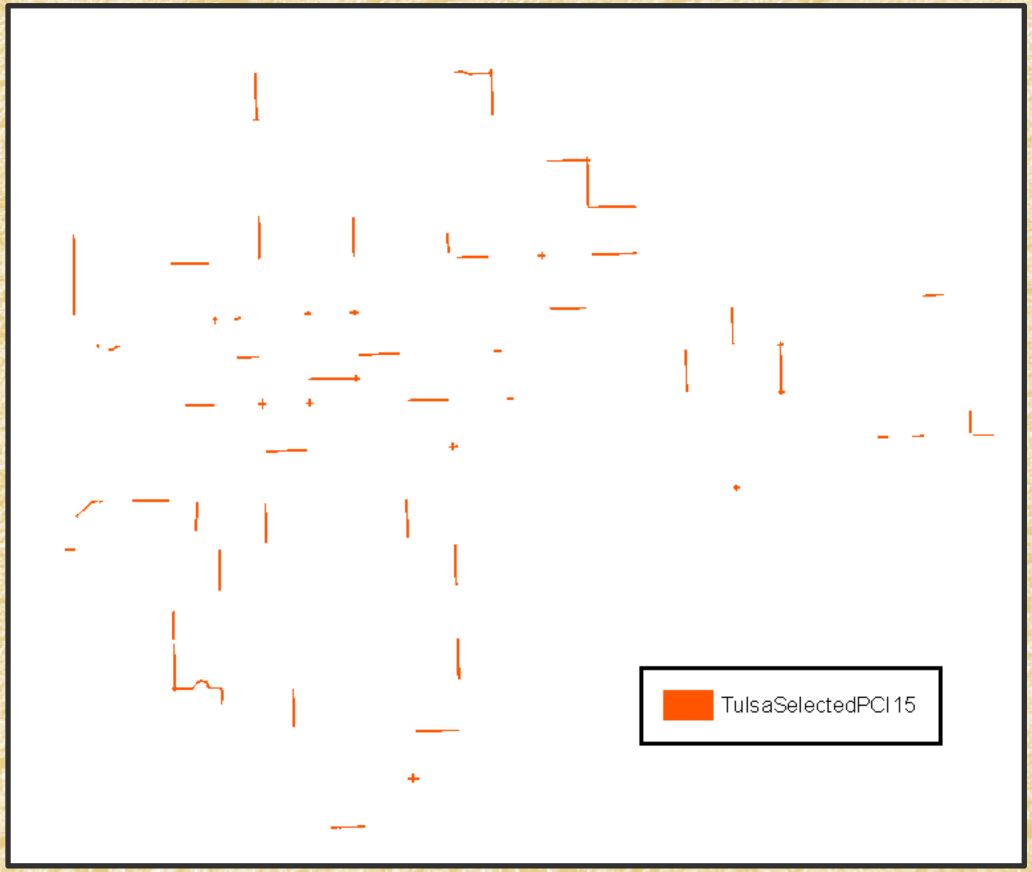
The selection script produces:

- Feature class with model picks
- Summary info: 9.8% area unused, 20% budget remains



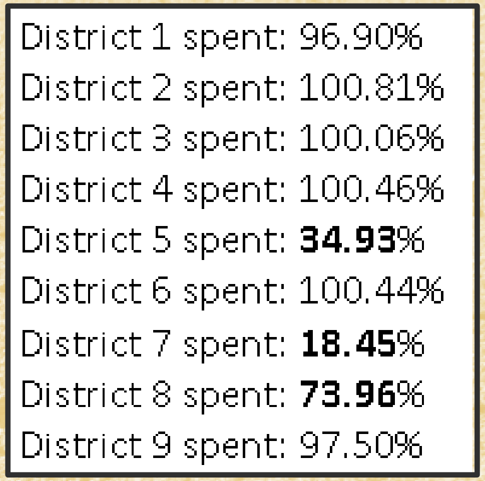
```
Water Budget Used: 4302256.19

Unused AC area: 1490784
Unused APC area: 61423
Unused PCC area: 53                9.8% Area remaining

Average PCI of selections from PCI table: 65.90
Average PCI of selections from Corridor: 65.87

Average Score per unit area of selections from Corridor: 252.5

80.13% Road Budget Spent
District 1 spent: 96.90%
District 2 spent: 100.81%
District 3 spent: 100.06%
District 4 spent: 100.46%
District 5 spent: 34.93%
District 6 spent: 100.44%
District 7 spent: 18.45%
District 8 spent: 73.96%
District 9 spent: 97.50%
```

Actual Model Results



OLD Model Test Run – (Scores not accurate)

Problem Solving



District 1 spent: 96.90%

District 2 spent: 100.81%

District 3 spent: 100.06%

District 4 spent: 100.46%

District 5 spent: **34.93**%

District 6 spent: 100.44%

District 7 spent: **18.45**%

District 8 spent: **73.96**%

District 9 spent: 97.50%

Three districts w/ unfilled obligations

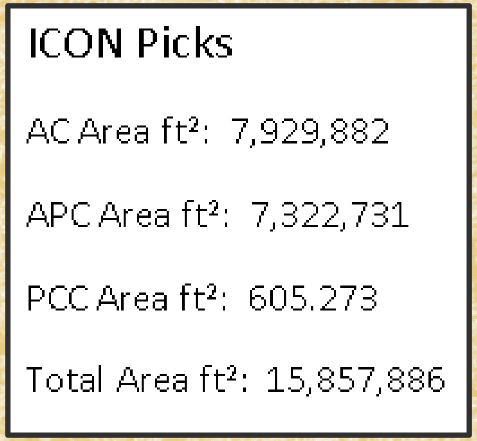
Analysis:

- Initial picks from consultant: only **3.8% PCC** by area.
- Problem districts (5,7,8): most of the need is in PCC type.

→ Ran out of PCC in tables (with higher scoring roads) before the script reached roads in these districts

Solution: Pre-select PCC roads in districts 5 & 7.

→ Very limited choices (and expensive roads)



ICON Picks

AC Area ft^2: 7,929,882

APC Area ft^2: 7,322,731

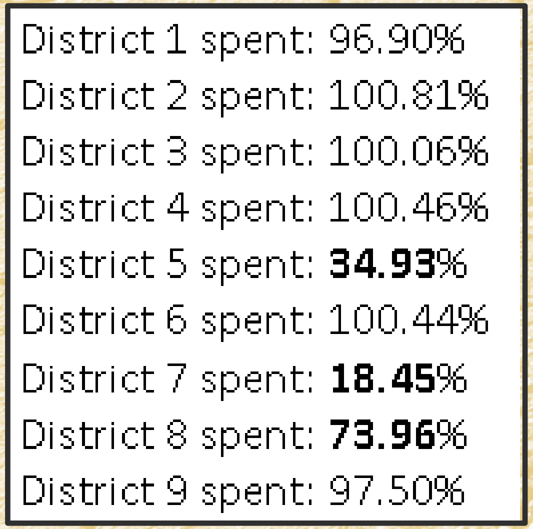
PCC Area ft^2: 605.273

Total Area ft^2: 15,857,886

Pre-select Run Results

8.2% area unused. 10% budget remains.

w/o Pre-select



District 1 spent: 96.90%

District 2 spent: 100.81%

District 3 spent: 100.06%

District 4 spent: 100.46%

District 5 spent: **34.93**%

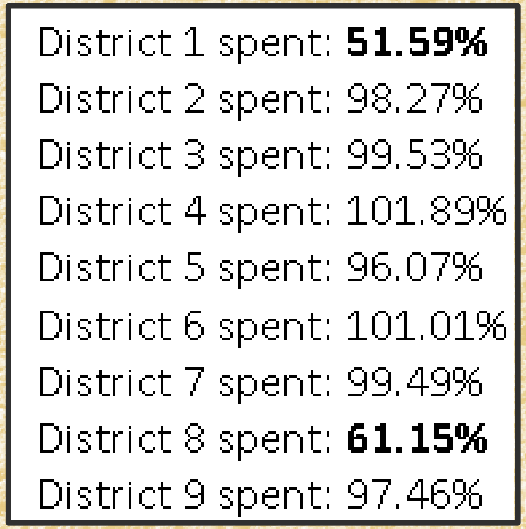
District 6 spent: 100.44%

District 7 spent: **18.45**%

District 8 spent: **73.96**%

District 9 spent: 97.50%

with Pre-select



District 1 spent: **51.59%**

District 2 spent: 98.27%

District 3 spent: 99.53%

District 4 spent: 101.89%

District 5 spent: 96.07%

District 6 spent: 101.01%

District 7 spent: 99.49%

District 8 spent: **61.15%**

District 9 spent: 97.46%

From 3 to 2 problem districts….

Districts 1 & 8 (Both districts dropped)

Main Impact – District 1

97% area used → 52% area used

Looked at what roads were eliminated from District 1 because PCC not available.

- Typically APC roads with small portions of PCC
- High scoring roads (desirable)

Were able to add desirable roads back in with flexible post-model selection method (remaining 8% of area):

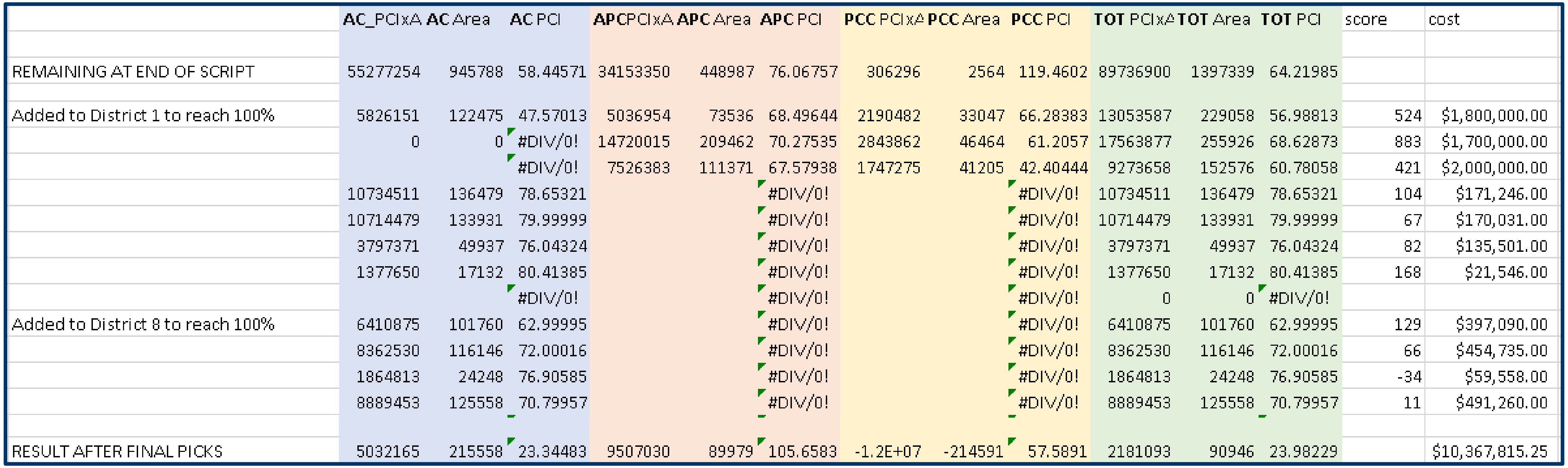
- An extra ~33% PCC area was added
- Focus on cost and overall PCI values

Post-Selection (by Hand) Results

Used a spreadsheet to add/subtract roads from Districts 1 & 8:

With the final 8% of area…could substitute pavement types and select “out-of-range” PCI values…if they:

- Meet the budget needs
- Ideally have equal or lower overall PCI values than the remainders.
  - → Looked for highest scores / lowest PCI values that were affordable with available budget



	AC_PCIxA	AC Area	AC PCI	APCPCIxA	APC Area	APC PCI	PCC PCIxA	PCC Area	PCC PCI	TOT PCIxA	TOT Area	TOT PCI	score	cost
REMAINING AT END OF SCRIPT	55277254	945788	58.44571	34153350	448987	76.06757	306296	2564	119.4602	89736900	1397339	64.21985		
Added to District 1 to reach 100%	5826151	122475	47.57013	5036954	73536	68.49644	2190482	33047	66.28383	13053587	229058	56.98813	524	$1,800,000.00
	0	0	#DIV/0!	14720015	209462	70.27535	2843862	46464	61.2057	17563877	255926	68.62873	883	$1,700,000.00
			#DIV/0!	7526383	111371	67.57938	1747275	41205	42.40444	9273658	152576	60.78058	421	$2,000,000.00
	10734511	136479	78.65321			#DIV/0!			#DIV/0!	10734511	136479	78.65321	104	$171,246.00
	10714479	133931	79.99999			#DIV/0!			#DIV/0!	10714479	133931	79.99999	67	$170,031.00
	3797371	49937	76.04324			#DIV/0!			#DIV/0!	3797371	49937	76.04324	82	$135,501.00
	1377650	17132	80.41385			#DIV/0!			#DIV/0!	1377650	17132	80.41385	168	$21,546.00
			#DIV/0!			#DIV/0!			#DIV/0!	0	0	#DIV/0!		
Added to District 8 to reach 100%	6410875	101760	62.99995			#DIV/0!			#DIV/0!	6410875	101760	62.99995	129	$397,090.00
	8362530	116146	72.00016			#DIV/0!			#DIV/0!	8362530	116146	72.00016	66	$454,735.00
	1864813	24248	76.90585			#DIV/0!			#DIV/0!	1864813	24248	76.90585	-34	$59,558.00
	8889453	125558	70.79957			#DIV/0!			#DIV/0!	8889453	125558	70.79957	11	$491,260.00
RESULT AFTER FINAL PICKS	5032165	215558	23.34483	9507030	89979	105.6583	-1.2E+07	-214591	57.5891	2181093	90946	23.98229		$10,367,815.25

Field Check - Engineers

Checked condition of road, etc. in the real world

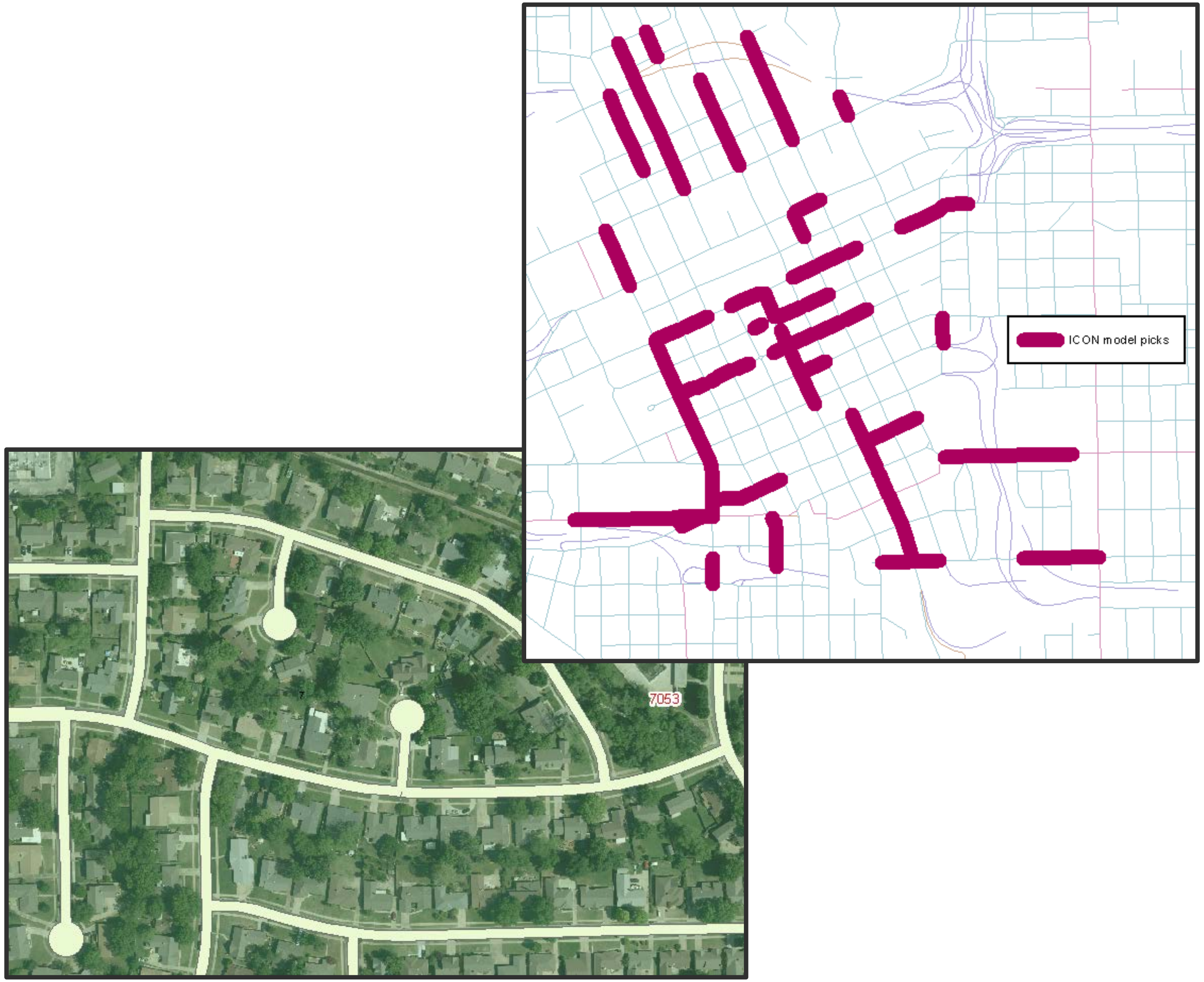
→ modifications to picks (more swapping)

Central Business District (CBD) and Non-Arterials

Downtown (CBD): Repeated process

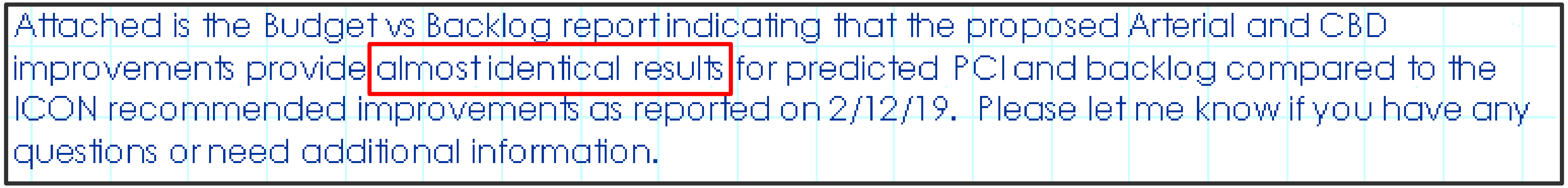
Non-Arterials

- Post-selection entirely up to engineers
- Similar process:
  - Cul-de-sacs!
  - Different grouping process

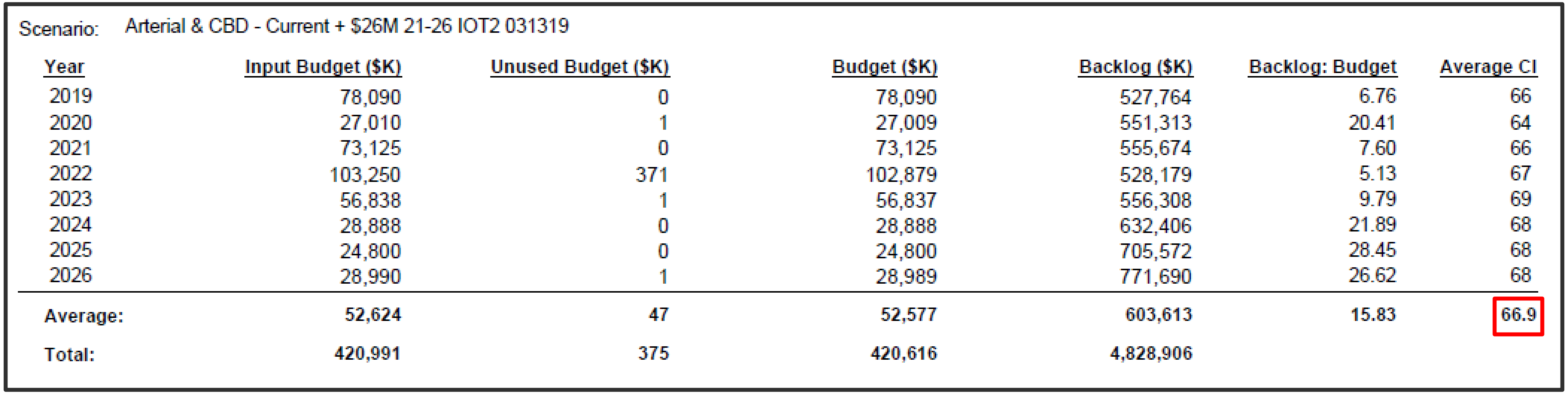


Consultant – Re-ran ICON model

Used model output with Engineer modifications:



Attached is the Budget vs Backlog report indicating that the proposed Arterial and CBD improvements provide almost identical results for predicted PCI and backlog compared to the ICON recommended improvements as reported on 2/12/19. Please let me know if you have any questions or need additional information.



Scenario: Arterial & CBD - Current + $26M 21-26 IOT2 031319

Year	Input Budget ($K)	Unused Budget ($K)	Budget ($K)	Backlog ($K)	Backlog: Budget	Average CI
2019	78,090	0	78,090	527,764	6.76	66
2020	27,010	1	27,009	551,313	20.41	64
2021	73,125	0	73,125	555,674	7.60	66
2022	103,250	371	102,879	528,179	5.13	67
2023	56,838	1	56,837	556,308	9.79	69
2024	28,888	0	28,888	632,406	21.89	68
2025	24,800	0	24,800	705,572	28.45	68
2026	28,990	1	28,989	771,690	26.62	68
Average:	**52,624**	**47**	**52,577**	**603,613**	**15.83**	**66.9**
Total:	**420,991**	**375**	**420,616**	**4,828,906**		

RESULTS ARE A STARTING POINT……

Results Presented to City Administration

Decisions about Civic Improvements are complex, involving input from:

- Model Output
- Engineers
- Citizens – Public meetings
- City Administration

Administration has to weigh all of those factors to come up with a final package to present to the community.

Citizens vote to approve the package.