Weight and Veterans' Environments Study

Street connectivity: Weight and Veterans' Environments Study GIS protocol

Xiang W, Jones KK, Matthews SA, Abu Zayd H, Slater S, Zenk SN.

UIC Neighborhoods + Health

Overview

This protocol describes the process by which street connectivity (i.e., intersection density, intersection type) measures were created.

Acknowledgements

This protocol was developed with grant support from the National Cancer Institute (R21CA195543) and the Department of Veterans Affairs (IIR 13-085), co-led by Shannon Zenk and Elizabeth Tarlov. Shubhayan Ukil and Abby Klemp helped to edit the protocol for dissemination.

Suggested Citation

Xiang W, Jones K, Matthews SA, Abu Zayd H, Zenk SN. (2018 Street connectivity: Weight and Veterans' Environments Study GIS protocol, Version 1. Retrieved from Weight and Veterans' Environments Study website: https://waves.uic.edu/.

Table of Contents

Background

This document describes the work process used by WAVES for constructing streets connectivity measures.

Data

Sources

The data source is 2010, 2011, 2013 Street file, Z-level file and land use file from ArcGIS StreetMap Premium.

Definitions

Z-level definition and years

Z-level represent the relative vertical position of the shape point. The Z-level is used to represent the crossing over or under of links with other links. This attribute is not to be used to indicate actual elevation gain or loss. It is used to prevent routing between links that do not connect in reality.

For example, if Z-level = 1, it represents an overpass, if Z-level=-1, it means an underpass. We are interested in points that have Z-level = 0, which means streets at grade.

Z-level points are available in year 2010, 2011 and 2013.

In the final Z-level grids, there are 3 types of Z-levels:

> 1 and 2-way intersection – where a single street segment ends, or 2 street segments meet. This intersection can have any angle greater than 0 up to 180°
> 3-way intersection – 3 street segments meet. Most often in the form of a T-intersection
> 4-way and more intersection – 4 or more street segments meet. May look like two streets crossing, or more complicated.

Cleaning

Decisions

Create eligible Z-level points

In order to identify Z-level points which could represent street connections associated with neighborhood walkability, we select the Z-level points that meet certain criteria. We do not select Z-level points that intersect highways, bridges, tunnels (all of which are identified using fields in the raw input data). We select Z-level points that are in particular land types, based on the "land use" data: neighborhood boundary, park (city/county), university/college. Selected Z-level points are exported as "eligible" Z-levels points.

This process is done in the following Python script. Selection criteria are documented within the scripts:

Page6: 2010&2011

Page 9: 2013

For reference, eligible Z-level points are stored in separate gdb by year, with naming convention as follows:

[year]DataWorking

with file naming conventions:

Zlevel_[Year]_FINAL

A variable/field (InterType) is included in the final eligible Z-level point files used to identify intersection types for further analysis.

Z-level grids generation process

After the eligible Z-level points have been generated, the next step is to create raster datasets for different intersection types.

1 and 2-way intersection
3-way intersection
4-way and more intersection

Rasters were constructed that included the number of each type of intersection within specified distances of each cell centroid. These distances are: 400m, 1600m, 4800m, 8000m.

The Python script is used to create grids is given in Appendix C (Page 12).

Appendix

Note: Script will need to be adjusted with project-specific file names and locations.

A: Selection Criteria - Zlevel 2010 and 2011

```
import arcpy, sys, os, time

#set up workspace
```

```
in_workspace = {insert file location}
out_workspace = {insert file location}
arcpy.env.extent = "-2493045.0 -1429501.25 2342655.0 1703218.75"
zlevel_Name = "zlevel"
streets_Name = "street"
landuse_Name = "MapLandArea"

arcpy.env.workspace = in_workspace
arcpy.env.overwriteOutput = True

#assign zlevel file to zlevel variable
#assign street file to street variable
features = arcpy.ListFeatureClasses()

for feature in features:
                if feature.find(zlevel_Name)>-1:
                                zlevel_point = feature
                if feature.find(streets_Name)>-1:
                                streets = feature
                if feature.find(landuse_Name)>-1:
                                landUse = feature

#define layer variables
lyr_zlevel = "lyr_zlevel"
lyr_streets = "lyr_streets"
lyr_landUse = "lyr_landUse"
lyr_streets_intrsctN =  "lyr_streets_intrsctN"

print "Inputs:\n"
print zlevel_point, streets, landUse
#make layer file for needed feature classes (SelectLayerByAttribute only works on lyr file)
arcpy.MakeFeatureLayer_management(zlevel_point, lyr_zlevel)
arcpy.MakeFeatureLayer_management(streets, lyr_streets)
arcpy.MakeFeatureLayer_management(landUse, lyr_landUse)

start = time.time()

#Step 1. Add field "IntrsctElg" to zlevel point and street
arcpy.AddField_management(lyr_zlevel, "IntrsctElg", "TEXT", "", "", "", "", "NULLABLE",
"NON_REQUIRED")
arcpy.AddField_management(lyr_streets, "IntrsctElg", "TEXT", "", "", "", "", "NULLABLE",
"NON_REQUIRED")
#Calculate unwanted streets as IntrsctElg = N
msg = "Step 1. Calculating unwanted streets as IntrsctElg = N..."
print msg
where = '"TUNNEL" = \'Y\' OR "BRIDGE" = \'Y\' OR "SPEED_CAT" = \'8\' OR "POIACCESS" = \'Y\' OR
"FERRY_TYPE" = \'R\' OR "FERRY_TYPE" = \'B\' OR "PLOT_ROAD" = \'Y\' OR "RAMP" = \'Y\' OR
"ROUNDABOUT" = \'Y\' OR "PAVED" = \'N\' OR "INTERINTER" = \'Y\' OR "CONTRACC" = \'Y\''
```

```
arcpy.SelectLayerByAttribute_management(lyr_streets, "NEW_SELECTION", where)
arcpy.CalculateField_management(lyr_streets, "IntrsctElg", '"N"',"PYTHON_9.3")
'''
#Step X. select unwanted streets
msg = "Selecting unwanted streets..."
print msg
where = '"IntrsctElg"=\'N\'' #STRING MUST BE ENCLOSED IN SINGLE QUOTES
arcpy.SelectLayerByAttribute_management(lyr_streets, "NEW_SELECTION", where)
msg = "Exporting unwanted streets..."
print msg
'''
#Step 2. export unwanted streets as another feature, then create layer from it. IF NOT, WILL PRODUCE
"OUT OF MEMEORY" ERROR.
msg = "Step 2. Export unwanted streets..."
print msg
arcpy.FeatureClassToFeatureClass_conversion(lyr_streets, out_workspace, streets+"_Intrsct_N")
arcpy.MakeFeatureLayer_management(out_workspace+streets+"_Intrsct_N", lyr_streets_intrsctN)

#Step 3. select zlevels that intersect unwanted streets
arcpy.SelectLayerByLocation_management(lyr_zlevel,"INTERSECT",lyr_streets_intrsctN)
#Calculate those zlevels as IntrsctElg = N
msg = "Step 3. Calculating zlevel intersecting unwanted streets as ineligible..."
print msg
arcpy.CalculateField_management(lyr_zlevel, "IntrsctElg", '"N"',"PYTHON_9.3")

#switch zlevel selection and calculate the inverse as IntrsctElg = Y
msg = "Switch selection, calculating zlevel as eligible..."
print msg
arcpy.SelectLayerByAttribute_management(lyr_zlevel, "SWITCH_SELECTION")
arcpy.CalculateField_management(lyr_zlevel, "IntrsctElg", '"Y"',"PYTHON_9.3")
#Step 4. select unwanted land use
msg = "Step 4. Selecting unwanted land use..."
print msg
where = '"FEAT_TYPE" <> \'MILITARY BASE\' AND "FEAT_TYPE" <> \'NEIGHBOURHOOD BOUNDARY\'
AND "FEAT_TYPE" <> \'PARK (CITY/COUNTY)\' AND "FEAT_TYPE" <> \'PARK (STATE)\' AND "FEAT_TYPE"
<> \'PARK/MONUMENT (NATIONAL)\' AND "FEAT_TYPE" <> \'UNIVERSITY/COLLEGE\' AND "FEAT_TYPE"
<> \'NATIVE AMERICAN RESERVATION\''
arcpy.SelectLayerByAttribute_management(lyr_landUse, "NEW_SELECTION", where)
#Step 5. select zlevel (those IntrsctElg = Y ) in unwanted land use
arcpy.SelectLayerByLocation_management(lyr_zlevel,"COMPLETELY_WITHIN",lyr_landUse,"","SUBSET_S
ELECTION")
#Calculate those zlevels as IntrsctElg = N
msg = "Step 5. Calculating zlevel in unwanted land as ineligible..."
print msg
arcpy.CalculateField_management(lyr_zlevel, "IntrsctElg", '"N"',"PYTHON_9.3")

'''
#Also export the full z-level dataset
```

```
arcpy.SelectLayerByAttribute_management(lyr_zlevel, "CLEAR_SELECTION")
arcpy.FeatureClassToFeatureClass_conversion(lyr_zlevel, out_workspace, zlevel_point)
'''

#Select intersection eligibility = Y, dissolve by NODE_ID field, to create x-way count
msg = "Export output..."
print msg
where = '"IntrsctElg"=\'Y\''
arcpy.SelectLayerByAttribute_management(lyr_zlevel, "NEW_SELECTION", where)
arcpy.Dissolve_management("lyr_zlevel", out_workspace+zlevel_point+"_FINAL",["NODE_ID"],
[["NODE_ID","COUNT"]], "SINGLE_PART", "DISSOLVE_LINES")
#rename output field # of x-way
arcpy.AlterField_management(out_workspace+zlevel_point+"_FINAL","COUNT_NODE_ID","InterType","
InterType")
end = time.time()
print str((end-start)/60)+" minutes"
```

B: Selection Criteria - Z Level 2013

```
#This is the same as the previous script, except optimized for input names in 2013
import arcpy, sys, os, time

#set up workspace
in_workspace = {insert file location}
out_workspace = {insert file location}
arcpy.env.extent = "-2493045.0 -1429501.25 2342655.0 1703218.75"
zlevel_Name = "zlevel"
streets_Name = "streets"
landuse_Name = "MapLandArea"

arcpy.env.workspace = in_workspace
arcpy.env.overwriteOutput = True

features = arcpy.ListFeatureClasses()

#assign zlevel file to zlevel variable
#assign street file to street variable
for feature in features:
        if feature.find(zlevel_Name)>-1:
                zlevel_point = feature
        if feature.find(streets_Name)>-1:
                streets = feature
        if feature.find(landuse_Name)>-1:
                landUse = feature

#define layer variables
lyr_zlevel = "lyr_zlevel"
```

```
lyr_streets = "lyr_streets"
lyr_landUse = "lyr_landUse"
lyr_streets_intrsctN =  "lyr_streets_intrsctN"

print "Inputs:\n"
print zlevel_point, streets, landUse
#make layer file for needed feature classes (SelectLayerByAttribute only works on lyr file)
arcpy.MakeFeatureLayer_management(zlevel_point, lyr_zlevel)
arcpy.MakeFeatureLayer_management(streets, lyr_streets)
arcpy.MakeFeatureLayer_management(landUse, lyr_landUse)

start = time.time()

#Step 0. Add field "PLOT_ROAD" to streets file
arcpy.AddField_management(lyr_streets, "PLOT_ROAD", "TEXT", "", "", "", "", "NULLABLE",
"NON_REQUIRED")
msg = "Step 0. Calculating streets PLOT_ROAD attribute..."
print msg
where = '"FuncClass" = 5 AND "SpeedCat" = 8 AND "LANE_CATEGORY" = 1 AND "RestrictCars" = \'N\'
AND "RestrictPedestrians" = \'N\' AND "RestrictThroughTraffic" = \'Y\''
arcpy.SelectLayerByAttribute_management(lyr_streets, "NEW_SELECTION", where)
arcpy.CalculateField_management(lyr_streets, "PLOT_ROAD", '"Y"',"PYTHON_9.3")
arcpy.SelectLayerByAttribute_management(lyr_streets, "CLEAR_SELECTION")
#Step 1. Add field "IntrsctElg" to zlevel point and street
arcpy.AddField_management(lyr_zlevel, "IntrsctElg", "TEXT", "", "", "", "", "NULLABLE",
"NON_REQUIRED")
arcpy.AddField_management(lyr_streets, "IntrsctElg", "TEXT", "", "", "", "", "NULLABLE",
"NON_REQUIRED")
#Calculate unwanted streets as IntrsctElg = N
msg = "Step 1. Calculating unwanted streets as IntrsctElg = N..."
print msg
where = '"TUNNEL" = \'Y\' OR "BRIDGE" = \'Y\' OR "SpeedCat" = 8 OR "POI_ACCESS" = \'Y\' OR
"FerryType" = \'R\' OR "FerryType" = \'B\' OR "PLOT_ROAD" = \'Y\' OR "Ramp" = \'Y\' OR "Roundabout"
= \'Y\' OR "Paved" = \'N\' OR "INTERSECTION_CATEGORY" = \'Y\' OR "ContrAcc" = \'Y\''
arcpy.SelectLayerByAttribute_management(lyr_streets, "NEW_SELECTION", where)
arcpy.CalculateField_management(lyr_streets, "IntrsctElg", '"N"',"PYTHON_9.3")
'''
#Step X. select unwanted streets
msg = "Selecting unwanted streets..."
print msg
where = '"IntrsctElg"=\'N\'' #STRING MUST BE ENCLOSED IN SINGLE QUOTES
arcpy.SelectLayerByAttribute_management(lyr_streets, "NEW_SELECTION", where)
msg = "Exporting unwanted streets..."
print msg
'''
#Step 2. export unwanted streets as another feature, then create layer from it. IF NOT, WILL PRODUCE
"OUT OF MEMEORY" ERROR.
msg = "Step 2. Export unwanted streets..."
```

```
print msg
arcpy.FeatureClassToFeatureClass_conversion(lyr_streets, out_workspace, streets+"_Intrsct_N")
arcpy.MakeFeatureLayer_management(out_workspace+streets+"_Intrsct_N", lyr_streets_intrsctN)

#Step 3. select zlevels that intersect unwanted streets
arcpy.SelectLayerByLocation_management(lyr_zlevel,"INTERSECT",lyr_streets_intrsctN)
#Calculate those zlevels as IntrsctElg = N
msg = "Step 3. Calculating zlevel intersecting unwanted streets as ineligible..."
print msg
arcpy.CalculateField_management(lyr_zlevel, "IntrsctElg", '"N"',"PYTHON_9.3")

#switch zlevel selection and calculate the inverse as IntrsctElg = Y
msg = "Switch selection, calculating zlevel as eligible..."
print msg
arcpy.SelectLayerByAttribute_management(lyr_zlevel, "SWITCH_SELECTION")
arcpy.CalculateField_management(lyr_zlevel, "IntrsctElg", '"Y"',"PYTHON_9.3")
#Step 4. select unwanted land use
msg = "Step 4. Selecting unwanted land use..."
print msg
#land feature in this list is the land types to keep
where = '"FEATURE_TYPE" <> 900108 AND "FEATURE_TYPE" <> 908002 AND "FEATURE_TYPE" <>
900150 AND "FEATURE_TYPE" <> 900130 AND "FEATURE_TYPE" <> 900103 AND "FEATURE_TYPE" <>
2000403 AND "FEATURE_TYPE" <> 900107'
arcpy.SelectLayerByAttribute_management(lyr_landUse, "NEW_SELECTION", where)
#Step 5. select zlevel (those IntrsctElg = Y ) in unwanted land use
arcpy.SelectLayerByLocation_management(lyr_zlevel,"COMPLETELY_WITHIN",lyr_landUse,"","SUBSET_S
ELECTION")
#Calculate those zlevels as IntrsctElg = N
msg = "Step 5. Calculating zlevel in unwanted land as ineligible..."
print msg
arcpy.CalculateField_management(lyr_zlevel, "IntrsctElg", '"N"',"PYTHON_9.3")

'''
#Also export the full z-level dataset
arcpy.SelectLayerByAttribute_management(lyr_zlevel, "CLEAR_SELECTION")
arcpy.FeatureClassToFeatureClass_conversion(lyr_zlevel, out_workspace, zlevel_point)
'''

#Select intersection eligibility = Y, dissolve by NODE_ID field, to create x-way count
msg = "Export output..."
print msg
where = '"IntrsctElg"=\'Y\''
arcpy.SelectLayerByAttribute_management(lyr_zlevel, "NEW_SELECTION", where)
arcpy.Dissolve_management("lyr_zlevel", out_workspace+zlevel_point+"_FINAL",["NODE_ID"],
[["NODE_ID","COUNT"]], "SINGLE_PART", "DISSOLVE_LINES")
#rename output field # of x-way
arcpy.AlterField_management(out_workspace+zlevel_point+"_FINAL","COUNT_NODE_ID","InterType","
InterType")
```

```
arcpy.AddField_management(out_workspace+zlevel_point+"_FINAL", "count", "SHORT", "", "", "", "",
"NULLABLE", "NON_REQUIRED")
arcpy.CalculateField_management(out_workspace+zlevel_point+"_FINAL", "count", 1,"PYTHON_9.3")

end = time.time()
print str((end-start)/60)+" minutes"
```

C: Create Grids

```
#This script generates zlevel grids from zlevel points

import arcpy, os, time
from arcpy.sa import *

in_workspace = {insert file location}
out_workspace = {insert file location}
arcpy.env.workspace = in_workspace
arcpy.env.extent = "-2493045.0 -1429501.25 2342655.0 1703218.75"
arcpy.env.overwriteOutput = True
arcpy.CheckOutExtension("Spatial")

start = time.time()
#2013
lyr = "zlevel_2013"
arcpy.MakeFeatureLayer_management("zlevel2013_FINAL", lyr)
#do 1 or 2 ways intersection
where = '"InterType"=1 OR "InterType"=2'
arcpy.SelectLayerByAttribute_management(lyr, "NEW_SELECTION", where)
# Set point statistics variables
inPointFeatures = lyr
field = "count"
cellSize = 30
neighborhood_400 = NbrCircle(400, "MAP")
neighborhood_1600 = NbrCircle(1600, "MAP")
neighborhood_4800 = NbrCircle(4800, "MAP")
neighborhood_8000 = NbrCircle(8000, "MAP")

# Execute point statistics
outPointStatistics = PointStatistics(inPointFeatures, field, cellSize,neighborhood_400, "SUM")
outPointStatistics.save(out_workspace+"IntersctDen_2013_1n2_400m")
print lyr
outPointStatistics = PointStatistics(inPointFeatures, field, cellSize,neighborhood_1600, "SUM")
outPointStatistics.save(out_workspace+"IntersctDen_2013_1n2_1600m")
print lyr
outPointStatistics = PointStatistics(inPointFeatures, field, cellSize,neighborhood_4800, "SUM")
outPointStatistics.save(out_workspace+"IntersctDen_2013_1n2_4800m")
```

```
print lyr
outPointStatistics = PointStatistics(inPointFeatures, field, cellSize,neighborhood_8000, "SUM")
outPointStatistics.save(out_workspace+"IntersctDen_2013_1n2_8000m")
print lyr
end = time.time()
print str((end-start)/60)+" minutes"


#do 3 ways intersection
where = '"InterType"=3'
arcpy.SelectLayerByAttribute_management(lyr, "NEW_SELECTION", where)
# Set point statistics variables
inPointFeatures = lyr
field = "count"
cellSize = 30
neighborhood_400 = NbrCircle(400, "MAP")
neighborhood_1600 = NbrCircle(1600, "MAP")
neighborhood_4800 = NbrCircle(4800, "MAP")
neighborhood_8000 = NbrCircle(8000, "MAP")

# Execute point statistics
outPointStatistics = PointStatistics(inPointFeatures, field, cellSize,neighborhood_400, "SUM")
outPointStatistics.save(out_workspace+"IntersctDen_2013_3_400m")
print lyr
outPointStatistics = PointStatistics(inPointFeatures, field, cellSize,neighborhood_1600, "SUM")
outPointStatistics.save(out_workspace+"IntersctDen_2013_3_1600m")
print lyr
outPointStatistics = PointStatistics(inPointFeatures, field, cellSize,neighborhood_4800, "SUM")
outPointStatistics.save(out_workspace+"IntersctDen_2013_3_4800m")
print lyr
outPointStatistics = PointStatistics(inPointFeatures, field, cellSize,neighborhood_8000, "SUM")
outPointStatistics.save(out_workspace+"IntersctDen_2013_3_8000m")
print lyr
end = time.time()
print str((end-start)/60)+" minutes"

#do 4 way or more intersection
where = '"InterType">=4'
arcpy.SelectLayerByAttribute_management(lyr, "NEW_SELECTION", where)
# Set point statistics variables
inPointFeatures = lyr
field = "count"
cellSize = 30
neighborhood_400 = NbrCircle(400, "MAP")
neighborhood_1600 = NbrCircle(1600, "MAP")
neighborhood_4800 = NbrCircle(4800, "MAP")
neighborhood_8000 = NbrCircle(8000, "MAP")
```

```
# Execute point statistics
outPointStatistics = PointStatistics(inPointFeatures, field, cellSize,neighborhood_400, "SUM")
outPointStatistics.save(out_workspace+"IntersctDen_2013_4nMore_400m")
print lyr
outPointStatistics = PointStatistics(inPointFeatures, field, cellSize,neighborhood_1600, "SUM")
outPointStatistics.save(out_workspace+"IntersctDen_2013_4nMore_1600m")
print lyr
outPointStatistics = PointStatistics(inPointFeatures, field, cellSize,neighborhood_4800, "SUM")
outPointStatistics.save(out_workspace+"IntersctDen_2013_4nMore_4800m")
print lyr
outPointStatistics = PointStatistics(inPointFeatures, field, cellSize,neighborhood_8000, "SUM")
outPointStatistics.save(out_workspace+"IntersctDen_2013_4nMore_8000m")
print lyr
end = time.time()
print str((end-start)/60)+" minutes"
```