

An Assessment of a Stacked Shale Gas Play and the Effect on Forest Fragmentation in Pennsylvania

Author: Benjamin Ogle, GISP, MGIS Candidate

Advisor: Patrick Drohan, Ph.D.

GEOG 596B: Individual Studies – Capstone Project

Masters in Geographic Information Systems

The Pennsylvania State University

Fall 2017

12/08/2017

Table of Contents

I.	Abstract.....	4
II.	Project Background.....	5
2.1	History of Oil & Gas in Pennsylvania	5
2.2	Shale Gas Exploration in the State	6
2.3	Directional Drilling & Hydraulic Fracturing.....	7
2.4	Facilities and Structures Involved in Extraction of Shale Gas	7
2.5	Shale Formations in Pennsylvania	8
2.5.1	Marcellus Shale	9
2.5.2	Utica Shale	10
2.5.3	Burket/Geneseo Shale.....	10
2.6	Oil & Gas Documents	11
2.6.1	Oil & Gas Leases	11
2.6.2	Surface Agreements	12
2.6.3	Declaration of Unitization	13
2.7	What is a Stacked Shale Play.....	14
2.8	Forest Fragmentation Literature Review	14
III.	Project Framework.....	16
3.1	Objectives & Key Research Questions.....	16
3.2	Study Area.....	16
3.3	Methodology.....	17
3.4	Data Management	17
3.4.1	Well Production Dataset	17
3.4.2	Digitizing Drilling Units & Generating the Study Area for Process 1 (check).....	19
3.5	Process 1: Forest Fragmentation Analysis.....	21
3.5.1	Landscape Fragmentation Tool (LFT) v 2.0	22
3.5.2	Susquehanna County Fragmentation Webmap.....	23
3.5.3	Susquehanna County Fragmentation Observations	24
3.5.4	Local Indicators of Spatial Association (LISA)	25
3.6	Process 2: Well Production Data Analysis	31
3.6.1	Point Pattern Analysis using Monte Carlo Simulations.....	32

3.6.2	Local indicators of spatial association (LISA) analysis for horizontal wells by highest yearly gas production	34
3.6.3	Well Production by Marcellus Formation Thickness and Depth	36
3.6.4	Anisotropic Semivariogram Methods	36
3.7	Process 3: Develop Tool based on Findings.....	39
3.7.1	Tool Mock-up	39
3.7.2	Slope Analysis at Wellpad	39
3.7.3	Tool User Interface	40
3.7.4	Tool Workflow.....	41
3.7.5	CSV Module	42
3.7.6	Tool Example.....	43
3.8	Sharing Developed Tools and Datasets	45
IV.	Results & Conclusions	45
4.1	Project Timeline	45
4.2	Challenges	46
	References	46
	Appendix A: Source Code.....	48
	Appendix A, Script 1: Source code for LISA Analysis Areas in Python	48
	Appendix A, Script 2: Source Code for density analysis and Monte Carlo simulation in R	53
	Appendix A, Script 3: Source code for Process 3 in Python	53
	Appendix A, Script 4: Source code for Process 3 validation in Python	64
	Appendix B: Data Sources	66

I. Abstract

In a period of sustained lower gas prices, oil and gas exploration companies are required to take steps to produce in a more efficient manner. Drilling techniques, which were profitable when crude oil prices were \$110 per barrel, may no longer be viable at \$50 per barrel seen today. In the Marcellus and Utica shale plays, gas exploration companies have become more efficient in their drilling techniques. Nevertheless, it has continued to be difficult for these companies to remain profitable. One process that has been implemented recently is stacking shale plays from the same well pad. In this project, research was conducted to understand the potential for stacked shale plays in the northeastern United States. By drilling wells into two, or even three, shale formations from a single well pad location, costs could be dramatically reduced with increased shale gas production. Additionally, by reducing the number of pad locations, gathering lines, and access roads, there should be a lower impact of habitat fragmentation resulting from "industrial linear corridors" in forested landscapes of the northeast. This project will consist of a forest fragmentation and well production analysis. Python, Feature Manipulation Engine (FME) Desktop, and Esri ModelBuilder will be used to process the datasets. R and GeoDa will be used to analyze well production the results. Based on these findings, a tool will be developed using Python that will allow a gas exploration company to locate areas where a stacked shale play is economically viable. Results will be shared with the organization by utilizing ArcGIS Server and ArcGIS Online.

Keywords: *Shale, Marcellus, Utica, Burket, Forest, Fragmentation, ArcGIS, R, FME, Monte Carlo, GeoDa, Oil & Gas, Pennsylvania, Washington, Susquehanna, Well, Formation, Autocorrelation*

II. Project Background

2.1 History of Oil & Gas in Pennsylvania

Pennsylvania has a long and significant history relating to oil and gas that dates to the early inhabitations. Seneca Indians told early explorers about the oil seeps they found along the banks of Oil Creek in Venango County, Pennsylvania. They skimmed oil off the surface of the oily water with blankets. The collected oil would eventually be used for trading, ceremonial acts, and medicinal purposes, including treatment of stomach ailments, aching muscles, and dry skin.

In the early 1800's, Samuel M. Kier operated a salt well near Tarentum, Pa. Salty water was pumped out of the well and distilled to create rock salt. At the time, salt drillers often became discouraged when their wells produced greasy crude oil with their desired salt water. Kier experimented with refining crude oil he produced into kerosene and is credited as the founder of the American oil refining industry. The kerosene burned brightly in lamps, provided good heat for warmth or cooking, and was considerably cheaper than whale oil (PA DNR, n.d.).

Because of Kier's discovery, there was an increased demand for crude oil, which caught the attention of east coast investors. "Colonel" Edwin Drake, funded by such investors, drilled the first Pennsylvania oil well in 1859 in Venango County, near Titusville. Oil was found at 69.5 feet ushering in the modern oil industry (Figure 1). More than 350,000 oil and gas wells have been drilled in Pennsylvania since that time.



Figure 1: "Colonel" Edwin Drake oil well in Titusville, Venango County

The frenzy around oil in Northwest Pennsylvania eventually slowed. By 1907, the decline of the Pennsylvania fields along with the great oil discoveries made in Texas, California, and Oklahoma, left Pennsylvania with less than 10% of the nation's oil production. From 1930 to 1980, deep vertical gas drilling in Pennsylvania continued, and one of the main targets was the Lower Devonian Oriskany

Sandstone. This sandstone was below the Marcellus Shale formation. Oftentimes, while drilling, numerous shows of gas occurred while penetrating the Marcellus Shale. In the 1970's, the U.S. Department of Energy initiated the Eastern Gas Shale Project (EGSP) to study the geology and production potential of organic-rich shales in the northeastern United States. A total of 595 separate reports, articles, and reviews were generated by researchers, leading to an increased knowledge of the Marcellus Shale (Zagorski, Wrightstone, & Bowman, 2012, p. 174-177).

2.2 Shale Gas Exploration in the State

The Marcellus shale play began in earnest in 2003, when Range Resources drilled through the Marcellus to the Lower Silurian in Washington County, PA. The targeted layer was not productive, but the Marcellus showed promise. The company drilled additional wells in Washington County and experimented with drilling and hydraulic fracturing techniques first used in the Burkett Shale located in Texas. They began to successfully produce Marcellus gas in 2005. Many other gas exploration companies followed suit and began leasing acreage in the region. In late 2007, Range Resources announced initial test rates between 1.4 and 4.7 mcf/d for five horizontal wells drilled in the Marcellus. That announcement coincided with a press release from Penn State University. Terry Engelder, professor of geosciences, working in conjunction with Gary Lash, a geoscience professor at SUNY Fredonia, had estimated the recoverable gas from the Marcellus Shale to be 50 Tcf, more than 25 times the current U.S. Geological Survey estimate (Penn State, 2008).

From 2008 to 2015, gas exploration companies leased properties and drilled wells in the Marcellus Shale basin at a hurried pace. Much capital was spent during this time frame. By 2015 the U.S. Energy Information Administration (EIA) reported that hydraulic fracturing accounted for more than one-half of U.S. oil production and two-thirds of U.S. gas production. The price of oil and natural gas fell dramatically in mid-2014 (Figure 2); subsequently, the pace of newly permitted wells slowed. Drilling techniques, which were profitable when crude oil prices were \$110 per barrel, would no longer be viable at \$50 per barrel. Gas exploration companies have become more efficient in their drilling techniques. Nevertheless, it has remained difficult for these companies to stay profitable. Companies need to be focused on returns on investment, rather than production growth, as the most significant metric for success in the exploration-and-production industry.

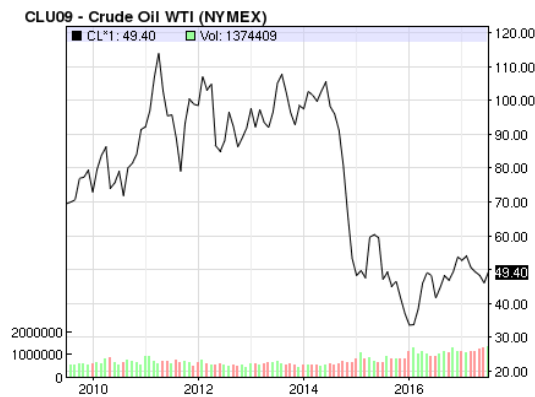


Figure 2: Price of crude oil (WTI)

2.3 Directional Drilling & Hydraulic Fracturing

These techniques were pioneered in the Barnett Shale in Texas by George Mitchell in the early 1980's. Organic matter deposited with the shale formation was compressed and heated deep within the Earth over geologic time, forming hydrocarbons, including natural gas. The gas occurs in fractures, in the pore spaces between individual mineral grains, and is chemically adsorbed onto organic matter within the shale (Soeder, 1988, p. 116-117). To produce commercial amounts of natural gas from such fine-grained rock, higher permeability flowpaths must be created in the formation. This is generally done using a technique called hydraulic fracturing, where water under high pressure forms fractures in the rock, which are propped open by sand or other materials to provide pathways for gas to move to the wellhead. Petroleum engineers refer to this fracturing process as "stimulation".

Step 1 – Directional Drill

- First the well is drilled vertically, once it reaches the "kickoff point" where the bit begins curving to become horizontal.
- A steel casing is cemented along the vertical length to prevent water contamination.
- The horizontal section of the well is drilled. An additional steel casing with cement is inserted into the horizontal length or lateral.

Step 2 - Hydraulic Fracturing

- A Perforating gun is fired along a section of the horizontal length of the well lateral, creating holes in the casing, cement, and into the target formation.
- A mixture of water, sand, and chemicals that are injected into the well and through the perforations at high pressures (5–10,000 psi) creating fractures into the formation.
- This section is isolated with a plug, and these steps are repeated along the horizontal length of the well lateral
- Once stimulation is complete, the plugs are drilled out and production begins.

During the initial production, 15% to 50% of the fracturing fluid is recovered. These fluids are recycled or safely disposed of per government regulations.

2.4 Facilities and Structures Involved in Extraction of Shale Gas

These can be seen in Figure 3.

- A. Well pad with horizontal drilling rig
- B. Water storage tanks at a water withdrawal station
- C. Water impoundment
- D. Well pad with horizontal drilling rig
- E. Completed well with "Christmas Tree"
- F. Condensate tanks to store produced water
- G. Hazard placards on the condensate tanks
- H. Pipeline construction in Washington County
- I. Liquids processing ("cryo") plant.



Figure 3: Facilities involved in the extraction of shale gas (Lampe & Stolz, 2015, p. 438)

2.5 Shale Formations in Pennsylvania

There are three shale formations that will be the focus of this project: Marcellus, Utica, and Burket/Geneseo (Figure 4). These could all be potentially incorporated into a stacked shale play.

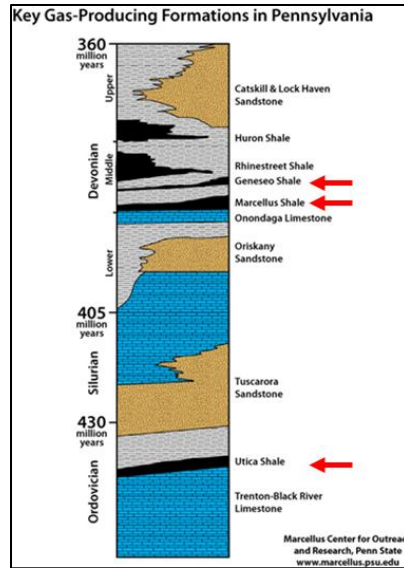


Figure 4: The Marcellus, Utica, and Burket/Genesee Shale formations will be the focus of this project

2.5.1 Marcellus Shale

The Marcellus Shale is a sedimentary rock formation deposited over 350 million years ago, in a shallow inland sea located in the eastern United States where the present-day Appalachian Mountains now stand (de Witt et al., 1993). The Marcellus Shale forms the bottom part of a thick sequence of Devonian age, sedimentary rocks in the Appalachian Basin. This shale formation contains significant quantities of natural gas. The basin subsided under the weight of the sediment, resulting in a wedge-shaped deposit that is thicker in the east and thins to the west. The eastern, thicker part of the sediment wedge is composed of sandstone, siltstone, and shale, whereas the thinner sediments to west consist of fine-grained, organic, rich black shale, interbedded with organic-lean gray shale (Figure 5).

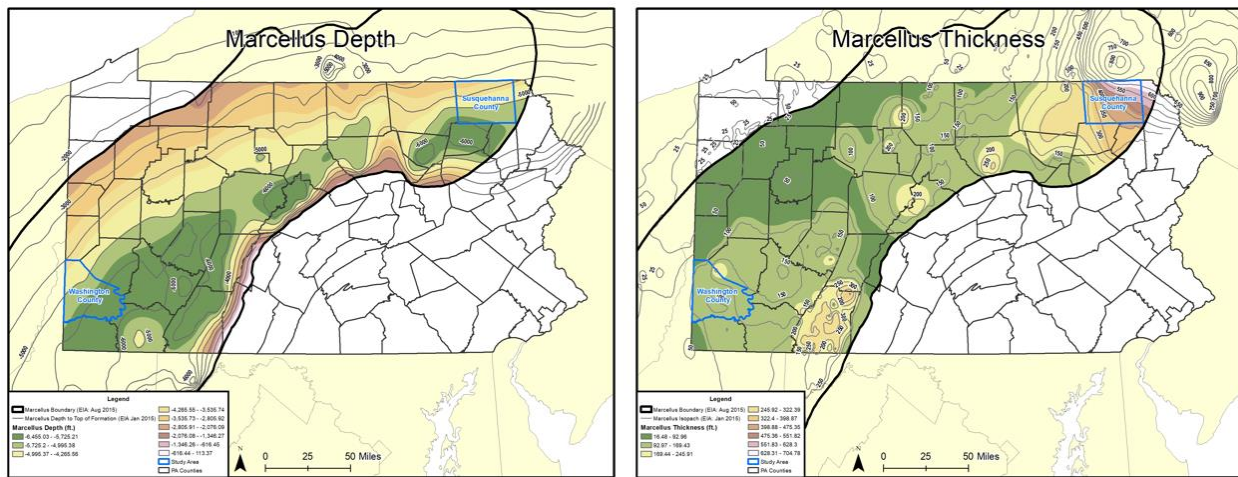


Figure 5: Marcellus Shale depth and thickness in Pennsylvania

The U.S. Energy Information Administration (EIA) recently estimated proven reserves in the Marcellus Shale play to be 77.2 trillion cubic feet (Tcf) in 2015, which makes it one of the largest natural gas plays in the U.S. For reference, the United States consumed an estimated 27.49 Tcf of natural gas in 2016. This estimate of proven reserves has fluctuated during this time of active exploration. As discussed, Terry Engelder, Penn State professor of geosciences and Gary Lash, a geoscience professor at SUNY Fredonia, had estimated the recoverable gas from the Marcellus Shale to be 50 Tcf in 2007; there have been estimates as high as 500 Tcf (Penn State, 2008);. Key geologic and technical criteria that control play boundaries include thermal maturity, total organic carbon (TOC), formation thickness, porosity, depth, pressure, and the ability to be fractured. Total Organic Carbon (TOC) content in the Marcellus formation ranges from less than 1% to 20% (Nyahay et al., 2007; Reed & Dunbar, 2008). Any TOC values less than 1% is considered poor while values greater than 12% is excellent conditions for gas extraction.

2.5.2 Utica Shale

The Utica Shale is a black, calcareous, organic-rich shale of Middle Ordovician age. The Utica Shale is located a few thousand feet below the Marcellus Shale (Figure 6). Because of its increased depth, there is an added cost to exploration companies to drill Utica wells. West Virginia University’s Appalachian Oil and Natural Gas Research Consortium said in 2015 the Utica contains technically recoverable resources of an astounding 782 Tcf of natural gas (Hohn, Pool, & Moore, 2015, p. 168). A large percentage of the technically recoverable resources in the Utica Shale falls outside of the Pennsylvania boundary; most well currently drilled into the Utica Shale are in eastern Ohio. The TOC for the Utica shale is estimated from 1% to 3% (U.S. Energy Information Administration, 2017).

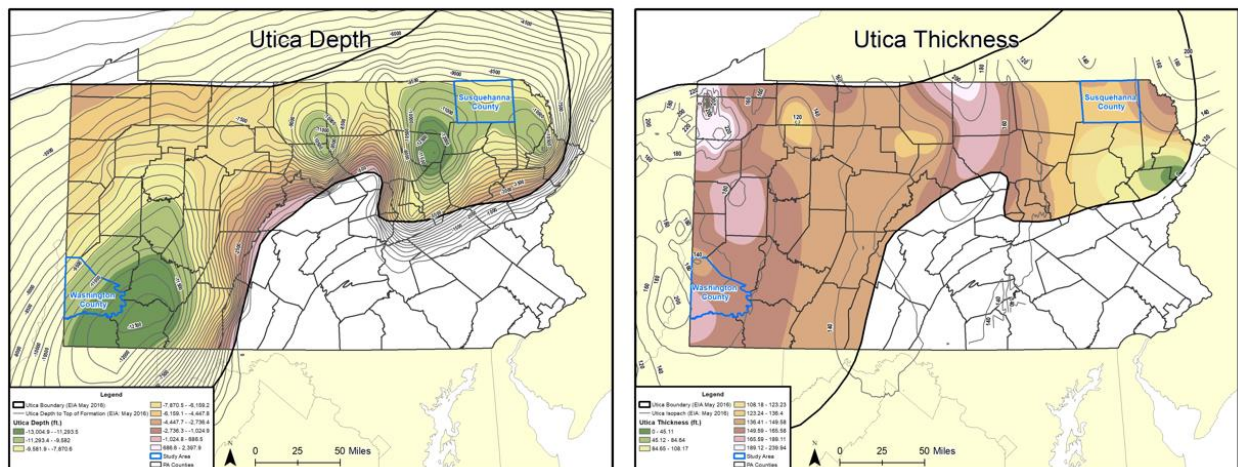


Figure 6: Utica Shale depth and thickness in Pennsylvania

2.5.3 Burket/Geneseo Shale

The organic-rich mudstone immediately above Tully Limestone is called Burket across most of Pennsylvania and West Virginia and Geneseo in northwest PA and New York (Figure 7). The distance from the Burket down to the Marcellus Shale ranges from 20 ft. in southwestern Pennsylvania and West

Virginia to more than 800 ft. in northeastern Pennsylvania. Gregory Wrightstone (2015) considered this formation to be Appalachia’s little brother to the Marcellus & Utica Shales. This formation is estimated that 33 TCF of recoverable gas. Wrightstone (2015) states that high volume production appears closely related to thicker, high-TOC quality areas, such as Washington County. Because the Burket is not as deep in Susquehanna County, it may not be economically viable in that location. The max TOC is estimated to be 3.8% (Arnold, 2015).

Drilling and completion costs could likely be reduced by utilizing existing drilling pads and infrastructure, especially completed Marcellus pads. There is little production data available as there are presently relatively few wells extracting gas from the formation. This formation could be part of the Appalachian Basin's stacked pay potential that producers could potentially develop (Wrightstone, 2015).

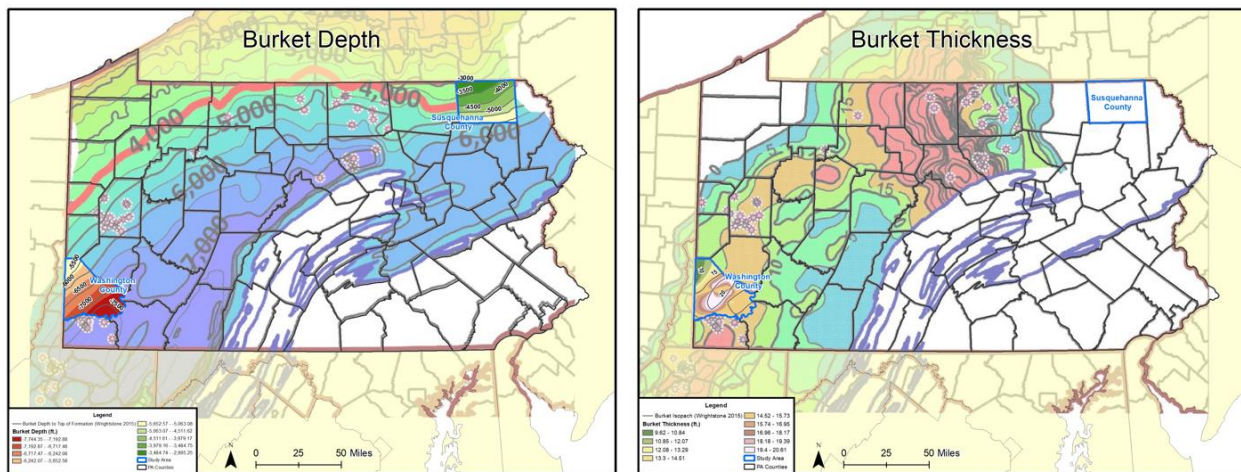


Figure 7: Burket/Geneseo Shale depth and thickness in Pennsylvania

2.6 Oil & Gas Documents

It is essential to develop an understanding of the legal documents that are made a public record by oil & gas companies. There is much that can be learned about the activities of these companies from recorded documents. This project will address three key documents: oil & gas leases, surface agreements, and declaration of unitization. These documents will be critical in developing the study area to realize forest fragmentation related to oil & gas activity.

2.6.1 Oil & Gas Leases

A landman with the gas exploration company contacts a mineral owner; if no prior lease is signed, the owner can sign with the company. There is typically a monetary per acre bonus when a lease is signed. Leases often last 5 years and can be renewed with an additional bonus, and have a gas royalty ranging from 12.5% to 22%. A mineral tract could have an active well (horizontal or vertical); coal mining activities, or underground storage facility from a previously signed lease. In this case, the property is “Held by Production” (HBP) by the gas company. After one year of production inactivity, the property is no longer held by the lease and can subsequently be leased again. Frequently, the owner of the

minerals differs from the owner of the surface. There may also be multiple owners of the minerals (example: a mineral owner leaves the rights to his mineral tract to his 20 grandchildren). Some Leases will only include mineral rights at certain depths or formations. Additionally, mineral owners may only own rights at certain depths or formations (Figure 8). These are central factors that gas companies need to consider when preparing for gas development. If a company does not have a large leasehold in a region, it may not be economically viable to pursue development.

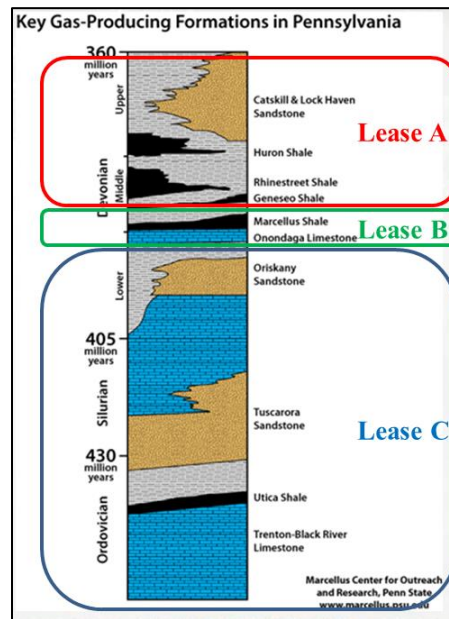


Figure 8: An illustration of how oil and gas lease ownership can vary by formation or depth - separate gas companies own minerals at different depths

2.6.2 Surface Agreements

There are two surface agreements that are important to develop an understanding, as they directly relate to forest fragmentation.

Surface Use Agreements (SUA): An agreement that is signed between the drilling company and the surface owner where oil and gas development, such as a well pad, is proposed to take place. A Surface Use Agreements (SUA) typically involves a monetary payment upon signing and an additional payment for damages to the surface owner’s property. Oftentimes, the surface owner is not the same as the mineral owner; in this case, the minerals have been severed from the surface. The SUA is usually filed in the county courthouse and is public record.

Right-of-Way Agreements: In most cases, a natural gas pipeline right-of-way agreement (or “Easement Agreement”) is used to construct, maintain, operate, protect, inspect, and/or replace one or more pipelines. The surface owner, who signs the agreement with the Pipeline Company, is typically compensated for the easement by payment per linear foot. Pipeline companies typically seek a 50 ft. or wider easement; the payment is based on the length of the easement. The right-of-way agreement is typically filed in the county courthouse and is public record.

2.6.3 Declaration of Unitization

The terms “pooling” and “unitization” are often used interchangeably in the oil & gas industry. A pooled unit is the joining together or a combination of small tracts or portions of tracts for the purpose of having sufficient acreage to receive a horizontal well drilling permit, and for the purpose of sharing production by interest owners in such a pooled unit (Kramer & Martin, 2006, p. 1-3). In most cases, mineral ownership for a horizontal well or wells is not held by one individual a Declaration of Unitization (or Pooling) is required, signed, and recorded in the county courthouse (Figure 9). An exhibit included in this document typically includes a map denoting the drilling unit outline by metes and bounds.

Declaration of Unitization allows mineral owners to understand their percentage of a drilling unit an expected monthly royalty. For example, if a unit produced \$100,000 in a month. An owner has mineral rights to 100 acres in a 1000-acre unit. The lease pays 20% royalties. The monthly royalty check would be \$2000.

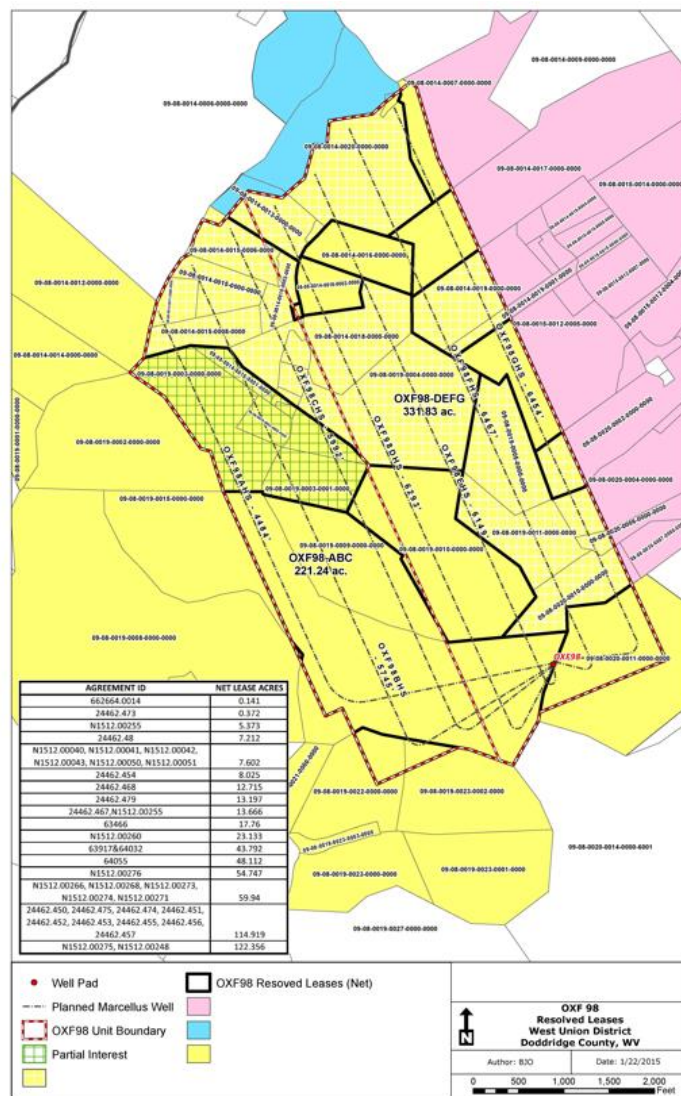


Figure 9: An example of a Declaration of Unitization (or Pooling) exhibit

2.7 What is a Stacked Shale Play

A stacked shale play is the technique of producing from multiple shale formations from the same well pad (Figure 10).

Hypothesis: By producing from multiple shale formations, gas exploration companies can increase overall well pad productivity and reduce costs, while reducing surface disruptions and forest fragmentation.

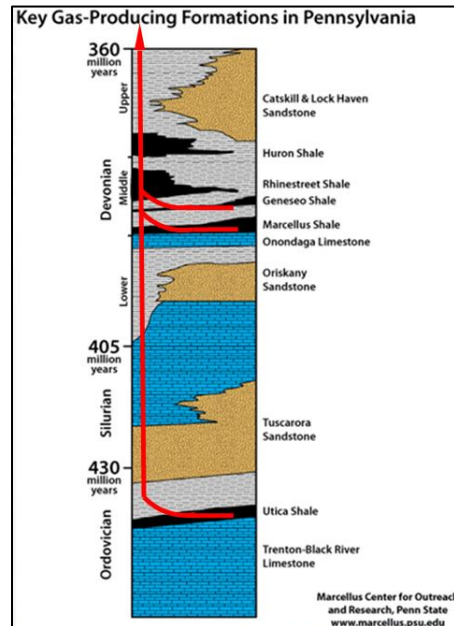


Figure 10: An illustration of a stacked shale play producing from multiple shale formations from the same well pad

2.8 Forest Fragmentation Literature Review

A literature review was conducted on studies of forest fragmentation attributed to Marcellus Shale exploration and resulting environmental or biological effects. Abrahams, Griffin, and Matthews (2012) explored policies aimed at reducing core forest fragmentation from Marcellus shale development in Pennsylvania. The case study focused on Bradford County, PA, which is a highly active region in the Marcellus shale play. The analysis of land use change was accomplished using a spatially explicit model in ArcGIS. This study considered two regulatory measures that could potentially reduce forest fragmentation: reducing well pad density by increasing the number of wells per pad and horizontal lateral length and requiring gathering lines to follow the path of pre-existing roadways in forested regions. The study determined that if laterals were drilled to approximately 3000 meters in horizontal length, an increase in forest core patches would increase by 25% above the 2012 levels. But by reducing well pad density by increasing lateral length, there would be a small, but positive impact on ecological conservation. These regulations would be cost-effective for the developers while providing a positive ecological impact (Abrahams et al., 2014, p. 157-59). Abrahams, Griffin, and Matthews concluded that gathering lines to be the largest infrastructure contributor to forest fragmentation.

In Drohan, Brittingham, Bishop, and Yoder's (2011) paper researched shale-gas development in Pennsylvania and the potential to cause substantial landscape disturbances. This paper used the [ArcGIS Landscape Fragmentation Tool ver. 2.0](#), which classifies forested areas into four main categories (patch, edge, perforated, and core) using a 100-m edge effect. A nonparametric two-sample Mood Median test was used to determine significant differences between well pads to a road or stream on private and public land by forest fragmentation class, physiographic section, and major hydrologic basin. An alpha of 0.05 indicated statistically significant differences, while an alpha of 0.10 identified marginally significant relationships. Statistics were given regarding gas exploration on public versus private land, the number of wells drilled per pad, the number of wells drilled across physiographic sections, and how gas exploration as changed land cover (Drohan et al., 2011, p. 1064). Results indicated that shale-gas development in Pennsylvania is increasing rapidly with time; is largely concentrated in the northcentral, northeast, and southwest parts of the state; and is mostly on private land. The paper concluded that a regional strategy should be developed to better manage habitat loss, farmland conservation, and risk to waterways.

In Kiviat's (2013) paper titled "Risks to Biodiversity from Hydraulic Fracturing for Natural Gas in the Marcellus and Utica Shales", research focused on how forest fragmentation, along with other potential risks, could affect biodiversity in the Marcellus and Utica Shales. Kiviat noted that 20% of forest cover may be removed and 80% of land may be affected if a 100-m edge effect is considered. Organisms sensitive to forest fragmentation include lichens, bryophytes, orchids, other herbs, the West Virginia white butterfly (*Pieris virginiensis*), amphibians, and birds. Furthermore, it was discussed how access roads act as corridors for the spread of non-native weeds that could spread into forested habitats. Vegetation along a pipeline's right-of-way are maintained by mowing or spraying herbicide, which the runoff could affect neighboring habitats. It was concluded that regulations generally occur at the level of the well pad, but little has been done to protect biodiversity and ecosystem services.

Lampe and Stolz's (2015) paper titled "Current perspectives on unconventional shale gas extraction in the Appalachian Basin", is a brief, but detailed assessment given on the shale gas process where environmental issues were addressed. This paper discussed the whole lifecycle of a project, from leasing to well pad reclamation. Additionally, potential environmental impacts of shale drilling were discussed. Lampe concluded that given the likelihood that upwards of 100,000 horizontal wells could be constructed over the next 20 years, it is critical that this activity is closely regulated so to avoid repeating past failures (Lampe & Stolz, 2015, p.443).

Manda, Heath, Klein, Griffin, and Montz (2014) paper "Evolution of multi-well pad development and influence of well pads on environmental violations and wastewater volumes in the Marcellus Shale" explored the development and influence of multi-well pads. The number of wells per pad has been steadily increasing as multi-well pads tend to be developed more because of economic rather than environmental considerations. This paper hypothesized that while multi-well pads will create less surface disturbance, they will produce more volumes of wastewater, and therefore potentially generate more environmental violations. Statistical analysis (regression and Mann-Whitney tests) were utilized to generate results (Manda et al., 2014, p. 38-39). The paper concluded that two to four times as much land surface disturbance would occur if there was only one well per pad. Additionally, it was concluded that more environmental violations were observed on multi-well pads (but when considering the overall

number of wells there are fewer violations). Finally, it was determined that there was a larger volume of wastewater and a greater proportion that was recycled at multi-well pads (Manda et al., 2014, p. 44-45).

III. Project Framework

This project will consist of a forest fragmentation analysis and a well production analysis. Python, Feature Manipulation Engine (FME) Desktop, and Esri ModelBuilder will be used to process the datasets. R and GeoDa will be used to analyze well production results. Based on these findings, a tool will be developed using Python that will allow a gas exploration company to locate areas where a stacked shale play is economically viable within the study area. Results will be shared with the CNX Resources Corp. organization by utilizing ArcGIS Server and ArcGIS Online.

3.1 Objectives & Key Research Questions

Objectives and key research questions for this project include:

- Where and to what extent is forest fragmentation occurring?
- Where are locations that a stacked well pad could be both viable and profitable in Pennsylvania?
- What impact does a stacked well pad have on reducing habitat fragmentation?
- How can GIS be better utilized to ensure a stacked well pad is viable, developed on time, and within budget?

3.2 Study Area

Susquehanna County was selected as the study area for process 1 to understand Forest Fragmentation resulting from Oil & Gas Exploration (Figure 11). Washington and Susquehanna counties will be the study area for Process 2 & 3, where a gas well production analysis will be performed on production data, and a tool will be developed based on the findings. These two counties were selected for several reasons. First, they are two of the more active counties in terms of permitted wells and gas production. Second, they both have had past environmental issues that could be attributed to oil & gas activity (Dunkard Creek fish kill in Washington County and water contamination in Dimock, Susquehanna County). Finally, they are on opposite sides of the state. Shale formations that are profitable in one county may not be as viable in another.

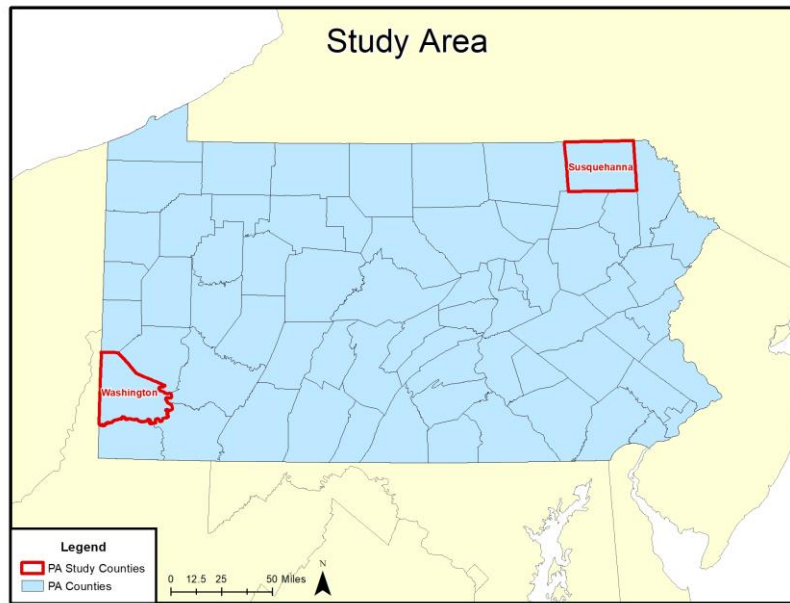


Figure 11: A map defining this project's study area

3.3 Methodology

The methodology of this project consists of data management and three main processes:

- Data Management
 - Well Production Dataset
 - Digitizing Drilling Units & Generating the Study Area for Process 1
- Process 1: Forest Fragmentation Analysis
- Process 2: Well Production Data Analysis
- Process 3: Develop Tool based on Findings

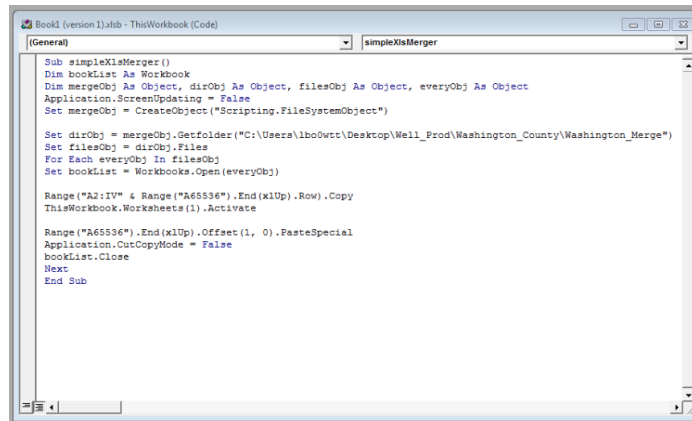
3.4 Data Management

To prepare for the three main processes associated with the project's methodology, data management must first be performed. This consists of the creation of a well production feature class by shale formation, and the digitizing of drilling units and the generations of the study area for process 1.

3.4.1 Well Production Dataset

The first data management process is to generate a well feature class from the reported Pennsylvania DEP well production data. Only unconventional or tight gas wells from 2005 to 2013 will be considered. Production data was recorded and made public as semi-annual reports. Unconventional production was separated from conventional production beginning in July 2008. Beginning in January 2015, the

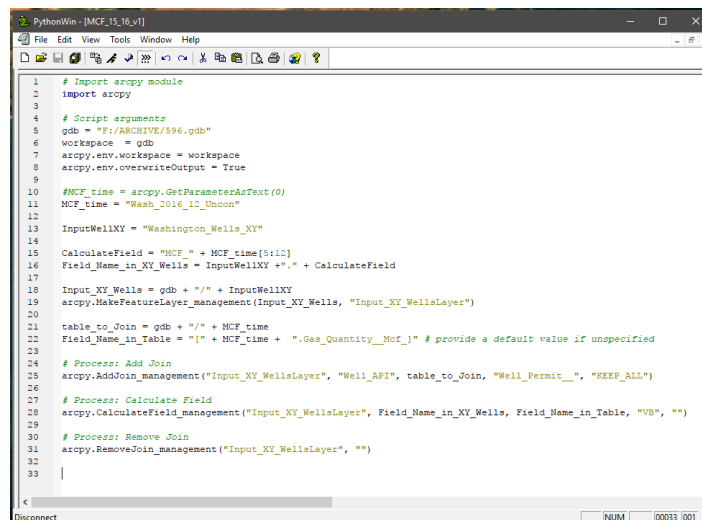
Pennsylvania DEP required production companies report results on a monthly basis. A total of 34 .xls reports were exported from the DEP website per county. A VB Macro was developed to merge the 34 reports together into one .xls spreadsheet (Figure 12).



```
Sub simpleXlsMerger()  
Dim bookList As Workbook  
Dim mergeObj As Object, dirObj As Object, filesObj As Object, everyObj As Object  
Application.ScreenUpdating = False  
Set mergeObj = CreateObject("Scripting.FileSystemObject")  
  
Set dirObj = mergeObj.GetFolder("C:\Users\lboDvrt\Desktop\Well_Prod\Washington_County\Washington_Merge")  
Set filesObj = dirObj.Files  
For Each everyObj In filesObj  
Set bookList = Workbooks.Open(everyObj)  
  
Range("A2:IV" & Range("A65536").End(xlUp).Row).Copy  
ThisWorkbook.Worksheets(1).Activate  
  
Range("A65536").End(xlUp).Offset(1, 0).PasteSpecial  
Application.CutCopyMode = False  
bookList.Close  
Next  
End Sub
```

Figure 12: VB Macro used to merge the 34 production .xls files into one spreadsheet

Duplicate permit numbers (API) were removed, leaving one row for all unconventional wells permitted during the timeframe (2005 to 2013). This .xls can now be converted into a feature class in a File Geodatabase using Feature Manipulation Engine (FME) Desktop (convert .xls to FileGDB) by the recorded latitude and longitude. Fields were added to store the semi-annual or monthly production values. The 34 .xls reports were then imported into the FileGDB as tables and a Python script was developed (Figure 13) to automate the join of the table with the production data to the feature class.



```
1 # Import arcpy module  
2 import arcpy  
3  
4 # Script arguments  
5 gdb = "F:/ARCHIVE/596.gdb"  
6 workspace = gdb  
7 arcpy.env.workspace = workspace  
8 arcpy.env.overwriteOutput = True  
9  
10 #MCF_time = arcpy.GetParameterAsText(0)  
11 MCF_time = "Wash_2016_10_Unknown"  
12  
13 InputWellXY = "Washington_Wells_XY"  
14  
15 CalculateField = "MCF" + MCF_time[5:12]  
16 Field_Name_in_XY_Wells = InputWellXY + "." + CalculateField  
17  
18 Input_XY_Wells = gdb + "/" + InputWellXY  
19 arcpy.MakeFeatureLayer_management(Input_XY_Wells, "Input_XY_WellsLayer")  
20  
21 table_to_Join = gdb + "/" + MCF_time  
22 Field_Name_in_Table = "[" + MCF_time + ".Gas_Quantity_Mcf_]" # provide a default value if unspecified  
23  
24 # Process: Add Join  
25 arcpy.AddJoin_management("Input_XY_WellsLayer", "Well_API", table_to_Join, "Well_Permit_", "KEEP_ALL")  
26  
27 # Process: Calculate Field  
28 arcpy.CalculateField_management("Input_XY_WellsLayer", Field_Name_in_XY_Wells, Field_Name_in_Table, "VB", "")  
29  
30 # Process: Remove Join  
31 arcpy.RemoveJoin_management("Input_XY_WellsLayer", "")  
32  
33
```

Figure 13: Python script used to automate the population of the production data into the feature class

Within this script, a HIGHEST_MCF_PRODUCTION field was created and attributed using an update cursor. The update cursor will search through yearly production data by MCF and update this field with the highest annual production value. This field will be useful to compare well production, as some wells may be active and producing for a year while others could be active and producing for multiple years.

As observed in the literature review, wellpads are oftentimes comprised of multiple permitted horizontal wells originating from the same pad location. To perform the point pattern analysis of wellpad locations within process 2, wellpad geographic centers were created using the [Median Center](#) geoprocessing tool within ArcGIS Desktop (Figure 14).



Figure 14: Median Center geoprocessing tool used to generate wellpad centers

3.4.2 Digitizing Drilling Units & Generating the Study Area for Process 1 (check)

The next step to prepare the datasets was the creation of a drilling units feature class. Drilling units digitized were from Declaration of Unitization documents, which were recorded in the county courthouse (figure 15).

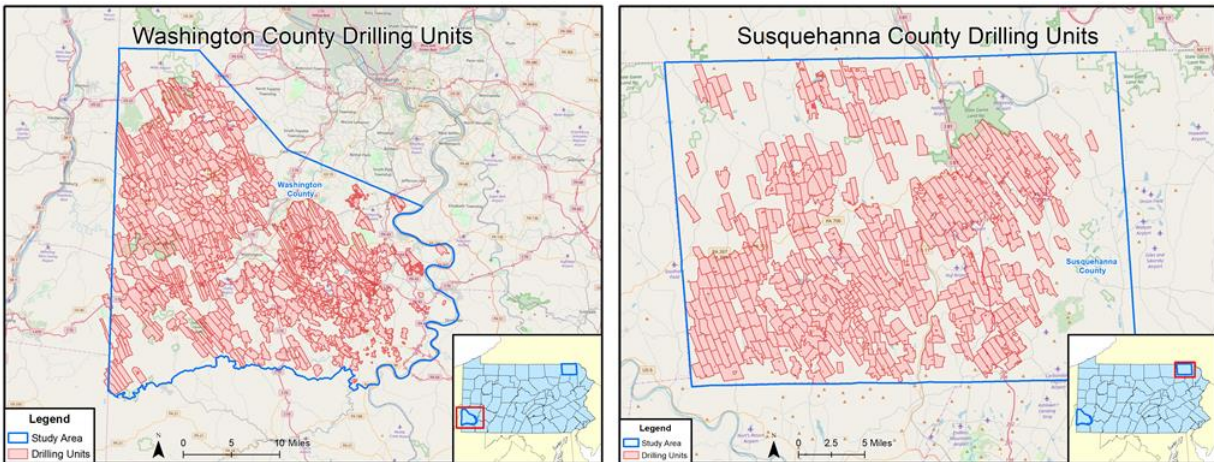


Figure 15: Digitized drilling units in Washington and Susquehanna counties

This drilling unit feature class was incorporated into the development of the study area in Susquehanna County to perform the forest fragmentation analysis. To better understand the effects of forest fragmentation that can be attributed to natural gas activity a study area will be created and comparisons will be conducted.

This study area consists of a 500-meter buffer around recorded drilling units of producing well pad locations and pipeline datasets. Esri ModelBuilder was used to assist in this process (Figure 16).

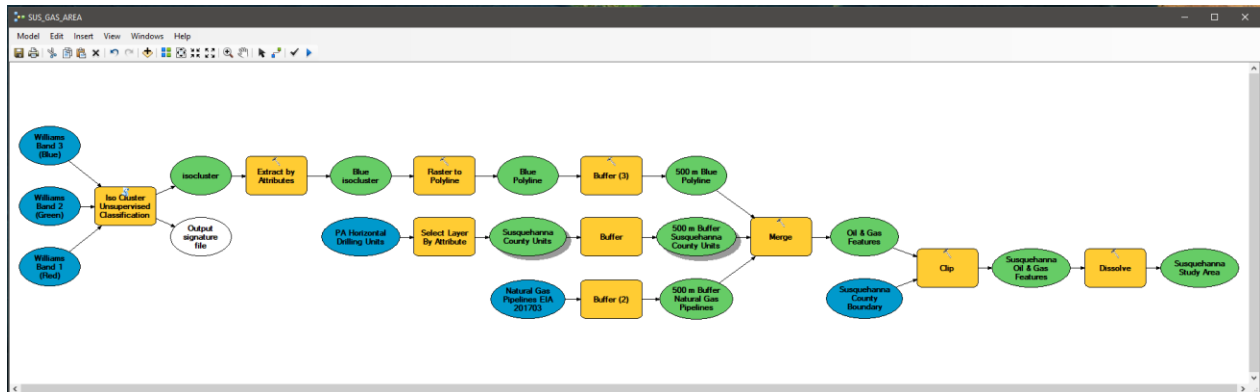


Figure 16: GIS workflow developed in ModelBuilder to create the Susquehanna County study area that has potential oil & gas activity

Midstream companies do not want their non-FERC regulated spatial datasets to be publicly available. A goal of this project is to perform analysis on forest fragmentation using only public and readily available information. Williams Partners L.P. existing Susquehanna County gathering lines were made publicly available as a .pdf as part of their approval for their [Atlantic Sunrise pipeline project](#). This .pdf will be used to locate their gathering lines in the county. This image was georeferenced and rectified. The [Iso Cluster Unsupervised Classification](#) tool was incorporated into the model to classify the three bands of this raster image. The goal was to separate the blue pipelines into a class and incorporate these areas into our study area. 20 separate classes were created and class 6 was the blue pipelines. This class was extracted, converted into a vector dataset and buffered with the other oil & gas features. The resulting study area is seen in Figure 17.

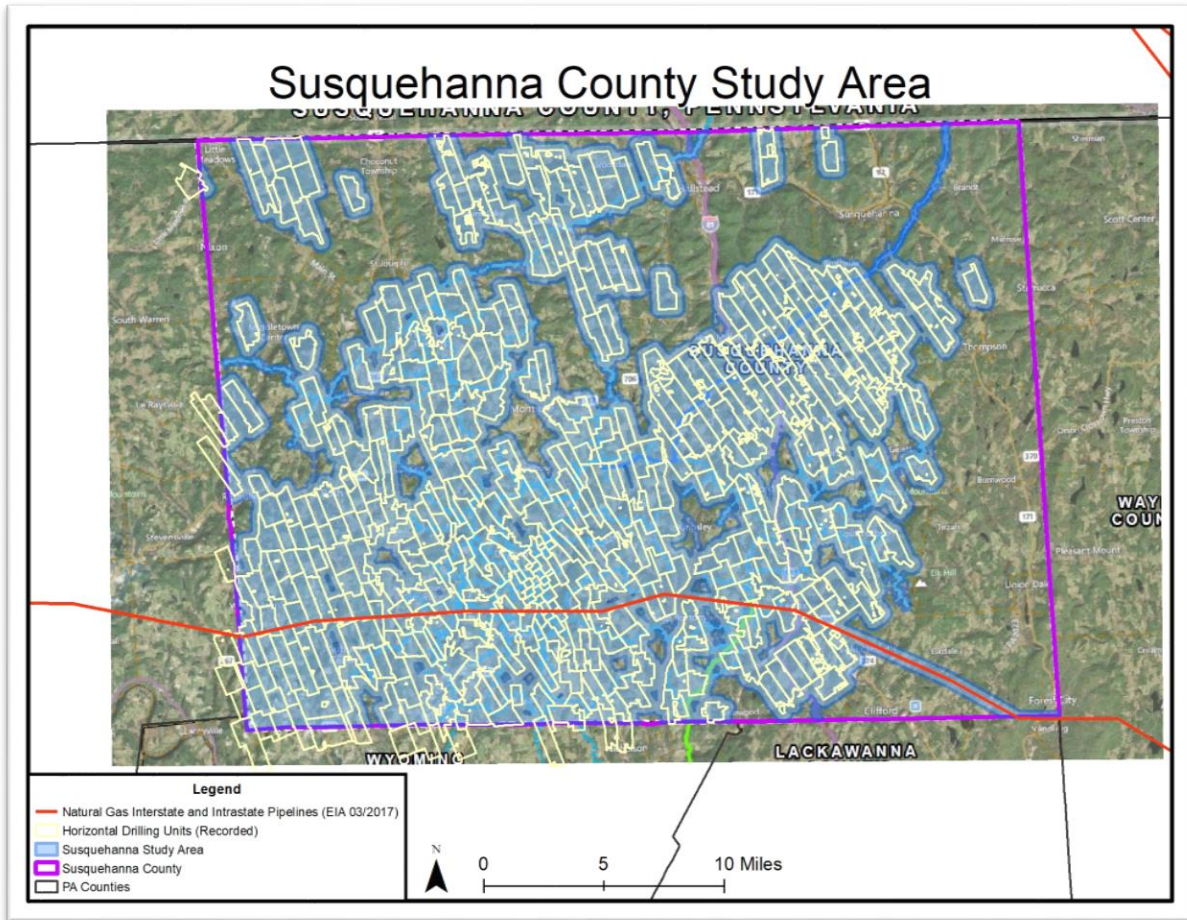


Figure 17: Susquehanna County study area with potential oil & gas activity shown as the blue region.

3.5 Process 1: Forest Fragmentation Analysis

For process 1, a Python script was planned to be developed to perform the first two steps. First, a [reclassification](#) of both land cover datasets for both the 2005 and 2013 land cover datasets will be performed. The output raster values will be set as 0 = not analyzed, 1 = non-forest, and 2 = forest. Second, the University of Connecticut's Landscape Fragmentation Tool (LTF) v 2.0 will be implemented into the Python script to categorize the forested areas into four main categories - patch, edge, perforated, and core. Acreages will be calculated for each category. Finally, this paper will analyze Local Indicators of Spatial Association (LISA) using GeoDa to understand the effects of fragmentation in the following:

- A. Forest fragmentation per drilling unit
- B. Forest fragmentation Municipality
- C. Forest loss per 1 km x 1 km grid

3.5.1 Landscape Fragmentation Tool (LFT) v 2.0

The Landscape Fragmentation Tool (LFT) v 2.0 was developed by Vogt et al. (2007) and classifies a land cover type of interest into 4 main categories - patch, edge, perforated, and core. The edge width for this analysis will be 100 meters, which is often used for general purpose analyses (Kiviat, 2013, p. 1-14). The core category was further divided into small core, medium core, and large core based on the area of the core tract.

- small core patches have an area of fewer than 250 acres
- medium core patches have an area between 250 and 500 acres
- large core patches have an area greater than 500 acres

Examples of Fragmentation Classes (Figure 18) include:

- Core (interior): occurs outside of the "edge effect" zone and so is not degraded by fragmentation.
- Perforated: occurs within the "edge effect" zone along the edge of a small clearing in a non-patch tract (example: the forested area surrounding the cleared house lot and enclosed within the boundary).
- Edge: occurs within the "edge effect" zone along the outside edge of a non-patch tract (example: the forested area along an urbanized region and enclosed within the boundary).
- Patch: small fragments that are completely degraded by the "edge effect" (example: the small woodlots enclosed within the boundaries).



Figure 18: Examples of fragmentation classes (Vogt et al., 2007)

This tool was readily available as a Python script tool within an Esri Toolbox (figure 19).

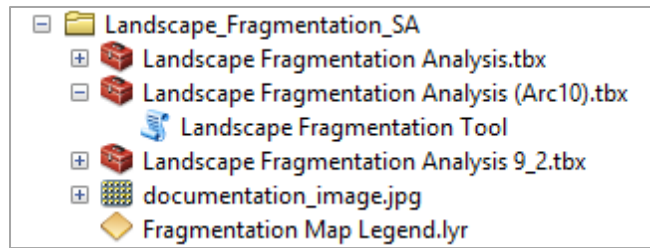


Figure 19: Esri Toolbox for the Landscape Fragmentation Tool (LFT) v 2.0

This tool should have been able to be implemented into this GIS workflow seamlessly, but a 9999 error occurred while the tool was processing. The exact cause of the error has not been confirmed (although it is hypothesized that a change in precision within the reclassification geoprocessing tool from ArcGIS 10, where the fragmentation tool was developed and tested, to ArcGIS 10.4 could be the source of the error). Nevertheless, this procedure was a critical step in this project’s analysis, and a workaround was developed. A model (figure 20) was developed in Esri Modelbuilder, which progress through Vogt’s script and workflow to create the four forest classes based on the input raster datasets. Reclassification, Euclidian Distance, Set Null, Zonal Statistics, Region Group, Plus, and Times geoprocessing tools were all incorporated into this workflow. ArcGIS Pro’s 64-bit geoprocessing helped speed up the progression of this model.

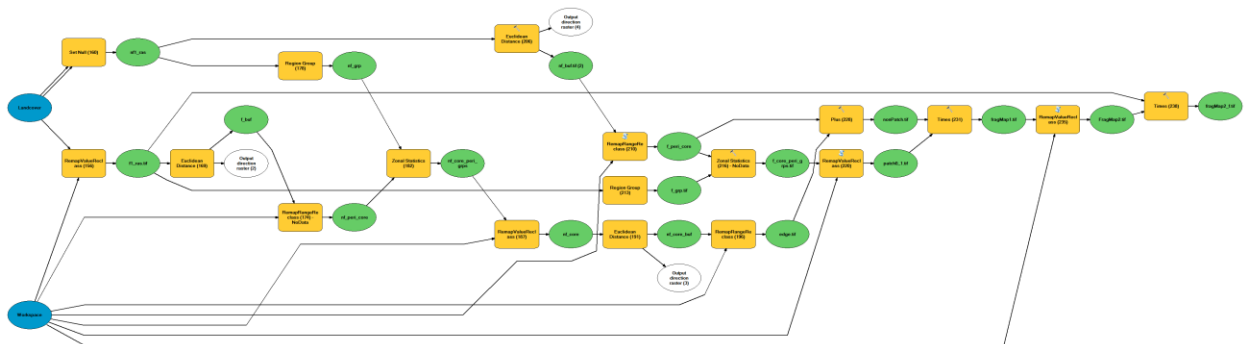


Figure 20: Model developed in Esri Modelbuilder that progress through Vogt’s workflow

3.5.2 Susquehanna County Fragmentation Webmap

To better display the results of the fragmentation analysis, a Forest Fragmentation Webmap was developed. The fragmentation classes from 2005 and 2013 were uploaded as map services to the organization’s ArcGIS Server. The web application was created using the Web AppBuilder for ArcGIS. One key feature of the application is a horizontal slider, which allows the user to compare the changes in fragmentation classes from 2005 (left) to 2013 (right). Only users within the CNX’s organization have access to the web application, but an example video showing the user interface can be viewed [here](#).

3.5.3 Susquehanna County Fragmentation Observations

When comparing fragmentation occurring within and outside the study area. There is a clear increase in forest fragmentation occurring near areas of oil & gas activity in Susquehanna County (Figure 21). The fragmentation classes that experienced that greatest change in the study area during the analyzed timeframe were perforated and patch. It should be noted, there was a slightly greater percent forest loss to core forest outside the study area.

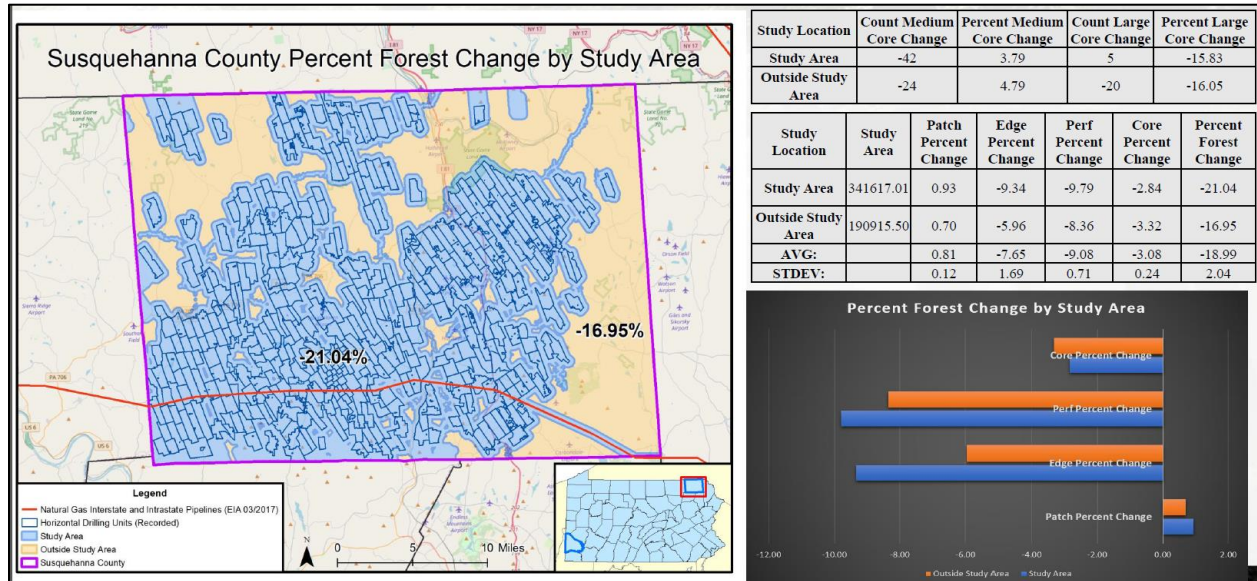


Figure 21: Comparison of forest fragmentation inside and outside the study area

To better understand how individual gas exploration companies contribute to forest fragmentation, drilling units were dissolved by gas exploration company per the recorded oil & gas document. Companies were given an alias of Company A through Company H based on the area of their drilling unit's footprint. A fragmentation analysis was performed based on these areas. All exploration companies contributed negatively to overall forest fragmentation and caused increased forest patches. Based on the results (Figure 22), Company D appears to have the best practices and contributed the least percent to forest fragmentation, while company H appears to have the worst practices and contributed the most to both percent forest and core loss.

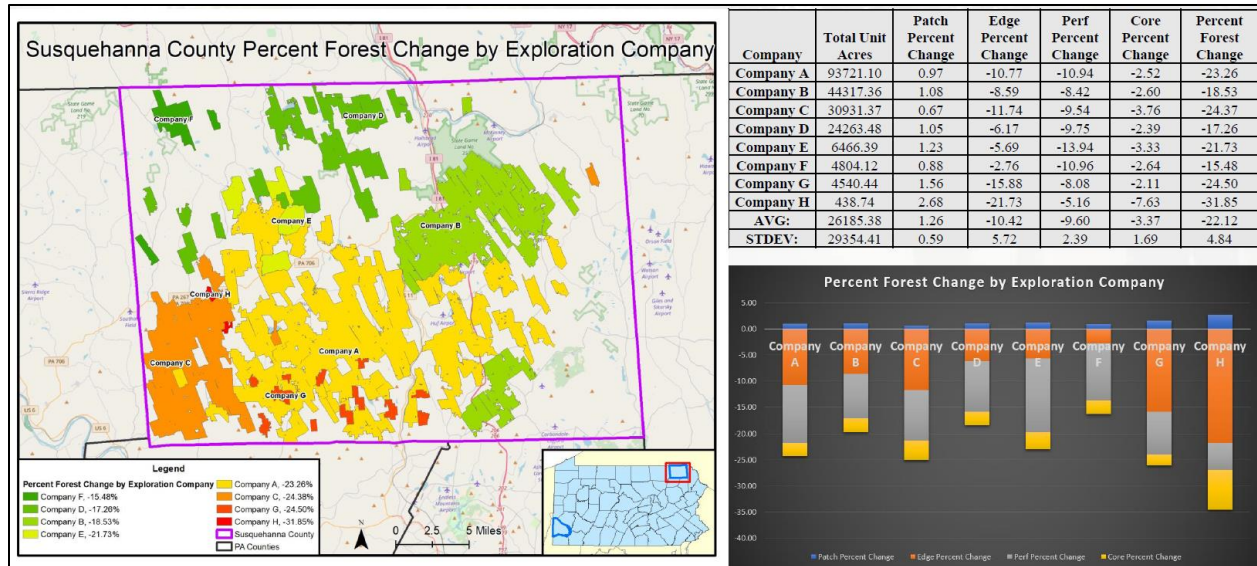


Figure 22: Forest fragmentation by gas exploration company

3.5.4 Local Indicators of Spatial Association (LISA)

To better understand the fragmentation that is occurring in Susquehanna County, the next step in this process was an analysis of local indicators of spatial association (LISA). Using GeoDa, which is a free, open-source software package that conducts spatial data analysis, geovisualization, spatial autocorrelation and spatial modeling. This analysis seeks to understand how forest fragmentation has occurred based within the following areas:

- Forest fragmentation per drilling unit
- Forest fragmentation Municipality
- Forest loss per 1 km x 1 km grid

To prepare these datasets, a python script (see [Appendix A, Script 1: Source code for LISA Analysis Areas in Python](#)) was developed. The workflow of this script is the following steps. First and intersection will be performed of the area and fragmentation feature class (lines 34-35). Next, a Dissolve (by fragmentation class will be performed (lines 37-38). Acreages will be calculated of the four fragmentation classes (lines 40-42). Then, a join will be conducted to transfer fragmentation values (lines 44-64). Null values will be removed using an update cursor (lines 207-215) to ensure proper calculations. Finally, percent change and acreage change will be calculated by fragmentation class (lines 218-235).

With the fragmentation values now calculated by the python script for the areas, the output datasets can be analyzed within GeoDa (O'Sullivan, 2014, p. 150-151). The analysis will be performed on total percent forest loss, from all four forest classes, and percent loss to core forest. As observed in the literature review, core forest is crucial to the biodiversity in the region. First, the forest fragmentation per drilling unit for all four forest classes will be analyzed (Figure 23).

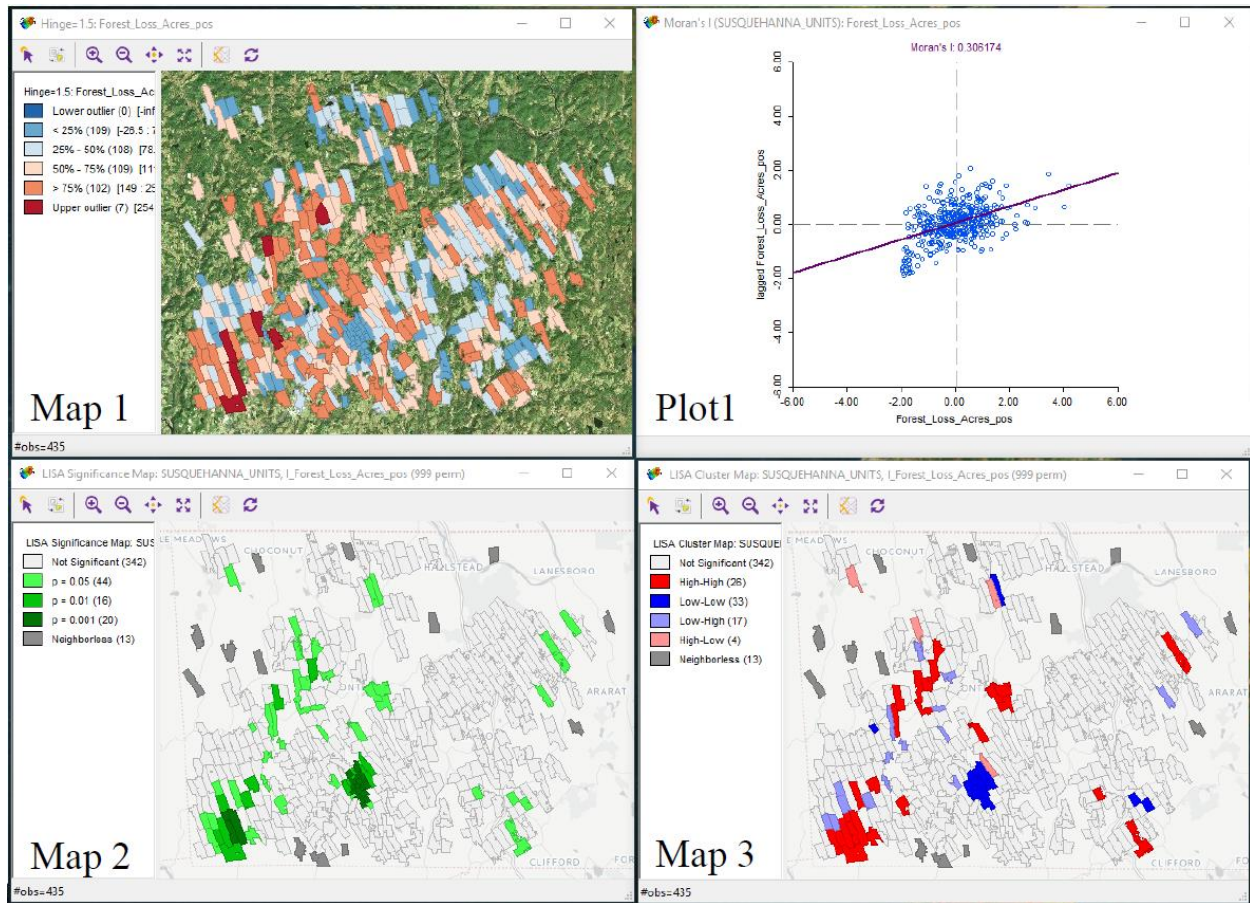


Figure 23: LISA analysis of percent total forest loss by drilling unit

Map 1 is a box map of Susquehanna County, PA drilling units symbolized by percent forest loss, which symbolized units with greater forest loss in red while units with lesser forest loss in blue. Plot 1 is a Moran's I scatterplot. A Moran's I index score range from -1 to 1, where a negative index score represents a negative correlation while a positive index score represents a positive correlation. Generally, index scores 0.3 or more, or -0.3 or less, are indicative of a relatively strong autocorrelation. For this dataset, Moran's I index score of 0.306, which is a positive value greater than 0.3, is indicative of a positive spatial autocorrelation. Map 2 is the LISA significance map, which shows the significance level of the contributions of each drilling unit to the autocorrelation. These values were determined by performing Monte Carlo simulations for 999 permutations. Areas in darker shades of green contribute to the local significance, while areas in white are non-significant locations. Map 3 is the LISA cluster map. This map displays drilling units that significantly contributed to either positive or negative autocorrelation. The drilling units in the southwest region of the county tend to have higher percent forest loss and have neighboring drilling units with high forest loss (high-high), which are shown in the darker red. Similarly, drilling units in the south-central parts of the county have low forest loss and have neighboring drilling units with low forest loss (low-low). These areas are symbolized in a darker blue. Both these areas (the high-high and low-low) contributed to the positive autocorrelation. For this

dataset, there was only 4 light pink (high-low) drilling units and 17 light blue (low-high) drilling units that contributed to a negative autocorrelation. For this dataset, using LISA and GeoDa, one can dismiss spatial randomness and can locate and characterize the clusters of drilling units by percent forest loss.

The drilling units will again be analyzed by percent core forest in figure 24.

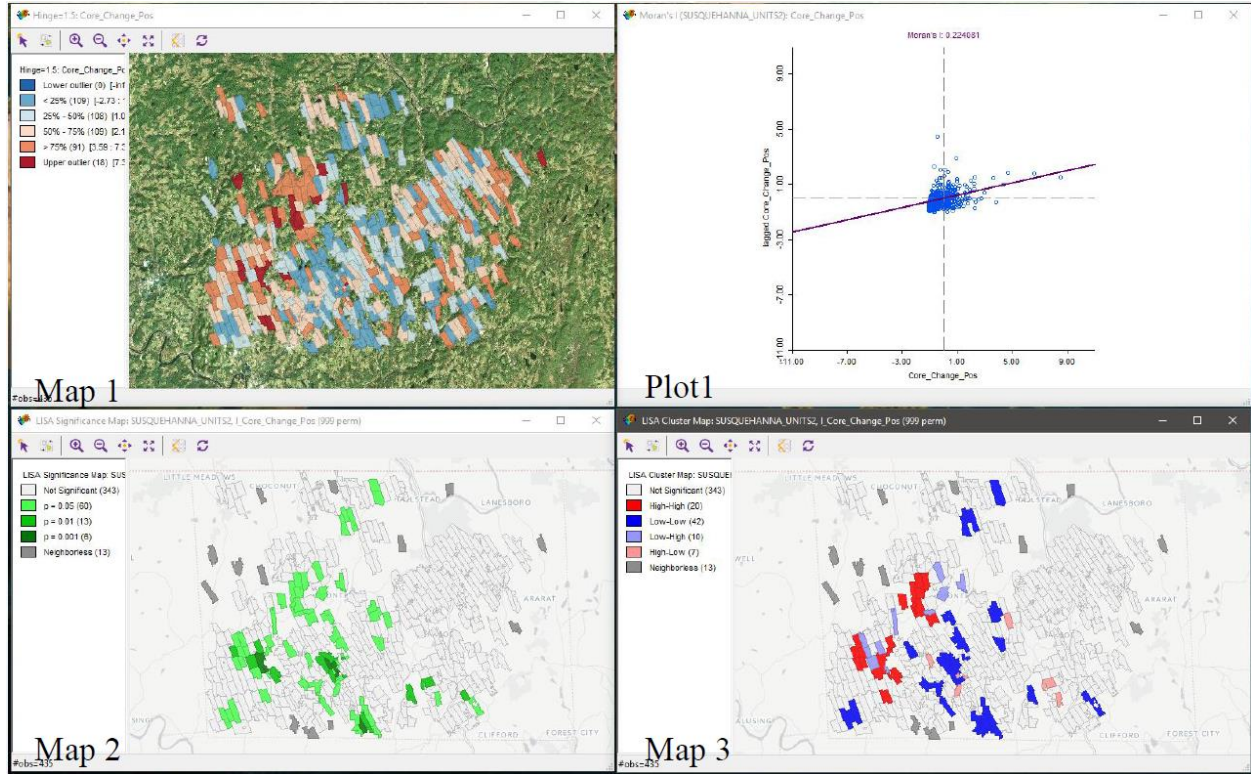


Figure 24: LISA analysis of percent core forest loss by drilling unit

Map 1 is a box map of drilling units symbolized by percent core forest loss. Plot 1 is a Moran's I scatterplot with a score of 0.224, which is indicative of a weak positive spatial autocorrelation. Map 2 is the LISA significance map, which shows the significance level of the contributions of each drilling unit to the autocorrelation with darker shades of green contributing to the local significance, while areas in white are non-significant locations. Map 3 is the LISA cluster map showing drilling units that significantly contributed to either positive or negative autocorrelation. In this instance, core forest loss by units appears to be irregular across the county, which is indicative of a lower Moran's I score.

The next area that will be analyzed in a similar fashion is percent forest loss by Susquehanna County municipality to all four forest classes (Figure 25).

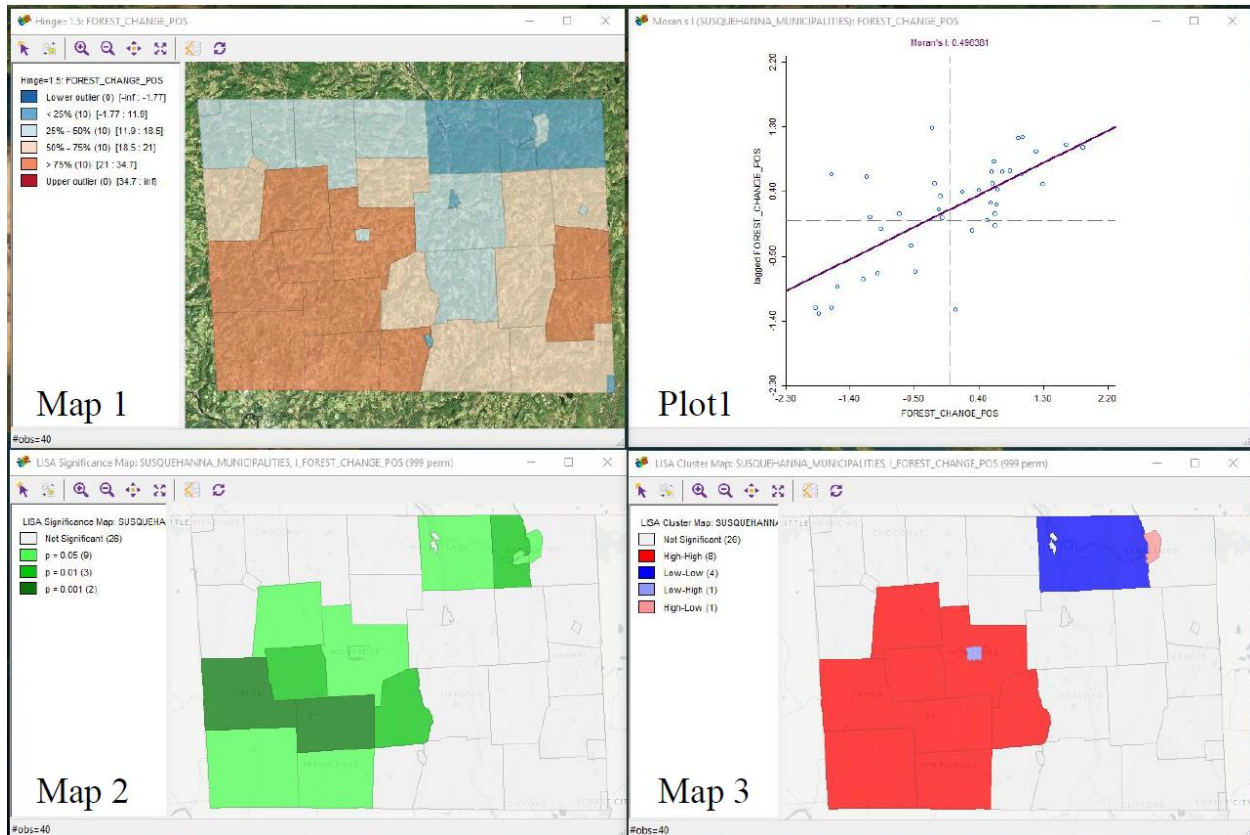


Figure 25: LISA analysis of total percent forest loss by municipality

Map 1 is a box map of municipalities symbolized by percent forest loss. Plot 1 is a Moran's I scatterplot. For this instance, Moran's I index score of 0.406, which is indicative of a positive spatial autocorrelation. Map 2 is the LISA significance map, which shows the significance level of the contributions of each municipality to the autocorrelation. Again, areas in darker shades of green contribute to the local significance, while areas in white are non-significant locations. Map 3 is the LISA cluster map. This shows municipalities that significantly contributed to either positive or negative autocorrelation. The municipalities in the southwest region of the county tend to have higher percent forest loss and have neighboring municipalities with high forest loss (high-high), which are shown in the darker red. Most of these municipalities fall within the study area. Similarly, municipalities in the northeast of the county have low forest loss and have neighboring drilling units with low forest loss (low-low), and are symbolized in a darker blue. Most of these municipalities fell outside the study area. Both these areas (the high-high and low-low) contributed to the positive autocorrelation. For this dataset, there was only 1 light pink (high-low) municipality and 1 light blue (low-high) municipality that contributed to negative autocorrelation. For this dataset, one can dismiss spatial randomness and can locate and characterize the clusters of municipalities by percent forest loss.

The municipalities will again be analyzed by percent core forest in figure 26.

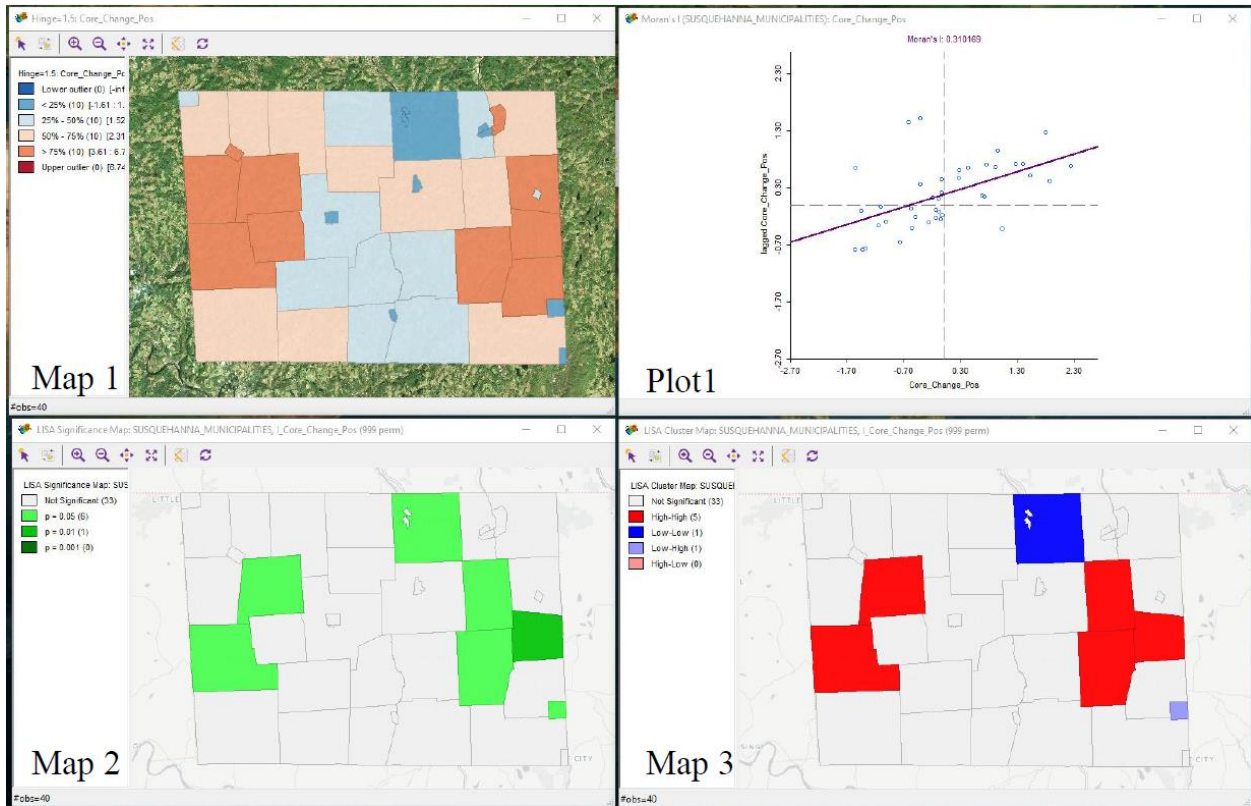


Figure 26: LISA analysis of percent core forest loss by municipality

Map 1 is a box map of municipalities symbolized by percent core forest loss. Plot 1 is the Moran's I scatterplot with a score of 0.310, which is indicative of a positive spatial autocorrelation. Map 2 is the LISA significance map, which shows the significance level of the contributions of each municipality to the autocorrelation with darker shades of green contributing to the local significance, while areas in white are non-significant locations. Map 3 is the LISA cluster map that shows municipalities that significantly contributed to either positive or negative autocorrelation. It is important to note, municipalities in the eastern parts of the county had higher percent core forest loss and have neighboring municipalities with high forest loss (high-high) even though oil & gas activities are not occurring in these areas. For this dataset, one can dismiss spatial randomness and can locate and characterize the clusters of municipalities by percent core forest loss.

The final area that will be analyzed in a similar fashion is percent forest loss by 1 km by 1 km grid for all four forest classes covering Susquehanna County (Figure 27).

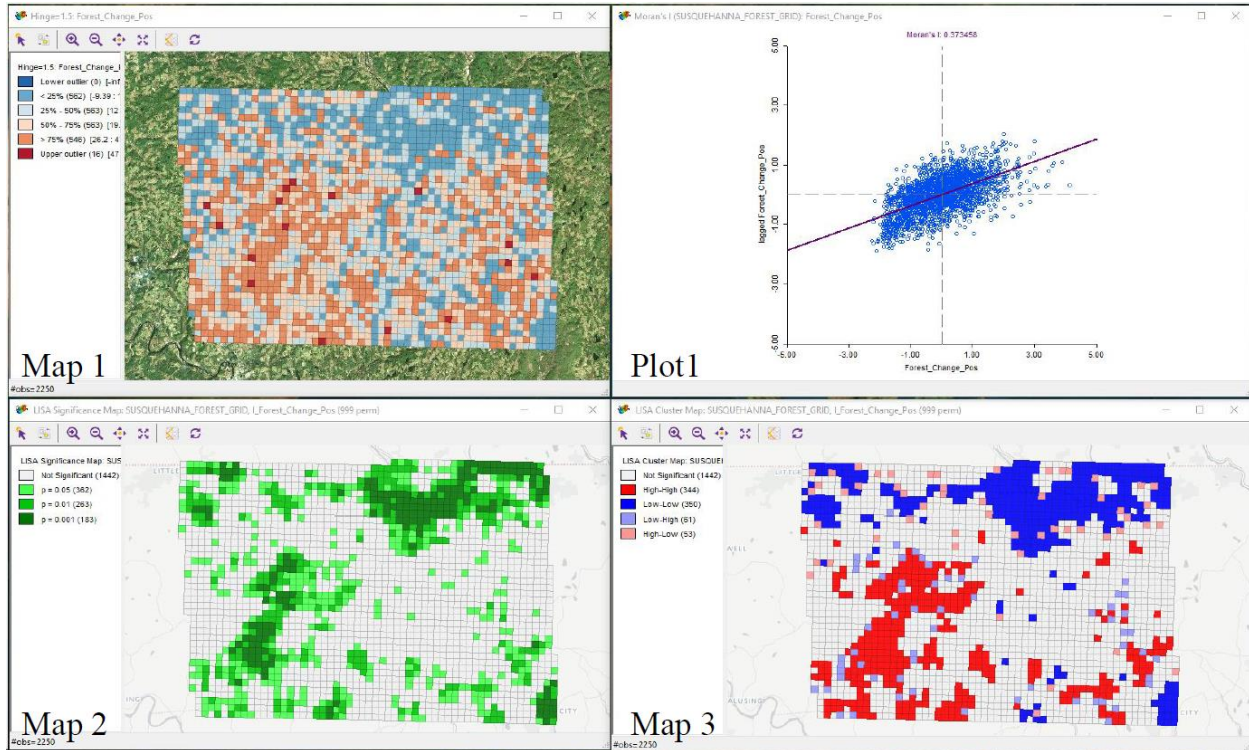


Figure 27: LISA analysis of total percent forest loss by 1 km by 1 km grid

The grids were generated using the [Grid Index Features](#) Geoprocessing tool in ArcMap. Map 1 is a box map of the grids symbolized by percent forest loss. For Plot 1, the Moran's I scatterplot I index score of 0.406, which being greater than 0.03 is indicative of a positive spatial autocorrelation. Map 2 is the LISA significance map, which shows the significance level of the contributions of each grid to the autocorrelation. Again, areas in darker shades of green contribute to the local significance, while areas in white are non-significant locations. Map 3 is the LISA cluster map, which shows grids that significantly contributed to either positive or negative autocorrelation. The grids in the southwest region of the county tend to have higher percent forest loss and have neighboring grids with high forest loss (high-high), which is shown in the darker red. Most of these grids fall within the study area. Similarly, grids in the northeast of the county have low forest loss and have neighboring grids with low forest loss (low-low). These areas are symbolized in a darker blue. Most of these grids fell outside the study area. Both these areas (the high-high and low-low) contributed to the positive autocorrelation. For this dataset, there was 53 light pink (high-low) grids and 61 light blue (low-high) grids that contributed to negative autocorrelation. Again, for this dataset, one could dismiss spatial randomness and can locate and characterize the clusters of grids by percent forest loss.

The 1 km by 1 km grids will again be analyzed by percent core forest in figure 28.

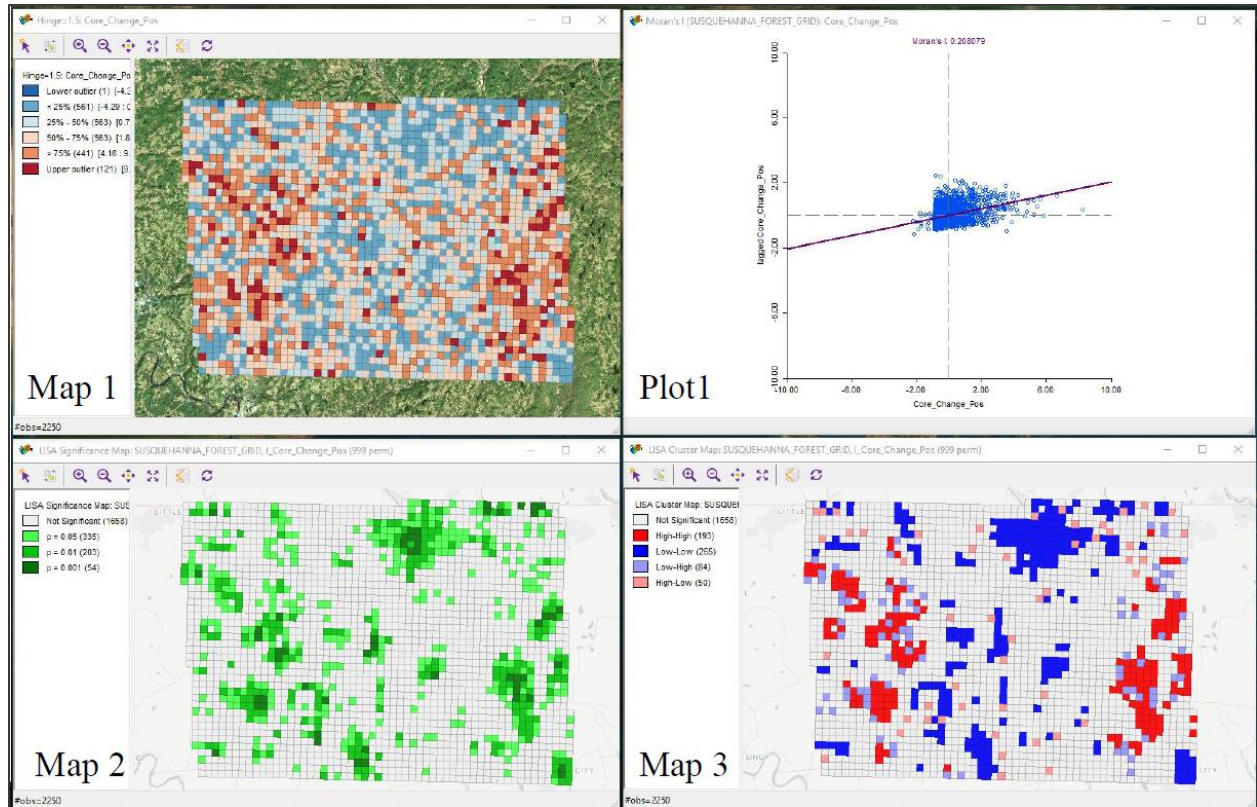


Figure 28: LISA analysis of percent core forest loss by 1 km by 1 km grid

Map 1 is a box map of a 1 km by 1 km grids symbolized by percent core forest loss. Plot 1 is the Moran's I scatterplot with a score of 0.208, which is indicative of a weak positive spatial autocorrelation. Map 2 is the LISA significance map, which shows the significance level of the contributions of each grid to the autocorrelation with darker shades of green contributing to the local significance, while areas in white are non-significant locations. Map 3 is the LISA cluster map showing grids that significantly contributed to either positive or negative autocorrelation. In this instance, core forest loss by 1 km by 1 km grid appears to be irregular across the county, which is indicative of a lower Moran's I score. It is significant to note, there were many grids in the western parts of the county where higher percent core forest loss occurred that have neighboring municipalities with high forest loss (high-high). There was no oil & gas activity in these areas.

3.6 Process 2: Well Production Data Analysis

Process 2 will consist of four steps. First, a kernel density analysis and Monte Carlo simulations of the G-function() using R will be conducted to understand the point pattern of producing wellpads. Next, well production by formation thickness and depth will be plotted using GeoDa and observations will be made. Then, a LISA analysis for MCF production by formation using GeoDa will be performed. Finally,

Kriging will be completed to locate and predict areas that are most productive by formation within the two study counties.

3.6.1 Point Pattern Analysis using Monte Carlo Simulations

In Figure 29, the plot on the left is a kernel density analysis for Marcellus wellpad locations in Susquehanna County. The plot on the right is the output of the G-function(), which estimates the nearest neighbor distance distribution function $G(r)$ from the point pattern. The source code for this analysis can be found in [Appendix A, Script 2: Source Code for density analysis and Monte Carlo simulation in R](#). This function is enveloped by 99 Monte Carlo simulations (shown as the gray region), so to compare the estimate with the $G(r)$ function to what is expected for the independent random process or complete spatial randomness (IRP/CSR). The black line is the function for the actual pattern for the dataset. When the observed value of the function falls outside of the 99 Monte Carlo simulations envelope, the point pattern is more clustered or more dispersed than what would be observed under the IRP/CSR. Each simulation was run 99 times in hopes that the generated envelopes will be more precise.

For the Marcellus wellpad locations in Susquehanna County, the observed values (black line) of the G-function remained above and outside the Monte Carlo simulation envelope for all r values on the plot. Therefore, it can be concluded for the whole range of the plot, the observed pattern is more clustered than one would expect to be generated by IRP/CSR. Furthermore, areas, where the observed values were observed above the theoretical red line, can be considered a more clustered distribution (O'Sullivan, 2014, p.150-151).

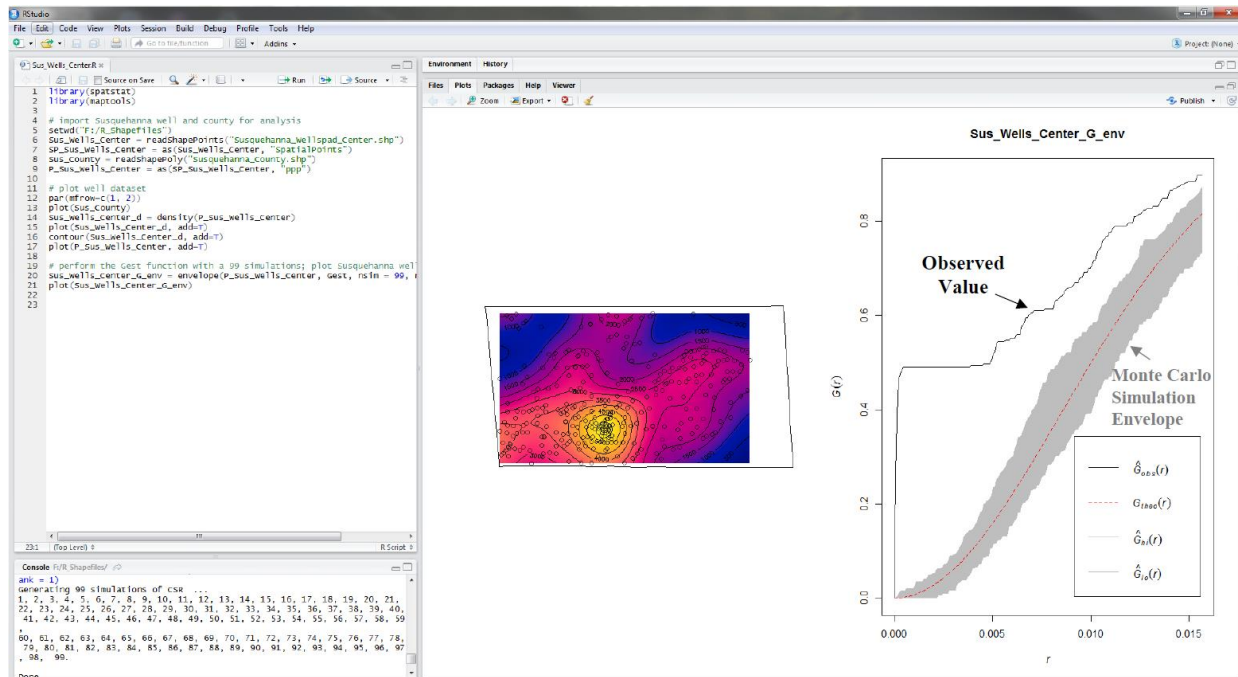


Figure 29: Kernel density analysis Monte Carlo simulation of Susquehanna County Marcellus wellpad centers

In Figure 30, the plot on the left is a kernel density analysis for Marcellus wellpad locations in Washington County, and the plot on the right is the output of the G-function(), which estimates the nearest neighbor distance distribution function $G(r)$ from the point pattern. This function is also enveloped by 99 Monte Carlo simulations so to compare the estimate with the $G(r)$ function to what one would expect to see for IRP/CSR. The actual pattern for the dataset is represented by the black line. When the observed value of the function falls outside of the 99 Monte Carlo simulations envelope, the point pattern is more clustered or more dispersed than what would be observed under the IRP/CSR.

For the Marcellus wellpad locations in Washington County, the observed values (black line) of the G-function remained above and outside the Monte Carlo simulation envelope from $0 - 0.005$ and $0.010 - 0.25$. Therefore, it can be concluded for those r -values, the observed pattern is more clustered than what one would expect to be generated by IRP/CSR. Again, areas, where the observed values were observed above the theoretical red line, can be considered a more clustered distribution (O'Sullivan, 2014, p.150-151).

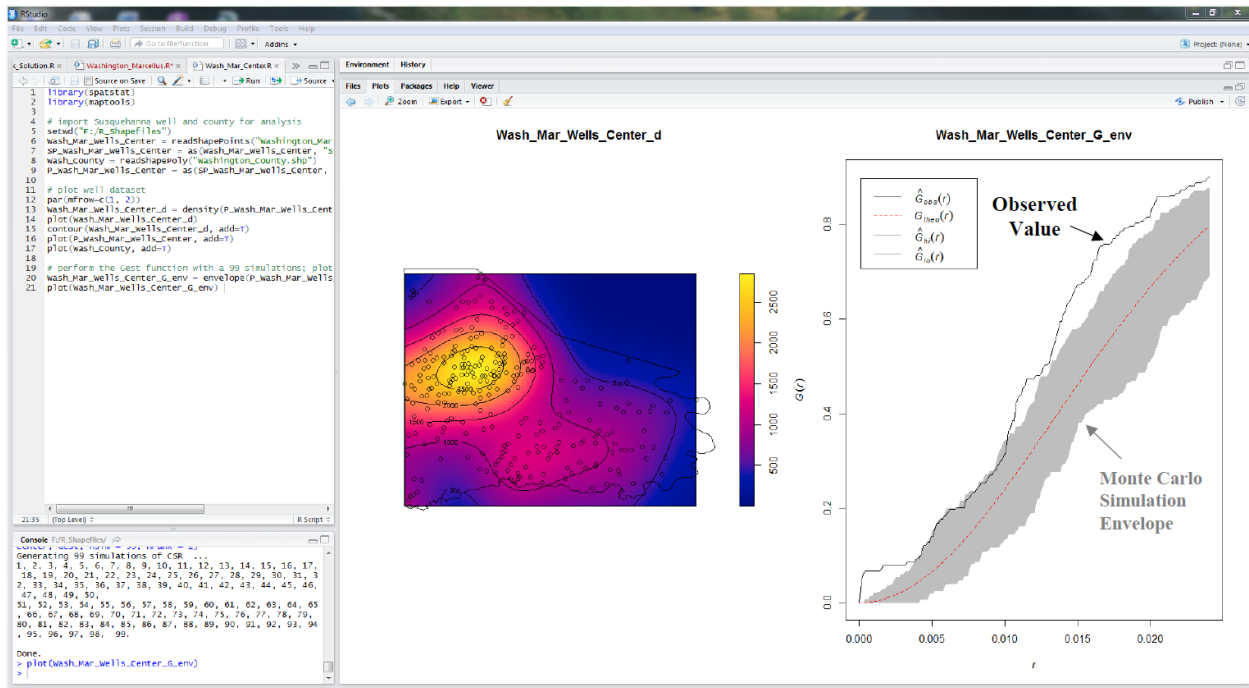


Figure 30: Kernel density analysis Monte Carlo simulation of Washington County Marcellus wellpad centers

Monte Carlo simulations were conducted on wells targeting the Utica and Burket shale formations in Washington County; however, there was not enough information to make a proper conclusion on the point pattern of wells at these depths.

3.6.2 Local indicators of spatial association (LISA) analysis for horizontal wells by highest yearly gas production

Next, this paper will determine if there are local indicators of spatial association (LISA) using GeoDa for both Susquehanna and Washington counties horizontal wells by highest yearly gas production (Figure 31).

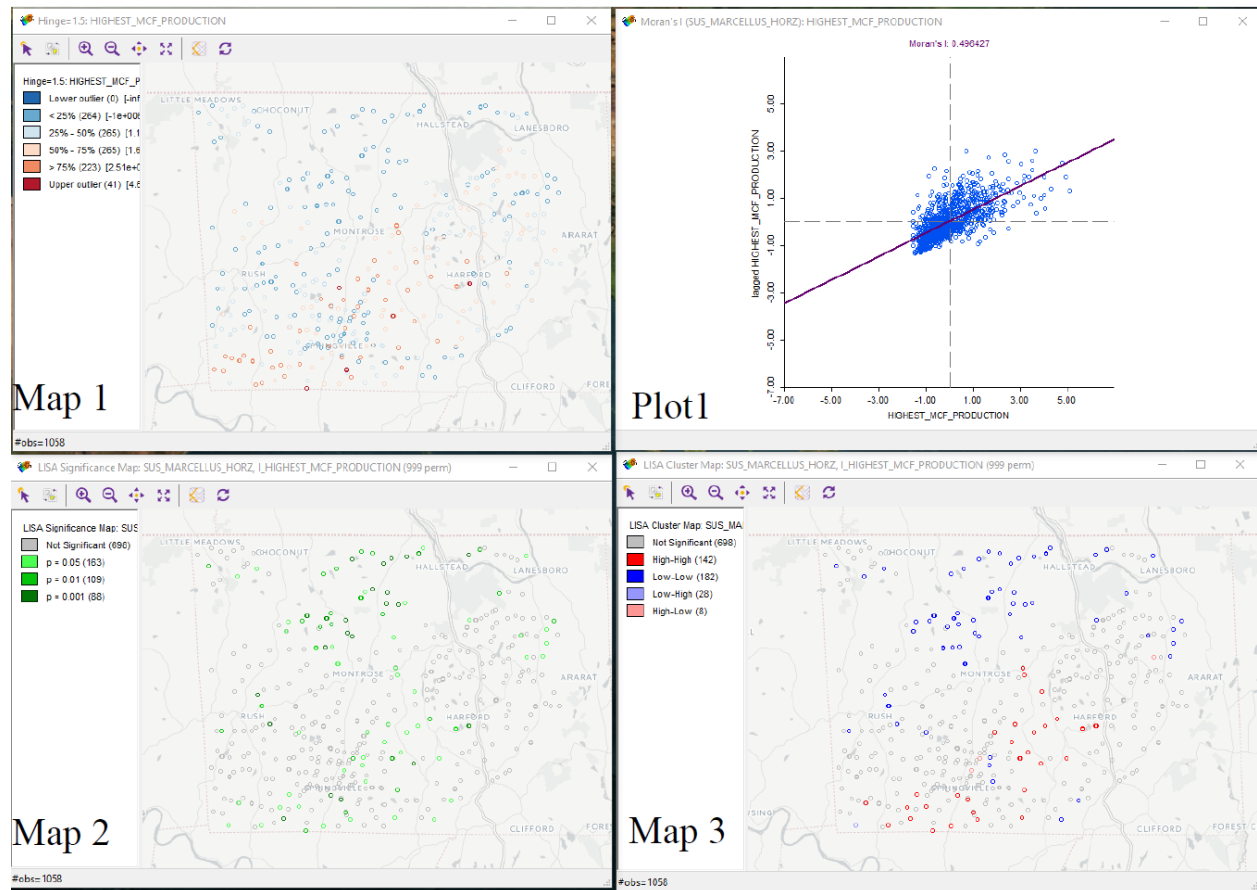


Figure 28: Local indicators of spatial association (LISA) analysis for Susquehanna County, PA horizontal wells by highest yearly gas production

Map 1 is a box map of Susquehanna County, PA horizontal wells by highest yearly gas production. Plot 1 is a Moran's I scatterplot. For this dataset, Moran's I index score of 0.49, which is indicative of a strong positive spatial autocorrelation as seen in plot 1. Map 2 is the LISA significance map, which shows the significance level of the contributions of each well to the autocorrelation. These values were determined by performing Monte Carlo simulations for 999 permutations. Wells in darker shades of green contribute to the local significance, while wells in white are non-significant locations. Map 3 is the LISA cluster map, which shows wells that significantly contributed to either positive or negative autocorrelation. The wells in the southwest regions of Washington County tend to have high production

values and have neighboring wells with high production values (high-high), which is shown in the darker red. Similarly, wells in the northern regions of the county have low production values and have neighboring wells with low production values (low-low), which are symbolized in a darker blue. Both these areas (the high-high and low-low) contributed to the positive autocorrelation. For this dataset, there was 8 light pink (high-low) wells and 28 light blue (low-high) wells that contributed to negative autocorrelation. Therefore, one could dismiss spatial randomness and can locate and characterize the clusters of wells in Susquehanna County based on highest yearly gas production.

In Figure 32, Map 1 is a box map of Washington County, PA horizontal wells by highest yearly gas production.

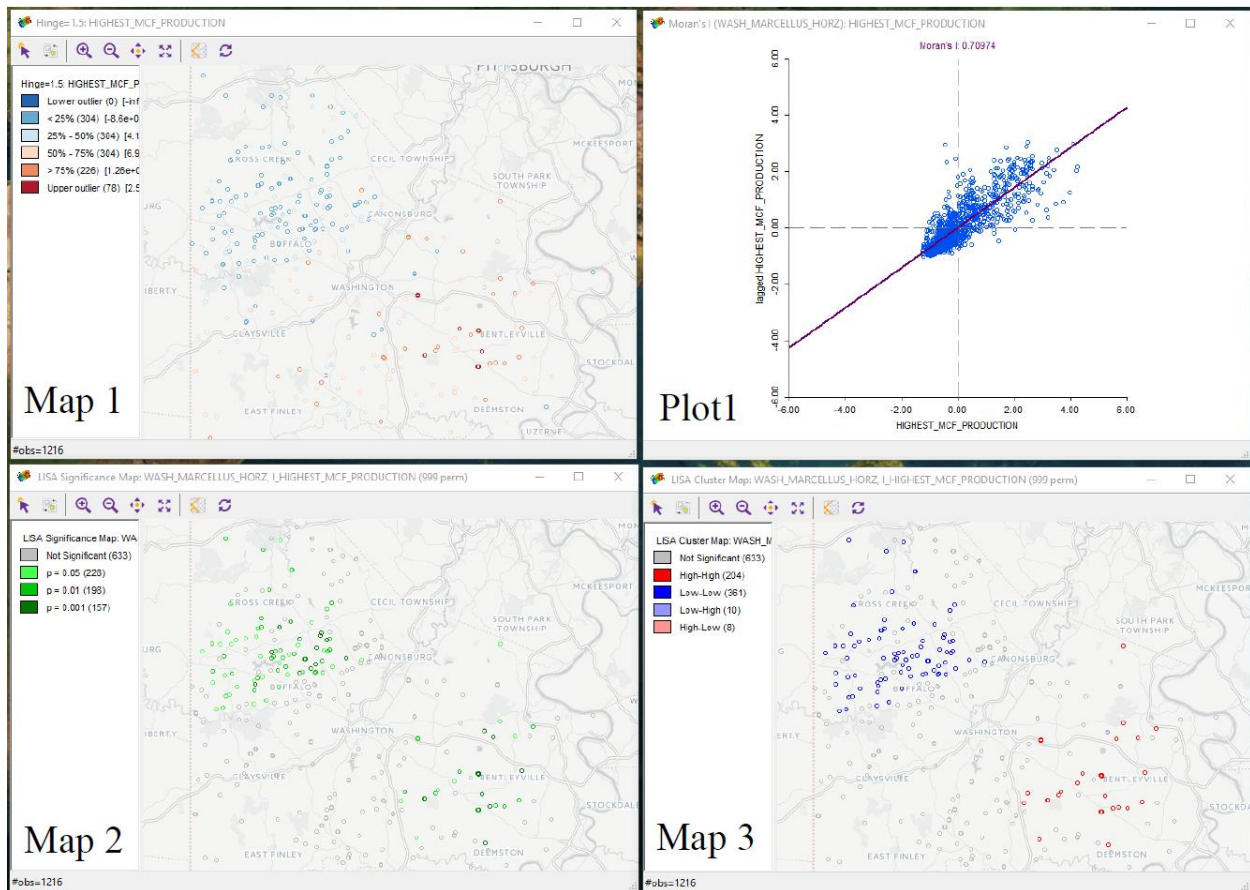


Figure 32: Local indicators of spatial association (LISA) analysis for Washington County, PA horizontal wells by highest yearly gas production

Plot 1 is a Moran's I scatterplot. For this dataset, Moran's I index score of 0.71, which is indicative of a strong positive spatial autocorrelation as seen in plot 1. Again, Map 2 is the LISA significance map, which shows the significance level of the contributions of each well to the autocorrelation. Wells in darker shades of green contribute to the local significance, while wells in white are non-significant locations. Map 3 is the LISA cluster map, which shows wells that significantly contributed to either

positive or negative autocorrelation. The wells in the southeast region of Washington County tend to have high production values and have neighboring wells with high production values (high-high), which is shown in the darker red. Similarly, wells in the northwestern region of the county have low production values and have neighboring wells with low production values (low-low). These areas are symbolized in a darker blue. Both these areas (the high-high and low-low) contributed to the positive autocorrelation. For this dataset, there was 8 light pink (high-low) wells and 10 light blue (low-high) wells that contributed to negative autocorrelation. Again, for this dataset, one could dismiss spatial randomness and can locate and characterize the clusters of grids by percent forest loss.

3.6.3 Well Production by Marcellus Formation Thickness and Depth

Plots were generated using GeoDa to understand highest yearly gas production as a function of shale formation thickness and depth. Higher yearly production was symbolized as larger red circles while lower production was represented in smaller blue circles. In general, it was observed that well production increases with greater formation depth (Figure 33). In Susquehanna County, areas where the Marcellus Shale was deeper and thinner generally resulted in increased MCF production. Decent production results were also observed where shale was thicker but not as deep. In Washington County, deeper Marcellus formation generally led to increased production regardless of shale thickness.

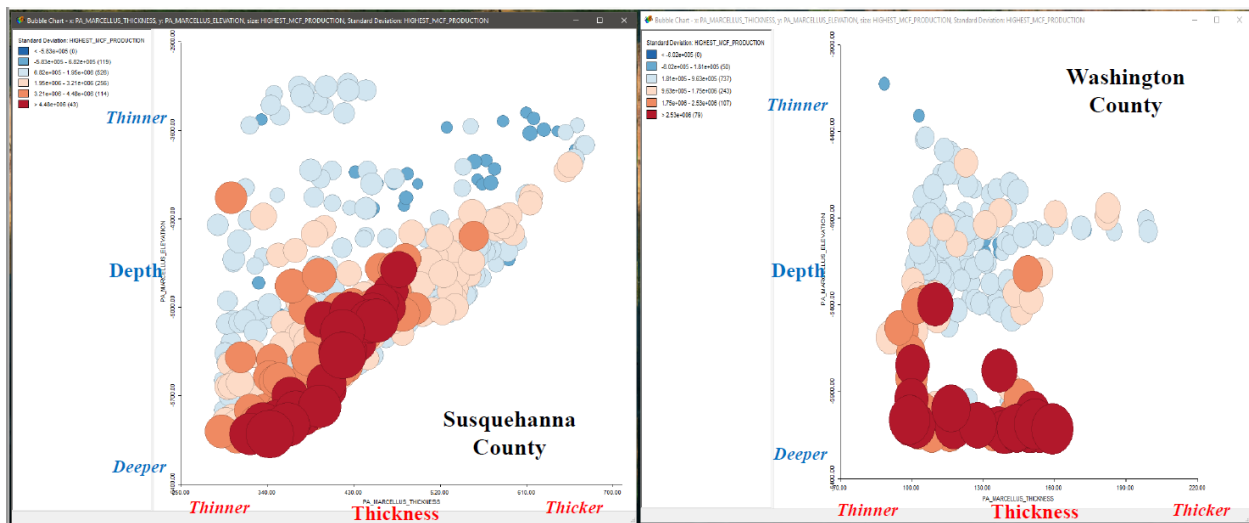


Figure 33: Plot of highest yearly gas production as a function of shale formation thickness and depth in Susquehanna and Washington County

3.6.4 Anisotropic Semivariogram Methods

Kriging is an interpolation technique in which the surrounding measured values are weighted to derive a predicted value for an unmeasured location. This technique will be utilized in this paper to predict well production by formation within the study area. Kriging assumes that the variation in a surface can be broken down into three main components: drift, local spatial autocorrelation, and random stochastic variation.

These characteristics of the semivariogram can be seen graphically in Figure 34:

Still: The semivariance value or amplitude along the y-axis where the variogram levels off.

Range: The distance along the x-axis where the semivariogram reaches the sill value. For distances that are greater than the range, points are likely to be similar and autocorrelation is essentially zero.

Nugget: The value at which the function meets the y-axis. Oftentimes, this value is not at the origin of the graph. Therefore, one can interpret the difference as the measure of random stochastic variation.

These semivariance characteristics are used to solve a series of linear equations whose weights will produce an interpolation that minimizes the amount of error in the predicted values (O'Sullivan, 299).

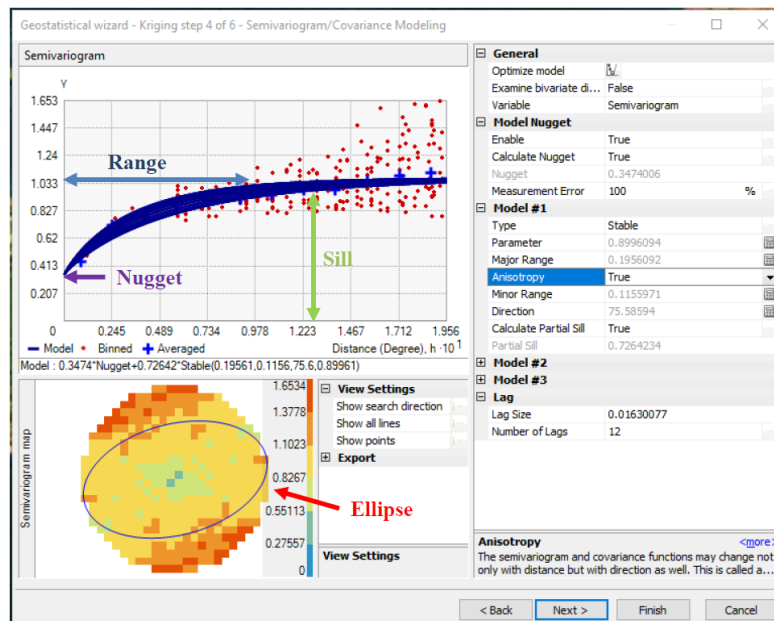


Figure 34: Isotropic semivariogram showing three characteristics: sill, range, and nugget

Anisotropy is a property of a spatial process in which spatial dependence (autocorrelation) changes with both the distance and the direction between two locations. For anisotropy, the shape of the semivariogram may vary with direction. Alternatively, isotropy exists when the semivariogram does not vary according to direction. Esri provides a clear illustration of the difference between these methods in Figure 35.

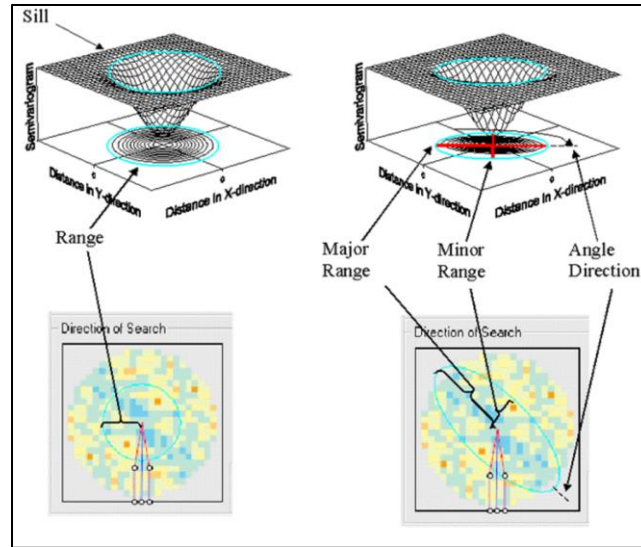


Figure 35: Esri [illustration](#) showing the differences in the area the models reach for an isotropic (left) and anisotropic (right) semivariogram

Rather than considering points in a sphere, as conducted by the isotropic semivariogram, an anisotropic semivariogram considers points within an ellipse with a major ridge, minor ridge, and angle direction. As seen in previous well production analysis, there are clearly defined regions of high and low production in both counties. By placing the ellipse so that areas with similar production results are placed along the major range, the semivariogram predicted production results will be more accurate. The predicted results for Susquehanna and Washington counties are shown in Figure 36.

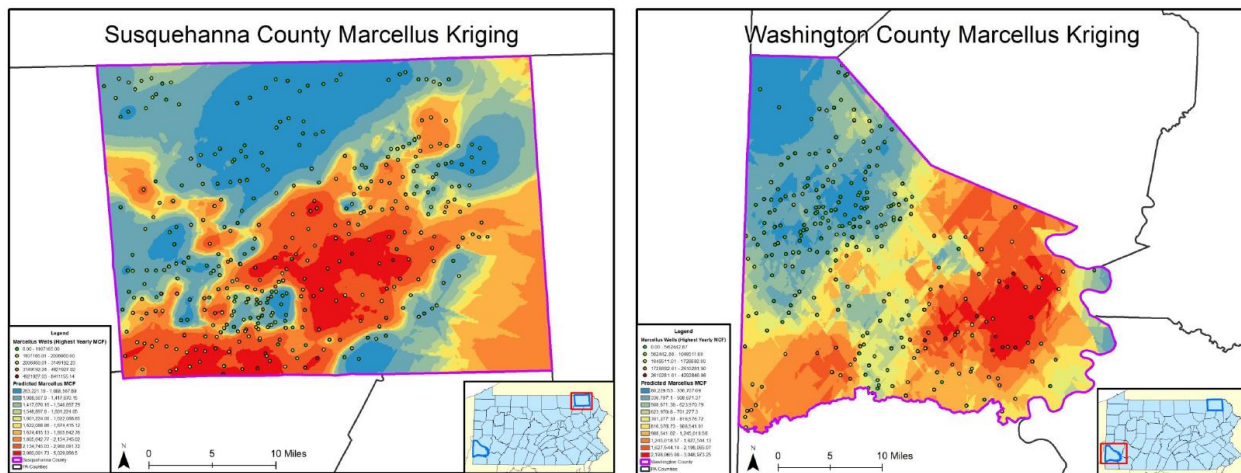


Figure 36: Predicted Marcellus production results for Susquehanna and Washington counties using an anisotropic semivariogram

3.7 Process 3: Develop Tool based on Findings

In process 3, a tool was developed using Python based on the findings in the first two steps. As discussed in previous sections, gas companies need to remain focused on returns on investment, rather than production growth, as the most significant metric for success in the exploration-and-production industry - this tool will help support this focus.

3.7.1 Tool Mock-up

The tool was designed by identifying system requirements in the needs assessment phase in July 2017 at CNX Resources Corp. The prototype was initially developed using [Balsamiq](#) (Figure 37).

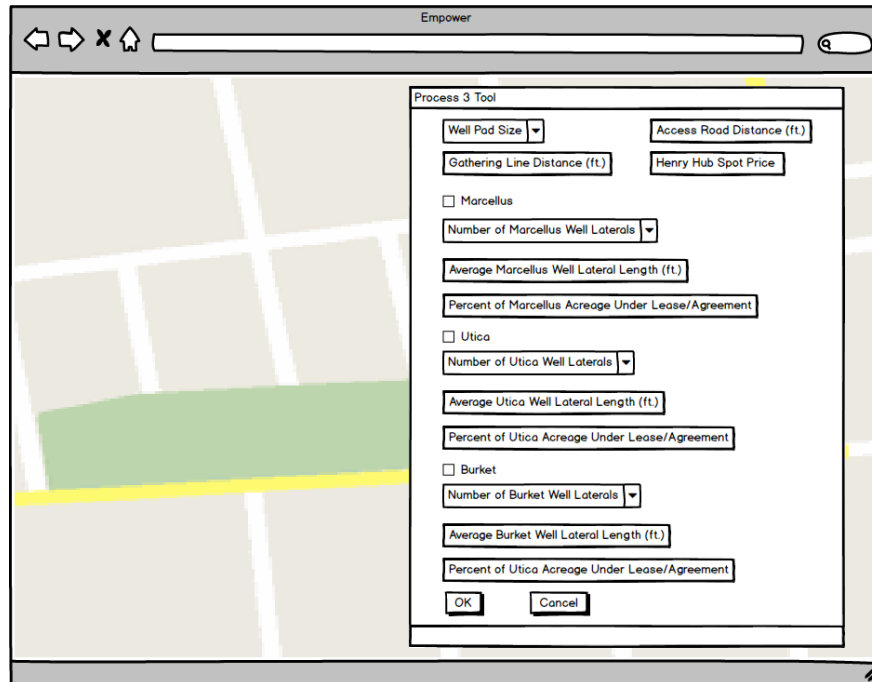


Figure 37: Wireframe of tool that was be developed in process 3

For this process, data within Susquehanna and Washington will again be the study area. A random 5% of producing wells will be sampled using the [subset features](#) tool in ArcMap. Drilling units for the sampled well will be digitized from courthouse records; both the sampled wells and drilling units will be treated as potential projects.

3.7.2 Slope Analysis at Wellpad

There is an added cost associated with placing a well pad in a location with steep topography, which is common in Pennsylvania given the state's terrain and will be accounted for in the developed tool. This added cost will be generated by performing a [Slope](#) analysis using the Digital Elevation Model from the 2006 - 2008 - DCNR PAMAP raster dataset. This raster was selected for this analysis as it was produced before most unconventional gas production occurred in the state.

3.7.3 Tool User Interface

The parameters, explanation, and corresponding data types that the user will need to input into the tool can be found in Figure 38. The parameters with a Boolean data type are checkboxes in the user interface.

Parameter	Explanation	Data Type
Workspace	Select a workspace location.	Workspace
Unit	Unit ID where the analysis will be performed.	String
Percent_Unit_Ownership	Percent ownership the exploration company has in the drilling unit. Allowed values from 0 - 100. 100 is the default value.	String
New_Wellpad_Location (Optional)	Is this a new wellpad location? If wellpad is already constructed the box can be left unchecked.	Boolean
Pad_Size (Optional)	User selects from Regular or Super Wellpad.	String
Access_Road_Length (Optional)	Length of access road in feet that will be constructed.	String
Gathering_Line_Distance (Optional)	Length of midstream gathering line in feet that will be constructed.	String
Henry_Hub_Price	Current Henry Hub spot price.	String
Marcellus (Optional)	Check if the Marcellus formation will be included in the analysis.	Boolean
Marcellus_Lateral_Count (Optional)	The number of Marcellus laterals that are proposed and input into the analysis.	String
Marcellus_Average_Lateral_Length (Optional)	Average Marcellus lateral length in ft.	String
Percent_Controlled_Marcellus_Leasehold (Optional)	Percent of Marcellus leasehold currently under company control. Allowed values from 0 - 100.	String
Utica (Optional)	Check if the Utica formation will be included in the analysis.	Boolean
Utica_Lateral_Count (Optional)	The number of Utica laterals that are proposed and input into the analysis.	String
Average_Utica_Lateral_Length (Optional)	Average Utica lateral length in ft.	String
Percent_Controlled_Utica_Leasehold (Optional)	Percent of Utica leasehold currently under company control. Allowed values from 0 - 100.	String
Burket (Optional)	Check if the Utica formation will be included in the analysis.	Boolean
Burket_Lateral_Count (Optional)	The number of Burket laterals that are proposed and input into the analysis.	String
Average_Burket_Lateral_Length (Optional)	Average Burket lateral length in ft.	String
Percent_Controlled_Burket_Leasehold (Optional)	Percent of Burket leasehold currently under company control. Allowed values from 0 - 100.	String

Figure 38: Table listing the input parameters, explanation, and corresponding data types for tool developed in Process 3

Customized validation (found in [Appendix A, Script 4: Source code for Process 3 validation in Python](#)) was utilized in this tool making the interface easier for the user input the parameters. For example, inputs relating to the Marcellus formation were only made accessible if the Marcellus checkbox was checked. If the Marcellus will not be explored at a given wellpad, the box will remain unchecked and the user will not input unnecessary values. The source code for this tool can be found in [Appendix A, Script 3: Source code for Process 3 in Python](#). A video showing the tool’s interface can be viewed [here](#).

3.7.4 Tool Workflow

The tool requires a Unit feature class or Wellpad feature class as inputs for the tool. The unit feature class will represent the area of the proposed drilling unit and the wellpad feature class is the proposed pad center. Both feature classes only need a “Unit_ID” field with matching attributes for the tool to successfully run. Error handling was incorporated into the Python script so that if wellpad fields were not created (if the user is running the tool for the first time) the fields would be added and attributed. If the fields were previously created, the existing fields would be attributed without duplicating fields. This will allow for easy comparison of units and wellpad locations, allowing the user to determine the most efficient and productive location with the least environmental impact. The output fields, a description as well as a column describing when the field will be populated can be found in Figure 39.

Wellpad Fields	Description	Field Populates
UNIT_ID	ID that correlates with feature in Units feature class.	N/A
Wellpad_Dev_Cost	The cost to build/develop well pad. Dependent on slope and pad size.	New Wellpad Location is True
Unit_Acres	Calculated acreage from the Units feature class.	Tool is run
Forest_Loss	Predicted forest in acres based on percent loss based on 2005 fragmentation class and wellpad location.	New Wellpad Location is True
Slope_Deg	Slope at wellpad location in degrees.	New Wellpad Location is True
Mar_Value	Company value of Marcellus gas extraction. Dependent on the Henry Hub price and assigned company priority (from .CSV file).	Marcellus Checkbox is True
Mar_Cost	Cost to drill Marcellus well(s) and lease outstanding Marcellus leasehold.	Marcellus Checkbox is True
Mar_Potential_MCF	Populated from Marcellus kriging analysis in Process 2.	Marcellus Checkbox is True
Uti_Value	Company value of Utica gas extraction. Dependent on the Henry Hub price and assigned company priority (from .CSV file)	Utica Checkbox is True
Uti_Cost	Cost to drill Utica well(s) and lease outstanding Marcellus leasehold.	Utica Checkbox is True
Uti_Potential_MCF	Populated from Utica Kriging analysis (assisted by Geology Dept.) in Process 2.	Utica Checkbox is True
Bur_Value	Company value of Burket gas extraction. Dependent on the Henry Hub price and assigned company priority (from .CSV file)	Burket Checkbox is True
Bur_Cost	Cost to drill Burket well(s) and lease outstanding Burket leasehold.	Burket Checkbox is True
Bur_Potential_MCF	Populated from Burket kriging analysis (assisted by Geology Dept.) in Process 2.	Burket Checkbox is True
Total_Cost	Total cost to lease minerals and extract gas by selected formations	Marcellus or Utica or Burket Checkbox is True
Total_Potential_MCF	Total potential gas MCF by selected formations.	Marcellus or Utica or Burket Checkbox is True

Figure 39: Tool output fields, description and when the field will be populated

Figure 40 provides a schematic of all that will be incorporated as inputs into the tool. Many input attributes will be stored within a .CSV file, which will be located on the company’s network; the next section will further describe the rationality for this decision.

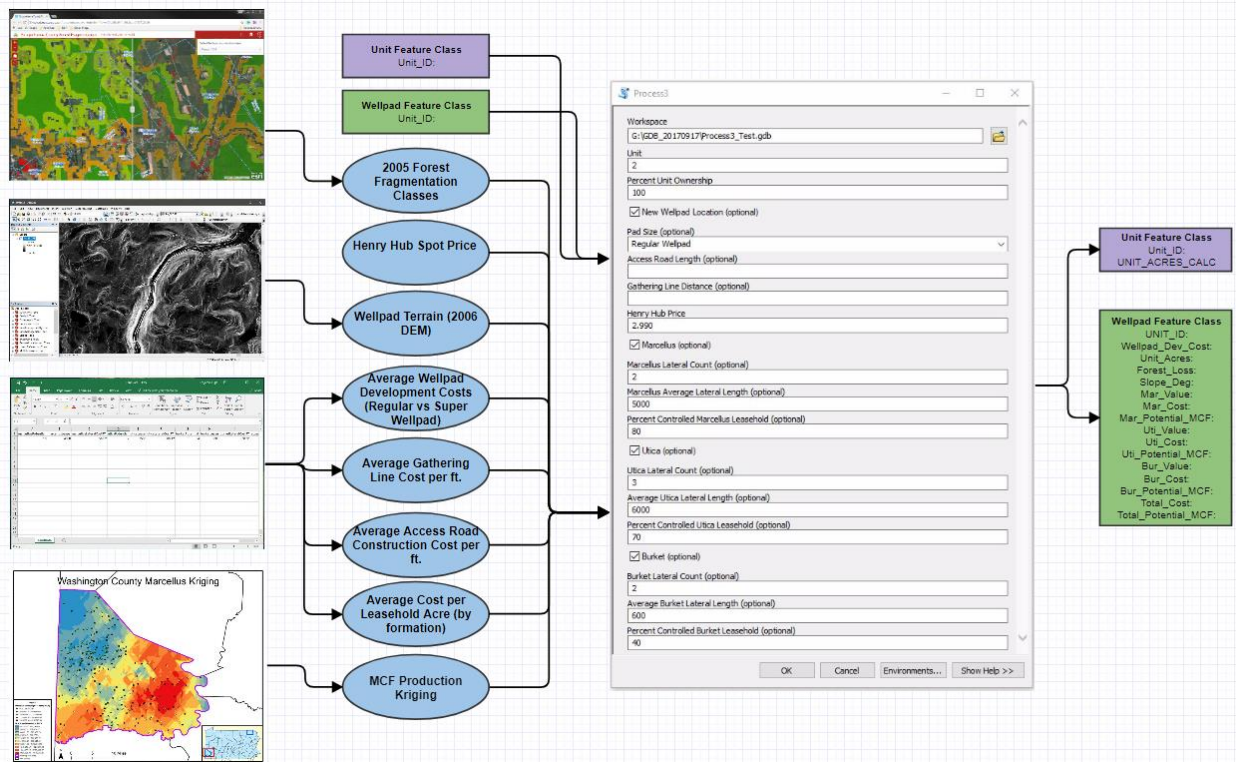


Figure 40: Schematic of all that will be inputs and output results of developed tool

3.7.5 CSV Module

This script utilized the CSV Python module to store some of the input values (Figure 41). These input values oftentimes change based on current economics or company priority. For instance, CNX Resources Corp. is currently placing a higher priority on extraction from the Marcellus formation, but this could change in the near future. Additionally, the bonus paid upon the signing of the lease can fluctuate based on economic conditions, third party competition, or the desire to drill in a particular geographic location. By placing these values in a .CSV file, the user can effortlessly adjust these values based on current economics without having to adjust coded values in the python script.

```

1. # Process the header
2. marcellusPotentialIndex = header.index("marcellusPotential")
3. marcellusLeaseIndex = header.index("marcellusLease")
4. marcellusLateralCostFTIndex = header.index("marcellusLateralCostFT")
5. uticaPotentialIndex = header.index("uticaPotential")
6. uticaLeaseIndex = header.index("uticaLease")
7. uticaLateralCostFTIndex = header.index("uticaLateralCostFT")
8. burketPotentialIndex = header.index("burketPotential")
9. burketLeaseIndex = header.index("burketLease")

```

```
10. burketLateralCostFTIndex = header.index("burketLateralCostFT")
11. accessRoadCostFTIndex = header.index("accessRoadCostFT")
12. gatheringCostFTIndex = header.index("gatheringCostFT")
13.
14. # Loop through the lines in the csv file and get each field
15. for row in csvReader:
16.     marcellusPotential_str = row[marcellusPotentialIndex]
17.     marcellusPotential = float(marcellusPotential_str)
18.     marcellusLease_str = row[marcellusLeaseIndex]
19.     marcellusLeaseCostAC = float(marcellusLease_str)
20.     marcellusLateralCostFT_str = row[marcellusLateralCostFTIndex]
21.     marcellusLateralCostFT = float(marcellusLateralCostFT_str)
22.     uticaPotential_str = row[uticaPotentialIndex]
23.     uticaPotential = float(uticaPotential_str)
24.     uticaLease_str = row[uticaLeaseIndex]
25.     uticaLeaseCostAC = float(uticaLease_str)
26.     uticalateralCostFT_str = row[uticalateralCostFTIndex]
27.     uticalateralCostFT = float(uticalateralCostFT_str)
28.     burketPotential_str = row[burketPotentialIndex]
29.     burketPotential = float(burketPotential_str)
30.     burketLease_str = row[burketLeaseIndex]
31.     burketLeaseCostAC = float(burketLease_str)
32.     burketLateralCostFT_str = row[burketLateralCostFTIndex]
33.     burketLateralCostFT = float(burketLateralCostFT_str)
34.     accessRoadCostFT_str = row[accessRoadCostFTIndex]
35.     accessRoadCostFT = float(accessRoadCostFT_str)
36.     gatheringCostFT_str = row[gatheringCostFTIndex]
37.     gatheringCostFT = float(gatheringCostFT_str)
```

Figure 41: Python code allowing the script to read values within the .CSV file

3.7.6 Tool Example

The following is an example of how the tool can be utilized at CNX Resources Corp. As seen in Figure 42, Company A (a third-party gas exploration company) already drilled and is producing the Marcellus formation from an existing and developed wellpad (Unit_ID = 7). Company A also currently leased 20% of the mineral right in the Utica formation in the area below their held Marcellus drilling unit. Company B (CNX Resources Corp.) leased the remaining 80% of mineral interest in the Utica formation in this area. Company B proposed a wellpad location (Unit_ID = 71). In this instance, CNX will be required to acquire the remaining 20% of Utica leasehold from Company A to drill the well and extract the Utica shale gas. There would also be included expense to develop a wellpad location, access road, and gathering pipelines to bring the gas to market. For this instance, the tool would be set by the user as shown below to estimate the costs to CNX, predicted MCF production, and potential fragmentation that could occur.

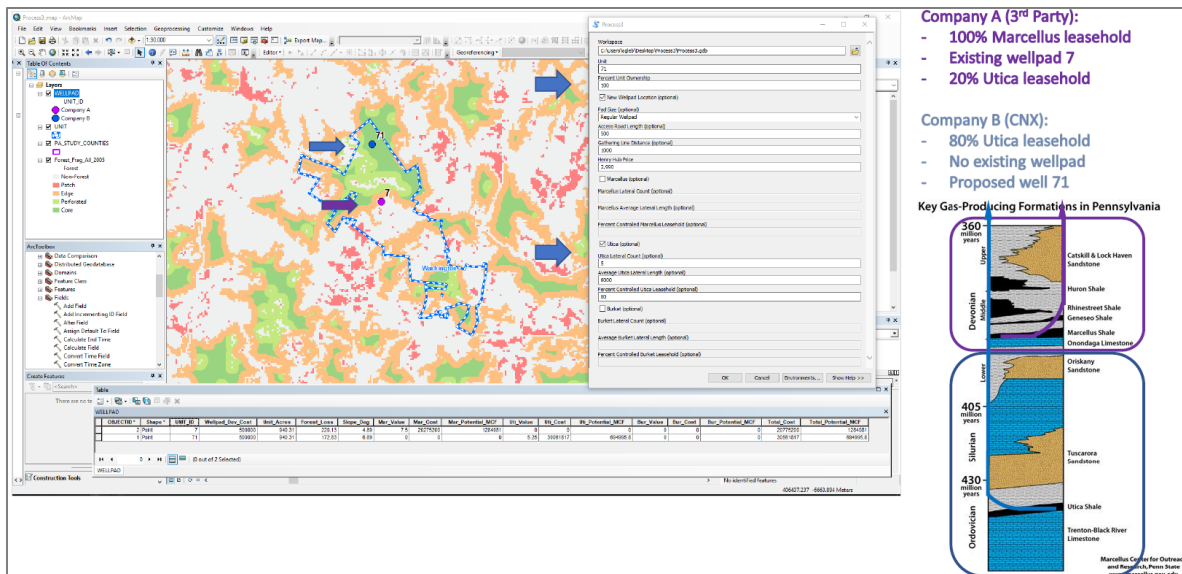


Figure 42: First instance of tool results showing how Utica shale gas can be extracted by developing an additional well pad

In the second instance (Figure 43), displays how forming a joint venture and stacking the shale plays at this location can benefit both companies and result in less overall forest fragmentation. In this instance, Company B (CNX) would utilize the existing wellpad (Unit_ID = 7) by forming a joint venture agreement with Company A. CNX would receive 589,085 less yearly predicted MCF production from the Utica formation as they are only an 80% owner in this instance. But, CNX would save an estimated 7 million dollars by not needing to develop a new wellpad, access road, and gathering pipelines. Additionally, an estimated 172.83 forested acres will not be fragmented in this case.

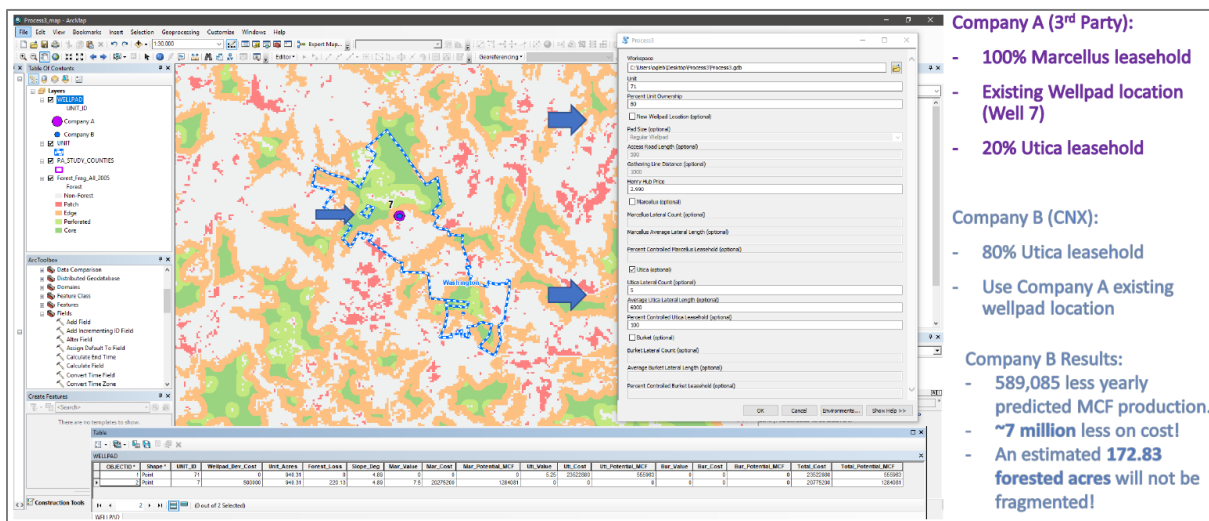


Figure 43: Second instance of tool results showing how Utica shale gas can be extracted by utilizing a joint venture

If gas exploration companies work together, by forming joint owner agreements or trading leasehold so that only one company has full ownership at all depths, a drilling unit will be more efficient and result in less overall forest fragmentation. This tool will be helpful in making that determination to pursue a joint venture agreement. Furthermore, this tool shows the value in stacking shale plays to increase production without causing additional surface disruptions.

3.8 Sharing Developed Tools and Datasets

Datasets and results were shared on CNX's ArcGIS Online organizational account. Feature classes were created in the enterprise SDE to store these datasets. Map services were created on ArcGIS Server and shared with the company. A Web mapping application was developed to display and share results using Esri Web AppBuilder. The tool developed in Process 3 will be shared with the organization as a geoprocessing REST Service.

IV. Results & Conclusions

Based on the previous analysis and developed tools, the following conclusions can be made. First, it can be determined, based on our finding in process 1, that most forest fragmentation that occurred during the researched time frame resulted from oil & gas activities. It is important to note, there were areas within our study area where core forest loss was occurring that also had no oil & gas activity. Second, it can be concluded that geographic areas within the shale basin where wells can be developed and produce from multiple shale formations from the same pad will be overall more productive and result in less forest fragmentation. As seen using the tool developed in process 3, it is advantageous for gas exploration companies to prioritize and develop in these areas. If gas exploration companies work together by forming joint owner agreements or trading leasehold so that only one company has full ownership at all depths a drilling unit will be more efficient and result in less forest fragmentation from "industrial linear corridors". Finally, regions, where drilling units were once economically viable, will be less attractive today because of the lower natural gas price. For example, Susquehanna County can only extract shale gas from the Marcellus formation. As for Washington County, gas can be potentially extracted from three shale layers. As exploration companies remain focused on returns on investment, rather than production growth, as the most significant metric for success in the exploration-and-production industry, the conclusions on production rates and forest fragmentation and the tools developed in this capstone project will contribute to this effort.

4.1 Project Timeline

This project followed the timeline in Figure 44. The capstone project was presented at the 12th Annual NW PA GIS Conference at Clarion University of Pennsylvania. The presentation occurred in the Gemmell Student Complex, Room 248 at 10:15 AM on October 19, 2017. The presentation was approximately forty minutes in length with five minutes for questions.

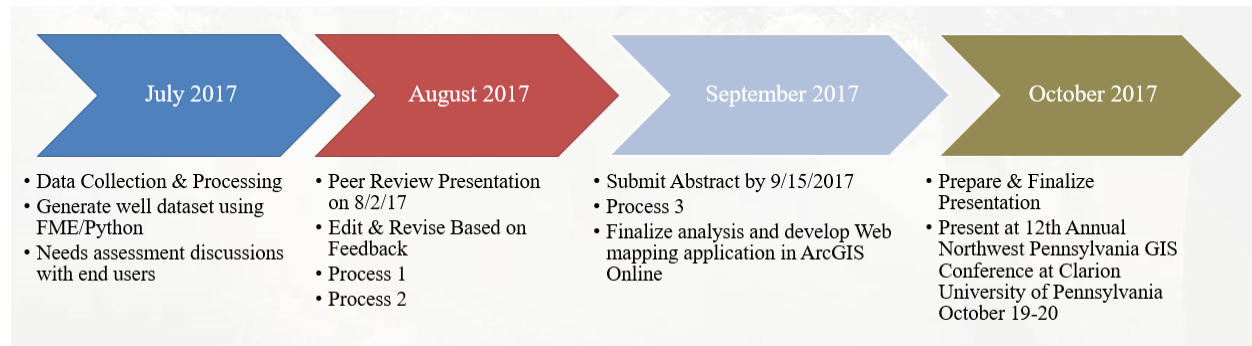


Figure 44: Capstone project timeline

4.2 Challenges

There were challenges that needed to be overcome throughout the course of this project. First, data is often difficult to source. Gas companies do not want their data, which is not required to become public record, to be made readily available as it will give their competitors an advantage. This includes pipeline data and company access roads to well pads. Another challenge Landscape Fragmentation Tool (LFT) v 2.0 did not run properly; as discussed, a model was developed to perform this analysis.

References

- ArcGIS Desktop Help 9.3 - Anisotropy: Directional semivariogram and covariance functions. (n.d.). Retrieved February 06, 2017, from http://webhelp.esri.com/ARCGISDESKTOP/9.3/index.cfm?TopicName=Anisotropy%3A_Directional_s emivariograms_and_covariance_functions
- Abrahams, L. S., Griffin, W. M., & Matthews, H. S. (2015). Assessment of policies to reduce core forest fragmentation from Marcellus shale development in Pennsylvania. *Ecological Indicators*, 52, 153-160. doi:10.1016/j.ecolind.2014.11.031
- Barlow, K. M., Mortensen, D. A., Drohan, P. J., & Averill, K. M. (2017). Unconventional gas development facilitates plant invasions. *Journal of Environmental Management*, 202, 208-216. doi:10.1016/j.jenvman.2017.07.005
- Bonhling, G. (2005, October 17). Introduction to Geostatistics and Variogram Analysis. Retrieved February 6, 2017, from <http://people.ku.edu/~gbohling/cpe940/Variograms.pdf>
- Drohan, P. J., Brittingham, M., Bishop, J., & Yoder, K. (2012). Early Trends in Landcover Change and Forest Fragmentation Due to Shale-Gas Development in Pennsylvania: A Potential Outcome for the Northcentral Appalachians. *Environmental Management*, 49(5), 1061-1075. doi:10.1007/s00267-012-9841-6

- Hohn, M., Pool, S. & Moore, J. (2015). Utica Play Resource Assessment, A Geologic Play Book for Utica Shale Appalachian Basin Exploration, Final report of the Utica Shale Appalachian basin exploration consortium, p. 159-183, from <http://www.wvgs.wvnet.edu/utica/playbook/docs/A-9.pdf>
- Johnson, N. (2010). Pennsylvania Energy Impacts Assessment Report 1: Marcellus Shale Natural Gas and Wind. Retrieved June 1, 2017, from https://www.nature.org/media/pa/tnc_energy_analysis.pdf .
- Kiviat, E. (2013). Risks to biodiversity from hydraulic fracturing for natural gas in the Marcellus and Utica shales. *Annals of the New York Academy of Sciences*, 1286(1), 1-14. doi:10.1111/nyas.12146
- Kramer, B. M., & Martin, P. H. (2006). *The Law of Pooling and Unitization* (3rd ed.).
- Lampe, D. J., & Stolz, J. F. (2015). Current perspectives on unconventional shale gas extraction in the Appalachian Basin. *Journal of Environmental Science and Health, Part A*, 50(5), 434-446. doi:10.1080/10934529.2015.992653
- Langlois, L. A., Drohan, P. J., & Brittingham, M. C. (2017). Linear infrastructure drives habitat conversion and forest fragmentation associated with Marcellus shale gas development in a forested landscape. *Journal of Environmental Management*, 197, 167-176. doi:10.1016/j.jenvman.2017.03.045
- Manda, A. K., Heath, J. L., Klein, W. A., Griffin, M. T., & Montz, B. E. (2014). Evolution of multi-well pad development and influence of well pads on environmental violations and wastewater volumes in the Marcellus Shale (USA). *Journal of Environmental Management*, 142, 36-45. doi:10.1016/j.jenvman.2014.04.011
- Nyahay, R et al. (2007). Update on Regional Assessment of Gas Potential in the Devonian Marcellus and Ordovician Utica Shales of New York. 10136th ser. Retrieved July 1, 2017, from <http://www.searchanddiscovery.com/documents/2007/07101nyahay/>
- O'Sullivan, D., & Unwin, D. (2014). *Geographic Information Analysis: Edition 2*. Hoboken: Wiley.
- PA DNR. (n.d.). The Marcellus Shale Play in Pennsylvania, Part 1: A Historical Overview. Retrieved June 2, 2017, from http://www.dcnr.state.pa.us/cs/groups/public/documents/document/dcnr_007596.pdf
- Parent, J. R., & Hurd, J. D. (n.d.). *Landscape Fragmentation Tool (LFT v2.0)*. Retrieved June 01, 2017, from <http://clear.uconn.edu/tools/lft/lft2/index.htm>
- Penn State. (2008, January 21). Unconventional Natural Gas Reservoir In Pennsylvania Poised To Dramatically Increase US Production. *ScienceDaily*. Retrieved June 30, 2017, from www.sciencedaily.com/releases/2008/01/080117094524.htm
- Reed, J. R., & Dunbar, D. (2008). Using ArcGIS to estimate thermogenic gas generation volumes by Upper and Middle Devonian shales in the Appalachian Basin. Retrieved July 1, 2017, from https://papgrocks.org/reed_pdf.
- Soeder, D. (1988). Porosity and Permeability of Eastern Devonian Gas Shale. *SPE Formation Evaluation*, 3(01), 116-124. doi:10.2118/15213-pa

Title 49 - Transportation, US Department of Transportation Pipeline and Hazardous Materials Safety Administration (PHMSA). § Section 192.5 - Class locations (2010).

<https://www.gpo.gov/fdsys/granule/CFR-2010-title49-vol3/CFR-2010-title49-vol3-sec192-5>

Utica Shale Play, Geology review (Rep.). (2017, April). U.S. Energy Information Administration, from

https://www.eia.gov/maps/pdf/UticaShalePlayReport_April2017.pdf

Vogt, P., Riitters, K. H., Estreguil, C., Kozak, J., Wade, T. G., & Wickham, J. D. (2007). Mapping Spatial Patterns with Morphological Image Processing. *Landscape Ecology*, 22(2), 171-177.
doi:10.1007/s10980-006-9013-2

Wrightstone, G. (2015, 07). Little brother to the Utica and Marcellus. *E & P*, , 1. Retrieved from <http://ezaccess.libraries.psu.edu/login?url=http://search.proquest.com.ezaccess.libraries.psu.edu/docview/1696925598?accountid=13158>

Zagorski, W. A., Wrightstone, G. R., & Bowman, D. C. (2012). The Appalachian Basin Marcellus Gas Play: Its History of Development, Geologic Controls on Production, and Future Potential as a World-class Reservoir. *Shale Reservoirs: Giant Resources for the 21st Century*, AAPG Memoir 97, 172-200.
doi:10.1306/13321465M973491

Appendix A: Source Code

Appendix A, Script 1: Source code for LISA Analysis Areas in Python

```
1. # Define Function
2. def removeNull(field):
3.     with arcpy.da.UpdateCursor(units, [field]) as cursor:
4.         for row in cursor:
5.             if row[0] == None:
6.                 row[0] = 0
7.                 cursor.updateRow(row)
8.
9.
10. # Script Body
11. import arcpy, os
12.
13. arcpy.env.workspace = r'E:\GDB_20170913\Ogle_GEOG596.gdb\FOREST'
14.
15. arcpy.env.overwriteOutput = True
16.
17. units = "E:\GDB_20170913\Ogle_GEOG596.gdb\UNITS\SUSQUEHANNA_UNITS"
18.
19. patchQuery = '"gridcode" = 1'
20. edgeQuery = '"gridcode" = 2'
21. perfQuery = '"gridcode" = 3'
22. coreQuery = '"gridcode" = 4'
23.
24. # Loop through Frag feature classes
25. for fc in arcpy.ListFeatureClasses():
26.     # Describe feature class in loop
27.     desc = arcpy.Describe(fc)
```



```
28.     baseName = desc.baseName
29.
30.     # Set name for output feature classes
31.     outIntersectFc = baseName + '_intersect'
32.     outDissolveFc = baseName + '_Intersect_Dissolve'
33.
34.     # Intersect Forest Frag feature classes with Units
35.     arcpy.Intersect_analysis([fc, units], outIntersectFc, "ALL", "", "")
36.
37.     # Dissolve Intersect by fragmentation type and unit FID
38.     arcpy.Dissolve_management(outIntersectFc, outDissolveFc, "gridcode;FID_SUSQUEHANNA_
UNITS", "", "MULTI_PART", "DISSOLVE_LINES")
39.
40.     # Add Field and calculate acreage
41.     arcpy.AddField_management(outDissolveFc, "FRAG_ACRES", "DOUBLE", "", "")
42.     arcpy.CalculateField_management (outDissolveFc, "FRAG_ACRES", "!shape.area@acres!",
"PYTHON_9.3")
43.
44.     if str(outDissolveFc) == "Forest_Frag_2005_Intersect_Dissolve":
45.         # Make 2005 Feature Layer
46.         arcpy.MakeFeatureLayer_management(outDissolveFc, "frag2005lyr")
47.
48.         # 2005 Patch, create feature and make feature layer
49.         arcpy.SelectLayerByAttribute_management("frag2005lyr", "NEW_SELECTION", patchQu
ery)
50.         arcpy.CopyFeatures_management("frag2005lyr", "Patch_2005")
51.
52.         arcpy.MakeFeatureLayer_management(units, "units05Patchlyr")
53.         arcpy.MakeFeatureLayer_management("Patch_2005", "2005Patchlyr")
54.
55.         # Join Field
56.         arcpy.AddJoin_management("units05Patchlyr", "OBJECTID", "2005Patchlyr", "FID_SU
SQUEHANNA_UNITS", "")
57.
58.         # Calculate Fields
59.         arcpy.CalculateField_management("units05Patchlyr", "Patch_Acres_05", "!Patch_20
05.FRAG_ACRES!", "PYTHON_9.3", "")
60.
61.         #Do some cleanup
62.         arcpy.RemoveJoin_management("units05Patchlyr", "Patch_2005")
63.         arcpy.Delete_management("units05Patchlyr")
64.         arcpy.Delete_management("2005Patchlyr")
65.
66.
67.         # 2005 Edge, create feature and make feature layer
68.         arcpy.SelectLayerByAttribute_management("frag2005lyr", "NEW_SELECTION", edgeQue
ry)
69.         arcpy.CopyFeatures_management("frag2005lyr", "Edge_2005")
70.
71.         arcpy.MakeFeatureLayer_management(units, "units05Edgelyr")
72.         arcpy.MakeFeatureLayer_management("Edge_2005", "2005Edgelyr")
73.
74.         # Join Field
75.         arcpy.AddJoin_management("units05Edgelyr", "OBJECTID", "2005Edgelyr", "FID_SUSQ
UEHANNA_UNITS", "")
76.
77.         # Calculate Fields
78.         arcpy.CalculateField_management("units05Edgelyr", "Edge_Acres_05", "!Edge_2005.
FRAG_ACRES!", "PYTHON_9.3", "")
```

```
79.
80.     #Do some cleanup
81.     arcpy.RemoveJoin_management("units05Edgelyr", "Edge_2005")
82.     arcpy.Delete_management("units05Edgelyr")
83.     arcpy.Delete_management("2005Edgelyr")
84.
85.
86.     # 2005 Perf, create feature and make feature layer
87.     arcpy.SelectLayerByAttribute_management("frag2005lyr", "NEW_SELECTION", perfQue
ry)
88.     arcpy.CopyFeatures_management("frag2005lyr", "Perf_2005")
89.
90.     arcpy.MakeFeatureLayer_management(units, "units05Perflyr")
91.     arcpy.MakeFeatureLayer_management("Perf_2005", "2005Perflyr")
92.
93.     # Join Field
94.     arcpy.AddJoin_management("units05Perflyr", "OBJECTID", "2005Perflyr", "FID_SUSQ
UEHANNA_UNITS", "")
95.
96.     # Calculate Fields
97.     arcpy.CalculateField_management("units05Perflyr", "Perf_Acres_05", "!Perf_2005.
FRAG_ACRES!", "PYTHON_9.3", "")
98.
99.     #Do some cleanup
100.    arcpy.RemoveJoin_management("units05Perflyr", "Perf_2005")
101.    arcpy.Delete_management("Units05Perflyr")
102.    arcpy.Delete_management("2005Perflyr")
103.
104.
105.    # 2005 Core, create feature and make feature layer
106.    arcpy.SelectLayerByAttribute_management("frag2005lyr", "NEW_SELECTION",
coreQuery)
107.    arcpy.CopyFeatures_management("frag2005lyr", "Core_2005")
108.
109.    arcpy.MakeFeatureLayer_management(units, "units05Corelyr")
110.    arcpy.MakeFeatureLayer_management("Core_2005", "2005Corelyr")
111.
112.    # Join Field
113.    arcpy.AddJoin_management("units05Corelyr", "OBJECTID", "2005Corelyr", "F
ID_SUSQUEHANNA_UNITS", "")
114.
115.    # Calculate Fields
116.    arcpy.CalculateField_management("units05Corelyr", "Core_Acres_05", "!Cor
e_2005.FRAG_ACRES!", "PYTHON_9.3", "")
117.
118.    #Do some cleanup
119.    arcpy.RemoveJoin_management("units05Corelyr", "Core_2005")
120.    arcpy.Delete_management("units05Corelyr")
121.    arcpy.Delete_management("2005Corelyr")
122.
123.    arcpy.Delete_management("frag2005lyr")
124.
125.    elif str(outDissolveFc) == "Forest_Frag_2013_Intersect_Dissolve":
126.        # Make 2013 Feature Layer
127.        arcpy.MakeFeatureLayer_management(outDissolveFc, "frag2013lyr")
128.
129.        # 2013 Patch, create feature and make feature layer
130.        arcpy.SelectLayerByAttribute_management("frag2013lyr", "NEW_SELECTION",
patchQuery)
```

```
131.         arcpy.CopyFeatures_management("frag2013lyr", "Patch_2013")
132.
133.         arcpy.MakeFeatureLayer_management(units, "units13Patchlyr")
134.         arcpy.MakeFeatureLayer_management("Patch_2013", "2013Patchlyr")
135.
136.         # Join Field
137.         arcpy.AddJoin_management("units13Patchlyr", "OBJECTID", "2013Patchlyr",
138. "FID_SUSQUEHANNA_UNITS", "")
139.
140.         # Calculate Fields
141.         arcpy.CalculateField_management("units13Patchlyr", "Patch_Acres_13", "!P
142. atch_2013.FRAG_ACRES!", "PYTHON_9.3", "")
143.
144.         #Do some cleanup
145.         arcpy.RemoveJoin_management("units13Patchlyr", "Patch_2013")
146.         arcpy.Delete_management("units13Patchlyr")
147.         arcpy.Delete_management("2013Patchlyr")
148.
149.         # 2013 Edge, create feature and make feature layer
150.         arcpy.SelectLayerByAttribute_management("frag2013lyr", "NEW_SELECTION",
151. edgeQuery)
152.         arcpy.CopyFeatures_management("frag2013lyr", "Edge_2013")
153.
154.         arcpy.MakeFeatureLayer_management(units, "units13Edgelyr")
155.         arcpy.MakeFeatureLayer_management("Edge_2013", "2013Edgelyr")
156.
157.         # Join Field
158.         arcpy.AddJoin_management("units13Edgelyr", "OBJECTID", "2013Edgelyr", "F
159. ID_SUSQUEHANNA_UNITS", "")
160.
161.         # Calculate Fields
162.         arcpy.CalculateField_management("units13Edgelyr", "Edge_Acres_13", "!Edg
163. e_2013.FRAG_ACRES!", "PYTHON_9.3", "")
164.
165.         #Do some cleanup
166.         arcpy.RemoveJoin_management("units13Edgelyr", "Edge_2013")
167.         arcpy.Delete_management("units13Edgelyr")
168.         arcpy.Delete_management("2013Edgelyr")
169.
170.         # 2013 Perf, create feature and make feature layer
171.         arcpy.SelectLayerByAttribute_management("frag2013lyr", "NEW_SELECTION",
172. perfQuery)
173.         arcpy.CopyFeatures_management("frag2013lyr", "Perf_2013")
174.
175.         arcpy.MakeFeatureLayer_management(units, "units13Perflyr")
176.         arcpy.MakeFeatureLayer_management("Perf_2013", "2013Perflyr")
177.
178.         # Join Field
179.         arcpy.AddJoin_management("units13Perflyr", "OBJECTID", "2013Perflyr", "F
180. ID_SUSQUEHANNA_UNITS", "")
181.
182.         # Calculate Fields
183.         arcpy.CalculateField_management("units13Perflyr", "Perf_Acres_13", "!Per
184. f_2013.FRAG_ACRES!", "PYTHON_9.3", "")
185.
186.         #Do some cleanup
187.         arcpy.RemoveJoin_management("units13Perflyr", "Perf_2013")
```

```

182.         arcpy.Delete_management("Units13Perflyr")
183.         arcpy.Delete_management("2013Perflyr")
184.
185.
186.         # 2013 Core, create feature and make feature layer
187.         arcpy.SelectLayerByAttribute_management("frag2013lyr", "NEW_SELECTION",
coreQuery)
188.         arcpy.CopyFeatures_management("frag2013lyr", "Core_2013")
189.
190.         arcpy.MakeFeatureLayer_management(units, "units13Corelyr")
191.         arcpy.MakeFeatureLayer_management("Core_2013", "2013Corelyr")
192.
193.         # Join Field
194.         arcpy.AddJoin_management("units13Corelyr", "OBJECTID", "2013Corelyr", "F
ID_SUSQUEHANNA_UNITS", "")
195.
196.         # Calculate Fields
197.         arcpy.CalculateField_management("units13Corelyr", "Core_Acres_13", "!Cor
e_2013.FRAG_ACRES!", "PYTHON_9.3", "")
198.
199.         #Do some cleanup
200.         arcpy.RemoveJoin_management("units13Corelyr", "Core_2013")
201.         arcpy.Delete_management("units13Corelyr")
202.         arcpy.Delete_management("2013Corelyr")
203.
204.         arcpy.Delete_management("frag2013lyr")
205.
206.
207.         # Remove Null Values
208.         removeNull("Patch_Acres_05")
209.         removeNull("Edge_Acres_05")
210.         removeNull("Perf_Acres_05")
211.         removeNull("Core_Acres_05")
212.         removeNull("Patch_Acres_13")
213.         removeNull("Edge_Acres_13")
214.         removeNull("Perf_Acres_13")
215.         removeNull("Core_Acres_13")
216.
217.
218.         # Calculate Percent
219.         arcpy.CalculateField_management(units, "Patch_Per_05", "(!Patch_Acres_05!/UNIT_
ACRES!) * 100", "PYTHON_9.3", "")
220.         arcpy.CalculateField_management(units, "Edge_Per_05", "(!Edge_Acres_05!/UNIT_AC
RES!) * 100", "PYTHON_9.3", "")
221.         arcpy.CalculateField_management(units, "Perf_Per_05", "(!Perf_Acres_05!/UNIT_AC
RES!) * 100", "PYTHON_9.3", "")
222.         arcpy.CalculateField_management(units, "Core_Per_05", "(!Core_Acres_05!/UNIT_AC
RES!) * 100", "PYTHON_9.3", "")
223.
224.         arcpy.CalculateField_management(units, "Patch_Per_13", "(!Patch_Acres_13!/UNIT_
ACRES!) * 100", "PYTHON_9.3", "")
225.         arcpy.CalculateField_management(units, "Edge_Per_13", "(!Edge_Acres_13!/UNIT_AC
RES!) * 100", "PYTHON_9.3", "")
226.         arcpy.CalculateField_management(units, "Perf_Per_13", "(!Perf_Acres_13!/UNIT_AC
RES!) * 100", "PYTHON_9.3", "")
227.         arcpy.CalculateField_management(units, "Core_Per_13", "(!Core_Acres_13!/UNIT_AC
RES!) * 100", "PYTHON_9.3", "")
228.
229.

```

```

230.     # Calculate Forest Change
231.     arcpy.CalculateField_management(units, "Patch_Change", "!Patch_Per_13!-
!Patch_Per_05!", "PYTHON_9.3", "")
232.     arcpy.CalculateField_management(units, "Edge_Change", "!Edge_Per_13!-
!Edge_Per_05!", "PYTHON_9.3", "")
233.     arcpy.CalculateField_management(units, "Perf_Change", "!Perf_Per_13!-
!Perf_Per_05!", "PYTHON_9.3", "")
234.     arcpy.CalculateField_management(units, "Core_Change", "!Core_Per_13!-
!Core_Per_05!", "PYTHON_9.3", "")
235.     arcpy.CalculateField_management(units, "Forest_Change", "(!Patch_Per_13! + !Edge
_Per_13! + !Perf_Per_13! + !Core_Per_13!)-
(!Patch_Per_05! + !Edge_Per_05! + !Perf_Per_05! + !Core_Per_05!)", "PYTHON_9.3", "")

```

Appendix A, Script 2: Source Code for density analysis and Monte Carlo simulation in R

```

1. library(spatstat)
2. library(maptools)
3.
4. # import Susquehanna well and county for analysis
5. setwd("F:/R_Shapefiles")
6. Sus_Wells_Center = readShapePoints("Susquehanna_Wellspad_Center.shp")
7. SP_Sus_Wells_Center = as(Sus_Wells_Center, "SpatialPoints")
8. Sus_County = readShapePoly("Susquehanna_County.shp")
9. P_Sus_Wells_Center = as(SP_Sus_Wells_Center, "ppp")
10.
11. # plot well dataset
12. par(mfrow=c(1, 2))
13. plot(Sus_County)
14. Sus_Wells_Center_d = density(P_Sus_Wells_Center)
15. plot(Sus_Wells_Center_d, add=T)
16. contour(Sus_Wells_Center_d, add=T)
17. plot(P_Sus_Wells_Center, add=T)
18.
19. # perform the Gest function with a 99 simulations; plot Susquehanna wells results
20. Sus_Wells_Center_G_env = envelope(P_Sus_Wells_Center, Gest, nsim = 99, nrank = 1)
21. plot(Sus_Wells_Center_G_env)

```

Appendix A, Script 3: Source code for Process 3 in Python

```

1. # Import arcpy, string, time, csv, and os
2. import arcpy, string, time, os, csv
3. from arcpy import env
4. from arcpy.sa import *
5.
6. arcpy.env.overwriteOutput = True
7.
8. # Check out the ArcGIS Spatial Analyst extension license
9. arcpy.CheckOutExtension("Spatial")
10.
11. # Set up CSV reader
12. landData = open("C:\\Users\\ogleb\\Desktop\\Process3\\Data\\LandData.csv", "r")
13. csvReader = csv.reader(landData)
14. header = csvReader.next() #csvReader.__next__()

```

```
15.
16. # Process the header
17. marcellusPotentialIndex = header.index("marcellusPotential")
18. marcellusLeaseIndex = header.index("marcellusLease")
19. marcellusLateralCostFTIndex = header.index("marcellusLateralCostFT")
20. uticaPotentialIndex = header.index("uticaPotential")
21. uticaLeaseIndex = header.index("uticaLease")
22. uticaLateralCostFTIndex = header.index("uticaLateralCostFT")
23. burketPotentialIndex = header.index("burketPotential")
24. burketLeaseIndex = header.index("burketLease")
25. burketLateralCostFTIndex = header.index("burketLateralCostFT")
26. accessRoadCostFTIndex = header.index("accessRoadCostFT")
27. gatheringCostFTIndex = header.index("gatheringCostFT")
28.
29. # Loop through the lines in the csv file and get each field
30. for row in csvReader:
31.     marcellusPotential_str = row[marcellusPotentialIndex]
32.     marcellusPotential = float(marcellusPotential_str)
33.     marcellusLease_str = row[marcellusLeaseIndex]
34.     marcellusLeaseCostAC = float(marcellusLease_str)
35.     marcellusLateralCostFT_str = row[marcellusLateralCostFTIndex]
36.     marcellusLateralCostFT = float(marcellusLateralCostFT_str)
37.     uticaPotential_str = row[uticaPotentialIndex]
38.     uticaPotential = float(uticaPotential_str)
39.     uticaLease_str = row[uticaLeaseIndex]
40.     uticaLeaseCostAC = float(uticaLease_str)
41.     uticaLateralCostFT_str = row[uticaLateralCostFTIndex]
42.     uticaLateralCostFT = float(uticaLateralCostFT_str)
43.     burketPotential_str = row[burketPotentialIndex]
44.     burketPotential = float(burketPotential_str)
45.     burketLease_str = row[burketLeaseIndex]
46.     burketLeaseCostAC = float(burketLease_str)
47.     burketLateralCostFT_str = row[burketLateralCostFTIndex]
48.     burketLateralCostFT = float(burketLateralCostFT_str)
49.     accessRoadCostFT_str = row[accessRoadCostFTIndex]
50.     accessRoadCostFT = float(accessRoadCostFT_str)
51.     gatheringCostFT_str = row[gatheringCostFTIndex]
52.     gatheringCostFT = float(gatheringCostFT_str)
53.
54. # Set up workspace
55. workspace = arcpy.GetParameterAsText(0)
56. arcpy.env.workspace = workspace
57. datasets = 'C:\\Users\\ogleb\\Desktop\\Process3\\Data\\Process3_Data.gdb\\'
58.
59. # Input datasets
60. units = workspace + "\\\" + 'UNIT'
61. wells = workspace + "\\\" + 'WELLPAD'
62. slope = datasets + 'PA_SLOPE'
63. forest_2005 = datasets + 'Forest_Frag_All_2005'
64. marcellus_kriging = datasets + 'WASH_SUS_MARCELLUS_KRIGING'
65. utica_kriging = datasets + 'WASH_SUS_UTICA_KRIGING'
66. burket_kriging = datasets + 'WASH_SUS_BURKET_KRIGING'
67.
68. # User inputs (well pad location)
69. unitQuery = arcpy.GetParameterAsText(1)
70. percentOwnership_str = arcpy.GetParameterAsText(2)
71. existingWellpad = arcpy.GetParameterAsText(3)
72. padSize = arcpy.GetParameterAsText(4)
73. accessRoadLength_str = arcpy.GetParameterAsText(5)
```

```
74. gatheringLineDistance_str = arcpy.GetParameterAsText(6)
75. hubPrice_str = arcpy.GetParameterAsText(7)
76.
77. # Calculations based on user inputs
78. percentOwnership = float(percentOwnership_str) / 100
79. hubPrice = float(hubPrice_str)
80.
81. # Allow user to not include values for access road length (if existing wellpad)
82. if accessRoadLength_str != "#":
83.     accessRoadLength = 0
84. else:
85.     accessRoadLength = int(accessRoadLength_str)
86.
87. # Allow user to not include values for gathering line distance (if existing wellpad)
88. if gatheringLineDistance_str != "#":
89.     gatheringLineDistance = 0
90. else:
91.     gatheringLineDistance = int(gatheringLineDistance_str)
92.
93. # User inputs (Marcellus)
94. marcellusTrue = arcpy.GetParameterAsText(8)
95. marcellusLateralCount_str = arcpy.GetParameterAsText(9)
96. marcellusLateralLength_str = arcpy.GetParameterAsText(10)
97. marcellusLeasehold_str = arcpy.GetParameterAsText(11)
98.
99. # User inputs (Utica)
100. uticaTrue = arcpy.GetParameterAsText(12)
101. uticaLateralCount_str = arcpy.GetParameterAsText(13)
102. uticaLateralLength_str = arcpy.GetParameterAsText(14)
103. uticaLeasehold_str = arcpy.GetParameterAsText(15)
104.
105. # User inputs (Burket)
106. burketTrue = arcpy.GetParameterAsText(16)
107. burketLateralCount_str = arcpy.GetParameterAsText(17)
108. burketLateralLength_str = arcpy.GetParameterAsText(18)
109. burketLeasehold_str = arcpy.GetParameterAsText(19)
110.
111. # Provide value based on pad size
112. if padSize == "Regular Wellpad":
113.     padValueRegular = 500000
114. elif padSize == "Super Wellpad":
115.     padValueSuper = 750000
116.
117. # Provide value based on Henry Hub Price
118. if hubPrice > 5:
119.     hubValue = 2
120. elif hubPrice >= 4:
121.     hubValue = 1.5
122. elif hubPrice >= 3:
123.     hubValue = 1
124. elif hubPrice >= 2:
125.     hubValue = 0.75
126. elif hubPrice >= 1:
127.     hubValue = 0.25
128. else:
129.     hubValue = 0
130.
131. # Make feature layer for user selected well and unit
```

```
132.     arcpy.MakeFeatureLayer_management(wells, "wellLyr", '"Unit_ID" = ' + unitQuery)
133.     arcpy.MakeFeatureLayer_management(units, "unitLyr", '"Unit_ID" = ' + unitQuery)
134.
135.     # Check if Wellpad_Dev_Cost field exist, create field if not
136.     fieldListUnit = arcpy.ListFields("unitLyr", "UNIT_ACRES_CALC")
137.     fieldCountUnit = len(fieldListUnit)
138.     if (fieldCountUnit == 1):
139.         print("UNIT_ACRES_TEMP field was already created")
140.         arcpy.CalculateField_management("unitLyr", "UNIT_ACRES_CALC", "round(!shape.
area@ACRES!, 2)", "PYTHON_9.3", "")
141.     else:
142.         arcpy.AddField_management("unitLyr", "UNIT_ACRES_CALC", "DOUBLE", "15", "2",
    "")
143.         arcpy.CalculateField_management("unitLyr", "UNIT_ACRES_CALC", "round(!shape.
area@ACRES!, 2)", "PYTHON_9.3", "")
144.
145.
146.     # Check if Wellpad_Dev_Cost field exist, create field if not
147.     fieldListCost = arcpy.ListFields("wellLyr", "Wellpad_Dev_Cost")
148.     fieldCountCost = len(fieldListCost)
149.     if (fieldCountCost == 1):
150.         print("Wellpad_Cost field was already created")
151.     else:
152.         arcpy.AddField_management("wellLyr", "Wellpad_Dev_Cost", "DOUBLE", "15", "2"
    , "")
153.
154.     # Perfrom an intersect of the vector datasets
155.     arcpy.Intersect_analysis(["wellLyr", units, forest_2005], "intersect_temp", "ALL
    ", "", "")
156.
157.     # Check if Unit_Acres field exist, create field if not
158.     fieldListUnitAc = arcpy.ListFields("wellLyr", "Unit_Acres")
159.     fieldCountUnitAc = len(fieldListUnitAc)
160.     if (fieldCountUnitAc == 1):
161.         print("Unit_Acres field was already created")
162.     else:
163.         arcpy.AddField_management("wellLyr", "Unit_Acres", "DOUBLE", "", "", "")
164.
165.     # Check if Forest_Loss field exist, create field if not
166.     fieldListForestLoss= arcpy.ListFields("wellLyr", "Forest_Loss")
167.     fieldCountForestLoss = len(fieldListForestLoss)
168.     if (fieldCountForestLoss == 1):
169.         print("ForestLoss field was already created")
170.     else:
171.         arcpy.AddField_management("wellLyr", "Forest_Loss", "DOUBLE", "", "", "")
172.
173.     # Check if Slope_Deg field exist, create field if not
174.     fieldListDeg = arcpy.ListFields("wellLyr", "Slope_Deg")
175.     fieldCountDeg = len(fieldListDeg)
176.     if (fieldCountDeg == 1):
177.
178.         # Execute ExtractValuesToPoints
179.         ExtractMultiValuesToPoints("wellLyr", [[slope, "Slope_Deg1"]], "BILINEAR")
180.
181.         # Create an update cursor to update the for Slope_Deg field
182.         with arcpy.da.UpdateCursor("wellLyr", ["Slope_Deg", "Slope_Deg1"]) as cursor
    :
```



```
183.         for row in cursor:
184.             row[0] = round(row[1], 2)
185.             cursor.updateRow(row)
186.         del row, cursor
187.
188.         # Execute DeleteField
189.         arcpy.DeleteField_management("wellyr", ["Slope_Deg1"])
190.
191.     else:
192.         # Execute ExtractValuesToPoints
193.         ExtractMultiValuesToPoints("wellyr", [[slope, "Slope_Deg"]], "BILINEAR")
194.
195.         # Add join from "wellyr" and the intersected datasets
196.         arcpy.AddJoin_management("wellyr", "OBJECTID", "intersect_temp", "FID_WELLPAD",
197.             "")
198.         # Calculate Forest_Loss and Unit_Acres Fields
199.         arcpy.CalculateField_management("wellyr", "Forest_Loss", "round(!intersect_temp
200.             .UNIT_ACRES_CALC! * !intersect_temp.Percent_Loss!, 2)", "PYTHON_9.3", "")
201.         arcpy.CalculateField_management("wellyr", "Unit_Acres", "!intersect_temp.UNIT_A
202.             CRES_CALC!", "PYTHON_9.3", "")
203.
204.         # Remove join and delete temporary feature class
205.         arcpy.RemoveJoin_management("wellyr", "intersect_temp")
206.         arcpy.Delete_management("intersect_temp")
207.
208.         # Provide message if a formation box was checked
209.         if (str(marcellusTrue) == 'true') or (str(uticaTrue) == 'true') or (str(burketTr
210.             ue) == 'true'):
211.             arcpy.AddMessage("A formation box was checked")
212.         else:
213.             arcpy.AddMessage("No formation box is checked")
214.
215.         # Create an update cursor to update the for the Wellpad_Dev_Cost field; account
216.             for pad size and terrain of location
217.         if str(padSize) == "Super Wellpad":
218.             # Update Wellpad_Dev_Cost for Super Wellpad
219.             with arcpy.da.UpdateCursor("wellyr", ["Slope_Deg", "Wellpad_Dev_Cost"]) as
220.                 cursor:
221.                     for row in cursor:
222.                         if row[0] >= 9:
223.                             row[1] = ((gatheringLineDistance * gatheringCostFT) + (accessRoa
224.                                 dLength * accessRoadCostFT) + padValueSuper + 25000) * percentOwnership
225.                         else:
226.                             row[1] = ((gatheringLineDistance * gatheringCostFT) + (accessRoa
227.                                 dLength * accessRoadCostFT) + padValueSuper) * percentOwnership
228.                             cursor.updateRow(row)
229.                         del row, cursor
230.
231.         else:
232.             # Update Wellpad_Dev_Cost for Regular Wellpad
233.             with arcpy.da.UpdateCursor("wellyr", ["Slope_Deg", "Wellpad_Dev_Cost"]) as
234.                 cursor:
235.                     for row in cursor:
236.                         if row[0] >= 9:
237.                             row[1] = round(((gatheringLineDistance * gatheringCostFT) + (acc
238.                                 essRoadLength * accessRoadCostFT) + padValueRegular + 25000) * percentOwnership, 2)
239.                         else:
```

```
231.             row[1] = round(((gatheringLineDistance * gatheringCostFT) + (acc
essRoadLength * accessRoadCostFT) + padValueRegular) * percentOwnership, 2)
232.             cursor.updateRow(row)
233.             del row, cursor
234.
235.
236.             # Marcellus check box
237.             if str(marcellusTrue) == 'true':
238.                 arcpy.AddMessage("The Marcellus box was checked")
239.
240.             # Do some Calculations
241.             marcellusLateralCount = int(marcellusLateralCount_str)
242.             marcellusLateralLength = int(marcellusLateralLength_str)
243.             unleasedMarcellusAC = 1 - (float(marcellusLeasehold_str) / 100)
244.
245.             # Check if Mar_Value field exist, create field if not
246.             fieldListMar = arcpy.ListFields("wellLyr", "Mar_Value")
247.             fieldCountMar = len(fieldListMar)
248.             if (fieldCountMar == 1):
249.                 print("Mar_Value field was already created")
250.             else:
251.                 arcpy.AddField_management("wellLyr", "Mar_Value", "DOUBLE", "", "", "")
252.
253.             # Create an update cursor to update the for Mar_Value field
254.             with arcpy.da.UpdateCursor("wellLyr", ["Mar_Value"]) as cursor:
255.                 for row in cursor:
256.                     row[0] = marcellusPotential * hubValue
257.                     cursor.updateRow(row)
258.                 del row, cursor
259.
260.             # Check if Mar_Cost field exist, create field if not
261.             fieldListMarCost = arcpy.ListFields("wellLyr", "Mar_Cost")
262.             fieldCountMarCost = len(fieldListMarCost)
263.             if (fieldCountMarCost == 1):
264.                 print("Mar_Value field was already created")
265.             else:
266.                 arcpy.AddField_management("wellLyr", "Mar_Cost", "DOUBLE", "", "", "")
267.
268.             # Create an update cursor to update the for Mar_Value field
269.             with arcpy.da.UpdateCursor("wellLyr", ["Unit_Acres", "Mar_Cost"]) as cursor:
270.                 for row in cursor:
271.                     row[1] = round(((marcellusLateralCostFT * marcellusLateralCount * ma
rcellusLateralLength) + (row[0] * unleasedMarcellusAC * marcellusLeaseCostAC)) * percen
tOwnership, 2)
272.                     cursor.updateRow(row)
273.                 del row, cursor
274.                 #arcpy.AddMessage("$" + str((marcellusLateralCostFT * marcellusLateralCount
* marcellusLateralLength) * percentOwnership) + " required to drill Marcellus laterals"
)
275.
276.             # Check if Mar_Potential_MCF field exist, create field if not
277.             fieldListMarMCF = arcpy.ListFields("wellLyr", "Mar_Potential_MCF")
278.             fieldCountMarMCF = len(fieldListMarMCF)
279.             if (fieldCountMarMCF == 1):
280.                 # Execute ExtractValuesToPoints
281.                 ExtractMultiValuesToPoints("wellLyr", [[marcellus_kriging, "Mar_Potentia
l_MCF_1"]], "BILINEAR")
```

```

282.
283.         # Create an update cursor to update the for Mar_Potential_MCF field
284.         with arcpy.da.UpdateCursor("wellLyr", ["Mar_Potential_MCF", "Mar_Potenti
al_MCF_1"]) as cursor:
285.             for row in cursor:
286.                 row[0] = percentOwnership * row[1]
287.                 cursor.updateRow(row)
288.             del row, cursor
289.
290.         # Execute DeleteField
291.         arcpy.DeleteField_management("wellLyr", ["Mar_Potential_MCF_1"])
292.
293.     else:
294.         # Execute ExtractValuesToPoints
295.         ExtractMultiValuesToPoints("wellLyr", [[marcellus_kriging, "Mar_Potentia
l_MCF"]], "BILINEAR")
296.
297.     else:
298.         # In this case, the check box value is 'false', user did not check the box
299.         arcpy.AddMessage("The Marcellus box was not checked")
300.
301.         # Check if Mar_Value field exist, create field if not. Calculate 0 for the
Mar_Value
302.         fieldListMarValueFalse = arcpy.ListFields("wellLyr", "Mar_Value")
303.         fieldCountMarValueFalse = len(fieldListMarValueFalse)
304.         if (fieldCountMarValueFalse == 1):
305.             print("Mar_Value field was already created")
306.             arcpy.CalculateField_management("wellLyr", "Mar_Value", "0", "PYTHON_9.3
", "")
307.         else:
308.             arcpy.AddField_management("wellLyr", "Mar_Value", "DOUBLE", "", "", "")
309.             arcpy.CalculateField_management("wellLyr", "Mar_Value", "0", "PYTHON_9.3
", "")
310.
311.         # Check if Mar_Cost field exist, create field if not. Calculate 0 for the M
ar_Cost
312.         fieldListMarCostFalse = arcpy.ListFields("wellLyr", "Mar_Cost")
313.         fieldCountMarCostFalse = len(fieldListMarCostFalse)
314.         if (fieldCountMarCostFalse == 1):
315.             print("Mar_Cost field was already created")
316.             arcpy.CalculateField_management("wellLyr", "Mar_Cost", "0", "PYTHON_9.3"
, "")
317.         else:
318.             arcpy.AddField_management("wellLyr", "Mar_Cost", "DOUBLE", "", "", "")
319.             arcpy.CalculateField_management("wellLyr", "Mar_Cost", "0", "PYTHON_9.3"
, "")
320.
321.         # Check if Mar_Potential_MCF field exist, create field if not with a value o
f 0
322.         fieldListMarMCFfalse = arcpy.ListFields("wellLyr", "Mar_Potential_MCF")
323.         fieldCountMarMCFfalse = len(fieldListMarMCFfalse)
324.         if (fieldCountMarMCFfalse == 1):
325.             print("Mar_Potential_MCF field was already created")
326.             arcpy.CalculateField_management("wellLyr", "Mar_Potential_MCF", "0", "PY
THON_9.3", "")
327.         else:
328.             arcpy.AddField_management("wellLyr", "Mar_Potential_MCF", "FLOAT", "", "
", "")

```

```
329.         arcpy.CalculateField_management("wellLyr", "Mar_Potential_MCF", "0", "PY
    THON_9.3", "")
330.
331.
332.     # Utica check box
333.     if str(uticaTrue) == 'true':
334.         arcpy.AddMessage("The Utica box was checked")
335.
336.     # Do some Calculations
337.     uticaLateralCount = int(uticaLateralCount_str)
338.     uticaLateralLength = int(uticaLateralLength_str)
339.     unleasedUticaAC = 1 - (float(uticaLeasehold_str) / 100)
340.
341.     # Check if Uti_Value field exist, create field if not
342.     fieldListUti = arcpy.ListFields("wellLyr", "Uti_Value")
343.     fieldCountUti = len(fieldListUti)
344.     if (fieldCountUti == 1):
345.         print("Uti_Value field was already created")
346.     else:
347.         arcpy.AddField_management("wellLyr", "Uti_Value", "DOUBLE", "", "", "")
348.
349.     # Create an update cursor to update the for Uti_Value field
350.     with arcpy.da.UpdateCursor("wellLyr", ["Uti_Value"]) as cursor:
351.         for row in cursor:
352.             row[0] = uticaPotential * hubValue
353.             cursor.updateRow(row)
354.         del row, cursor
355.
356.     # Check if Uti_Cost field exist, create field if not
357.     fieldListUtiCost = arcpy.ListFields("wellLyr", "Uti_Cost")
358.     fieldCountUtiCost = len(fieldListUtiCost)
359.     if (fieldCountUtiCost == 1):
360.         print("Uti_Cost field was already created")
361.     else:
362.         arcpy.AddField_management("wellLyr", "Uti_Cost", "DOUBLE", "", "", "")
363.
364.     # Create an update cursor to update the for Uti_Cost field
365.     with arcpy.da.UpdateCursor("wellLyr", ["Unit_Acres", "Uti_Cost"]) as cursor:
366.         for row in cursor:
367.             row[1] = round(((uticaLateralCostFT * uticaLateralCount * uticaLater
    allength) + (row[0] * unleasedUticaAC * uticaLeaseCostAC)) * percentOwnership, 2)
368.             cursor.updateRow(row)
369.         del row, cursor
370.
371.     # Check if Uti_Potential_MCF field exist, create field if not
372.     fieldListUtiMCF = arcpy.ListFields("wellLyr", "Uti_Potential_MCF")
373.     fieldCountUtiMCF = len(fieldListUtiMCF)
374.     if (fieldCountUtiMCF == 1):
375.         # Execute ExtractValuesToPoints
376.         ExtractMultiValuesToPoints("wellLyr", [[utica_kriging, "Uti_Potential_MC
    F_1"]], "BILINEAR")
377.
378.     # Create an update cursor to update the for Uti_Potential_MCF field
379.     with arcpy.da.UpdateCursor("wellLyr", ["Uti_Potential_MCF", "Uti_Potenti
    al_MCF_1"]) as cursor:
380.         for row in cursor:
381.             row[0] = percentOwnership * row[1]
```

```
382.             cursor.updateRow(row)
383.             del row, cursor
384.
385.             # Execute DeleteField
386.             arcpy.DeleteField_management("wellLyr", ["Uti_Potential_MCF_1"])
387.
388.         else:
389.             # Execute ExtractValuesToPoints
390.             ExtractMultiValuesToPoints("wellLyr", [[utica_kriging, "Uti_Potential_MC
F"]], "BILINEAR")
391.
392.         else:
393.             # In this case, the check box value is 'false', user did not check the box
394.             arcpy.AddMessage("The Utica box was not checked")
395.
396.             # Check if Uti_Value field exist, create field if not. Calculate 0 for the
Uti_Value
397.             fieldListUtiValueFalse = arcpy.ListFields("wellLyr", "Uti_Value")
398.             fieldCountUtiValueFalse = len(fieldListUtiValueFalse)
399.             if (fieldCountUtiValueFalse == 1):
400.                 print("Uti_Value field was already created")
401.                 arcpy.CalculateField_management("wellLyr", "Uti_Value", "0", "PYTHON_9.3
", "")
402.             else:
403.                 arcpy.AddField_management("wellLyr", "Uti_Value", "DOUBLE", "", "", "")
404.
405.                 arcpy.CalculateField_management("wellLyr", "Uti_Value", "0", "PYTHON_9.3
", "")
406.
407.             # Check if Uti_Cost field exist, create field if not. Calculate 0 for the Ut
i_Cost
408.             fieldListUtiCostFalse = arcpy.ListFields("wellLyr", "Uti_Cost")
409.             fieldCountUtiCostFalse = len(fieldListUtiCostFalse)
410.             if (fieldCountUtiCostFalse == 1):
411.                 print("Uti_Cost field was already created")
412.                 arcpy.CalculateField_management("wellLyr", "Uti_Cost", "0", "PYTHON_9.3
", "")
413.             else:
414.                 arcpy.AddField_management("wellLyr", "Uti_Cost", "DOUBLE", "", "", "")
415.                 arcpy.CalculateField_management("wellLyr", "Uti_Cost", "0", "PYTHON_9.3
", "")
416.
417.             # Check if Uti_Potential_MCF field exist, create field if not with a value o
f 0
418.             fieldListUtiMCFfalse = arcpy.ListFields("wellLyr", "Uti_Potential_MCF")
419.             fieldCountUtiMCFfalse = len(fieldListUtiMCFfalse)
420.             if (fieldCountUtiMCFfalse == 1):
421.                 print("Uti_Potential_MCF field was already created")
422.                 arcpy.CalculateField_management("wellLyr", "Uti_Potential_MCF", "0", "PY
THON_9.3", "")
423.             else:
424.                 arcpy.AddField_management("wellLyr", "Uti_Potential_MCF", "FLOAT", "", "
", "")
425.                 arcpy.CalculateField_management("wellLyr", "Uti_Potential_MCF", "0", "PY
THON_9.3", "")
426.
427.             # Burket check box
428.             if str(burketTrue) == 'true':
```

```
429.         arcpy.AddMessage("The Burket box was checked")
430.
431.         # Do some Calculations
432.         burketLateralCount = int(burketLateralCount_str)
433.         burketLateralLength = int(burketLateralLength_str)
434.         unleashedBurketAC = 1 - (float(burketLeasehold_str) / 100)
435.
436.         # Check if Bur_Value field exist, create field if not
437.         fieldListBur = arcpy.ListFields("wellLyr", "Bur_Value")
438.         fieldCountBur = len(fieldListBur)
439.         if (fieldCountBur == 1):
440.             print("Bur_Value field was already created")
441.         else:
442.             arcpy.AddField_management("wellLyr", "Bur_Value", "DOUBLE", "", "", "")
443.
444.         # Create an update cursor to update the for Bur_Value field
445.         with arcpy.da.UpdateCursor("wellLyr", ["Bur_Value"]) as cursor:
446.             for row in cursor:
447.                 row[0] = burketPotential * hubValue
448.                 cursor.updateRow(row)
449.             del row, cursor
450.
451.         # Check if Bur_Cost field exist, create field if not
452.         fieldListBurCost = arcpy.ListFields("wellLyr", "Bur_Cost")
453.         fieldCountBurCost = len(fieldListBurCost)
454.         if (fieldCountBurCost == 1):
455.             print("Bur_Cost field was already created")
456.         else:
457.             arcpy.AddField_management("wellLyr", "Bur_Cost", "DOUBLE", "", "", "")
458.
459.         # Create an update cursor to update the for Bur_Cost field
460.         with arcpy.da.UpdateCursor("wellLyr", ["Unit_Acres", "Bur_Cost"]) as cursor:
461.             for row in cursor:
462.                 row[1] = round(((burketLateralCostFT * burketLateralCount * burketLa
463.                 teralLength) + (row[0] * unleashedBurketAC * burketLeaseCostAC)) * percentOwnership, 2)
464.                 cursor.updateRow(row)
465.             del row, cursor
466.
467.         # Check if Bur_Potential_MCF field exist, create field if not
468.         fieldListBurMCF = arcpy.ListFields("wellLyr", "Bur_Potential_MCF")
469.         fieldCountBurMCF = len(fieldListBurMCF)
470.         if (fieldCountBurMCF == 1):
471.             # Execute ExtractValuesToPoints
472.             ExtractMultiValuesToPoints("wellLyr", [[burket_kriging, "Bur_Potential_M
473.             CF_1"]], "BILINEAR")
474.
475.         # Create an update cursor to update the for Bur_Potential_MCF field
476.         with arcpy.da.UpdateCursor("wellLyr", ["Bur_Potential_MCF", "Bur_Potenti
477.         al_MCF_1"]) as cursor:
478.             for row in cursor:
479.                 row[0] = round(percentOwnership * row[1], 2)
480.                 cursor.updateRow(row)
481.             del row, cursor
482.
483.         # Execute DeleteField
484.         arcpy.DeleteField_management("wellLyr", ["Bur_Potential_MCF_1"])
```

```
482.
483.         else:
484.             # Execute ExtractValuesToPoints
485.             ExtractMultiValuesToPoints("wellLyr", [[burket_kriging, "Bur_Potential_M
CF"]], "BILINEAR")
486.
487.         else:
488.             # In this case, the check box value is 'false', user did not check the box
489.             arcpy.AddMessage("The Burket box was not checked")
490.
491.             # Check if Bur_Value field exist, create field if not. Calculate 0 for the
Bur_Value
492.             fieldListBurValueFalse = arcpy.ListFields("wellLyr", "Bur_Value")
493.             fieldCountBurValueFalse = len(fieldListBurValueFalse)
494.             if (fieldCountBurValueFalse == 1):
495.                 print("Bur_Value field was already created")
496.                 arcpy.CalculateField_management("wellLyr", "Bur_Value", "0", "PYTHON_9.3
", "")
497.             else:
498.                 arcpy.AddField_management("wellLyr", "Bur_Value", "DOUBLE", "", "", "")
499.                 arcpy.CalculateField_management("wellLyr", "Bur_Value", "0", "PYTHON_9.3
", "")
500.
501.             # Check if Mar_Cost field exist, create field if not. Calculate 0 for the B
ur_Cost
502.             fieldListBurCostFalse = arcpy.ListFields("wellLyr", "Bur_Cost")
503.             fieldCountBurCostFalse = len(fieldListBurCostFalse)
504.             if (fieldCountBurCostFalse == 1):
505.                 print("Bur_Cost field was already created")
506.                 arcpy.CalculateField_management("wellLyr", "Bur_Cost", "0", "PYTHON_9.3"
, "")
507.             else:
508.                 arcpy.AddField_management("wellLyr", "Bur_Cost", "DOUBLE", "", "", "")
509.                 arcpy.CalculateField_management("wellLyr", "Bur_Cost", "0", "PYTHON_9.3"
, "")
510.
511.             # Check if Bur_Potential_MCF field exist, create field if not with a value o
f 0
512.             fieldListBurMCFfalse = arcpy.ListFields("wellLyr", "Bur_Potential_MCF")
513.             fieldCountBurMCFfalse = len(fieldListBurMCFfalse)
514.             if (fieldCountBurMCFfalse == 1):
515.                 print("Bur_Potential_MCF field was already created")
516.                 arcpy.CalculateField_management("wellLyr", "Bur_Potential_MCF", "0", "PY
THON_9.3", "")
517.             else:
518.                 arcpy.AddField_management("wellLyr", "Bur_Potential_MCF", "FLOAT", "", "
", "")
519.                 arcpy.CalculateField_management("wellLyr", "Bur_Potential_MCF", "0", "PY
THON_9.3", "")
520.
521.
522.             # Check is wellpad is existing, update field as needed
523.             if str(existingWellpad) == 'true':
524.                 print("Wellpad_Dev_Cost is calculated, do nothing")
525.             else:
526.                 # Create an update cursor to update the for Wellpad_Dev_Cost and Forest_Loss
fields to 0 if pad is existing
```

```

527.         with arcpy.da.UpdateCursor("wellLyr", ["Wellpad_Dev_Cost", "Forest_Loss"]) a
           s cursor:
528.             for row in cursor:
529.                 row[0] = 0
530.                 row[1] = 0
531.                 cursor.updateRow(row)
532.             del row, cursor
533.
534.         # Calculate total wellpad cost
535.         fieldListTotalCost = arcpy.ListFields("wellLyr", "Total_Cost")
536.         fieldCountTotalCost = len(fieldListTotalCost)
537.         if (fieldCountTotalCost == 1):
538.             print("Total_Cost field was already created")
539.             arcpy.CalculateField_management("wellLyr", "Total_Cost", "round(!Wellpad_Dev
_Cost! + !Mar_Cost! + !Uti_Cost! + !Bur_Cost!, 2)", "PYTHON_9.3", "")
540.         else:
541.             arcpy.AddField_management("wellLyr", "Total_Cost", "DOUBLE", "", "", "")
542.             arcpy.CalculateField_management("wellLyr", "Total_Cost", "round(!Wellpad_Dev
_Cost! + !Mar_Cost! + !Uti_Cost! + !Bur_Cost!)", "PYTHON_9.3", "")
543.
544.         # Calculate total potential MCF
545.         fieldListTotalMCF = arcpy.ListFields("wellLyr", "Total_Potential_MCF")
546.         fieldCountTotalMCF = len(fieldListTotalMCF)
547.         if (fieldCountTotalCost == 1):
548.             print("Total_Potential_MCF field was already created")
549.             arcpy.CalculateField_management("wellLyr", "Total_Potential_MCF", "round(!Ma
r_Potential_MCF! + !Uti_Potential_MCF! + !Bur_Potential_MCF!, 2)", "PYTHON_9.3", "")
550.         else:
551.             arcpy.AddField_management("wellLyr", "Total_Potential_MCF", "DOUBLE", "", ""
, "")
552.             arcpy.CalculateField_management("wellLyr", "Total_Potential_MCF", "round(!Ma
r_Potential_MCF! + !Uti_Potential_MCF! + !Bur_Potential_MCF!, 2)", "PYTHON_9.3", "")

```

Appendix A, Script 4: Source code for Process 3 validation in Python

```

1. import arcpy
2. class ToolValidator(object):
3.     """Class for validating a tool's parameter values and controlling
4.     the behavior of the tool's dialog."""
5.
6.     def __init__(self):
7.         """Setup arcpy and the list of tool parameters."""
8.         self.params = arcpy.GetParameterInfo()
9.
10.    def initializeParameters(self):
11.        """Refine the properties of a tool's parameters. This method is
12.        called when the tool is opened."""
13.        return
14.
15.    def updateParameters(self):
16.        """Modify the values and properties of parameters before internal
17.        validation is performed. This method is called whenever a parameter
18.        has been changed."""
19.        return
20.
21.    def updateMessages(self):

```



```
22.     """Modify the messages created by internal validation for each tool
23.     parameter. This method is called after internal validation."""
24.     return
25.
26.     def updateParameters(self):
27.         # If the option to use a weights file is selected (the user chose
28.         # "Get Spatial Weights From File"), enable the parameter for specifying
29.         # the file, otherwise disable it
30.         #
31.         # Display if the New Wellpad Location checkbox is checked (True)
32.         if self.params[3].value == True:
33.             self.params[4].enabled = True
34.             self.params[5].enabled = True
35.             self.params[6].enabled = True
36.         else:
37.             self.params[4].enabled = False
38.             self.params[5].enabled = False
39.             self.params[6].enabled = False
40.
41.         # Display if the Marcellus checkbox is checked (True)
42.         if self.params[8].value == True:
43.             self.params[9].enabled = True
44.             self.params[10].enabled = True
45.             self.params[11].enabled = True
46.         else:
47.             self.params[9].enabled = False
48.             self.params[10].enabled = False
49.             self.params[11].enabled = False
50.
51.         # Display if the Utica checkbox is checked (True)
52.         if self.params[12].value == True:
53.             self.params[13].enabled = True
54.             self.params[14].enabled = True
55.             self.params[15].enabled = True
56.         else:
57.             self.params[13].enabled = False
58.             self.params[14].enabled = False
59.             self.params[15].enabled = False
60.
61.         # Display if the Burket checkbox is checked (True)
62.         if self.params[16].value == True:
63.             self.params[17].enabled = True
64.             self.params[18].enabled = True
65.             self.params[19].enabled = True
66.         else:
67.             self.params[17].enabled = False
68.             self.params[18].enabled = False
69.             self.params[19].enabled = False
```

Appendix B: Data Sources

Land Cover (Pre-Exploration) - PAMAP Program Land Cover for Pennsylvania, 2005 (30-meter resolution)

<http://www.pasda.psu.edu/uci/DataSummary.aspx?dataset=1100>

Land Cover (Post-Exploration) - High-Resolution Land Cover, Commonwealth of Pennsylvania, Chesapeake Bay Watershed and Delaware River Basin, 2013 (1-meter resolution)

<http://www.pasda.psu.edu/uci/DataSummary.aspx?dataset=3193>

Well Data - Reported Production from the Pennsylvania DEP

http://www.depreportingservices.state.pa.us/ReportServer/Pages/ReportViewer.aspx?%2fOil_Gas%2fOil_Gas_Well_Production

Unit Declaration Data – Digitized from data recorded in PA County Courthouses

Williams Partners L.P. existing Susquehanna County gathering lines

<http://atlanticsunriseexpansion.com/wp-content/uploads/2015/06/Susquehanna-4-29-15.pdf>

Digital Elevation Model from the 2006 - 2008 - DCNR PAMAP Program -

<http://www.pasda.psu.edu/uci/DataSummary.aspx?dataset=1247>

EIA shale formation isopach and elevation data

https://www.eia.gov/maps/layer_info-m.php

EIA Natural Gas Interstate and Intrastate Pipelines

https://www.eia.gov/maps/layer_info-m.php