

2021

EEMS 3.1 Manual

Environmental Evaluation Modeling System
Manual Version 1.0.2

*An introduction to the principles of fuzzy logic, and an overview of the
Environmental Evaluation Modeling System (EEMS) framework.*

Tim Sheehan; Mike Gough; James Strittholt
Conservation Biology Institute
4/7/2021



Table of Contents

TABLE OF CONTENTS.....	1
I. INTRODUCTION.....	2
1.1 Purpose and Contents	2
1.2 Software and Technical Requirements.....	2
II. OVERVIEW OF EEMS & FUZZY LOGIC.....	3
2.1 About EEMS	3
2.2 An Overview of Fuzzy Logic	3
2.3 EEMS Tools (Operators).....	8
Choosing Operators for Your Logic Model.....	9
III. EEMS For ArcGIS.....	12
3.1 The ABC's of EEMS for ArcGIS	12
3.2 The EEMS Modeling Process	14
Preliminary Steps	14
Phase 1 (Preparing Input Data).....	16
Phase 2 (Creating or defining reporting units)	17
Phase 3 (Quantitative Evaluation, Calculating Densities)	18
Phase 4 (Building and running an EEMS model)	19
3.3 Tutorial:	
Building and Running an EEMS model in ArcGIS.....	19
REFERENCES CITED	30

I. Introduction

1.1 Purpose and Contents

This document is intended to help the reader understand how to use the EEMS Fuzzy Logic Modeling System (Sheehan and Gough, 2016).

How this document is organized

Section I provides an introduction to the EEMS manual. Section II provides a background on fuzzy logic as well as a high level overview of the EEMS Modeling Framework. Section III describes the EEMS modeling process as implemented from within the ArcGIS environment, and concludes with a tutorial on building and running an EEMS model using some of the tools available in the EEMS for ArcGIS Toolbox.

It is recommended that the user read through Sections I, II, and III in order to obtain a conceptual understanding of the EEMS modeling process prior to working through the tutorial at the end of section III.

1.2 Software and Technical Requirements

1. **EEMS 3.1.x ArcGIS Toolbox (Download Link Below)**
https://eemsonline.org/static/arc/EEMS3.1.0_ArcGIS.zip
2. **Microsoft Windows 7 or greater**
3. **ArcGIS Desktop 10.6 or greater, including the Spatial Analyst extension.**
While earlier versions of ArcGIS may work, they have not been tested.
4. **MPilot** - a plugin-based environmental modeling framework written in python.
To install MPilot, follow the instructions below:
 - i. Open up a command prompt (in Windows 10, you can do this by typing "cmd" into the Search box in the bottom left hand corner of your screen).
 - ii. Type in "cd C:\Python27\ArcGIS10.X\Scripts", replacing "X" with the version of ArcGIS Desktop you have installed.
 - iii. Type in "pip install mpilot".
 - iv. Restart ArcCatalog & ArcMap.

More information about MPilot can be found on GitHub: <https://github.com/consbio/mpilot>

In addition to the software requirements listed above, the user should have formal GIS training, and be experienced with ArcGIS and ArcGIS Model Builder.

II. OVERVIEW OF EEMS & FUZZY LOGIC

2.1 About EEMS

The Environmental Evaluation Modeling System (EEMS) is an implementation of an evaluative logic modeling system. EEMS was developed by Tim Sheehan, an ecological modeler at the Conservation Biology Institute, and has been applied by CBI scientists in a range of ecological evaluations. In the Tehachapi and Southern Sierra regions of California, a model incorporating data for habitat presence, habitat linkage, and disturbance was used to find areas of high ecological value and to provide guidance for reserve design and to inform siting wind energy. For the Bureau of Land Management Rapid Ecological Assessments of the Sonoran Desert and Colorado Plateau ecoregions, several EEMS models were developed and used to evaluate a variety of current and projected ecological metrics.

2.2 An Overview of Fuzzy Logic

In the discipline of GIS evaluative modeling, a logic model is a cognitive map (Jensen et al. 2009) that presents networks of various spatial data components and their logical relationships to explain the process used to evaluate a complex topic such as terrestrial intactness.

EEMS logic models rely solely on spatial data layers arranged in a tree-based, hierarchical fashion (Figure 1-1, Figure 1-2) to answer a primary question. Data and analysis flow from the bottom up. At the bottom of the tree, the *leaf nodes* represent the initial data inputs. Data from the leaves are combined recursively, moving up *branches*, until the single *root node*, representing the answer to the primary question, is reached. In this example, the evaluation pertains to the level of terrestrial landscape intactness.

Unlike conventional GIS applications that use Boolean logic (True/False or 1/0) or scored input layers, evaluative logic models rely on fuzzy logic. Simply put, fuzzy logic allows the user to assign shades of gray to thoughts and ideas rather than being restricted to black (false) and white (true) determinations. All data inputs (regardless of the type—ordinal, nominal, or continuous) are converted into fuzzy values between -1 (false) and +1 (true) up to six decimal places. A user defined function converts inputs from original values (often referred to as *raw values* or values in *raw space*) to fuzzy values (values in fuzzy space). Values in raw space may be combined using mathematical operators (e.g. sum, weighted mean) before being translated into fuzzy space. Values in fuzzy space are combined using fuzzy logic operators (e.g. AND, OR). There are many advantages of this modeling approach: (1) it is highly interactive and flexible; (2) it is easy to visualize thought processes; (3) the logic components are modular making it easy to include or exclude pieces of the logic design; (4) the logic can be managed using a number of different mechanisms; and (5) numerous, diverse topics can be included into a single integrated analysis. Raw, or non-fuzzy, spatial data source inputs (boxes in the example) are populated by one or more GIS data layers (indicated by the stack of gray files). Moving up the diagram, these data are arranged and

analyzed to form intermediate map products (purple boxes), which are then arranged and analyzed to generate the final results (green box). One way the user controls the logic of the information is the arrangement of the various data inputs and intermediate products—the higher up in the diagram, the greater the influence on the final result.

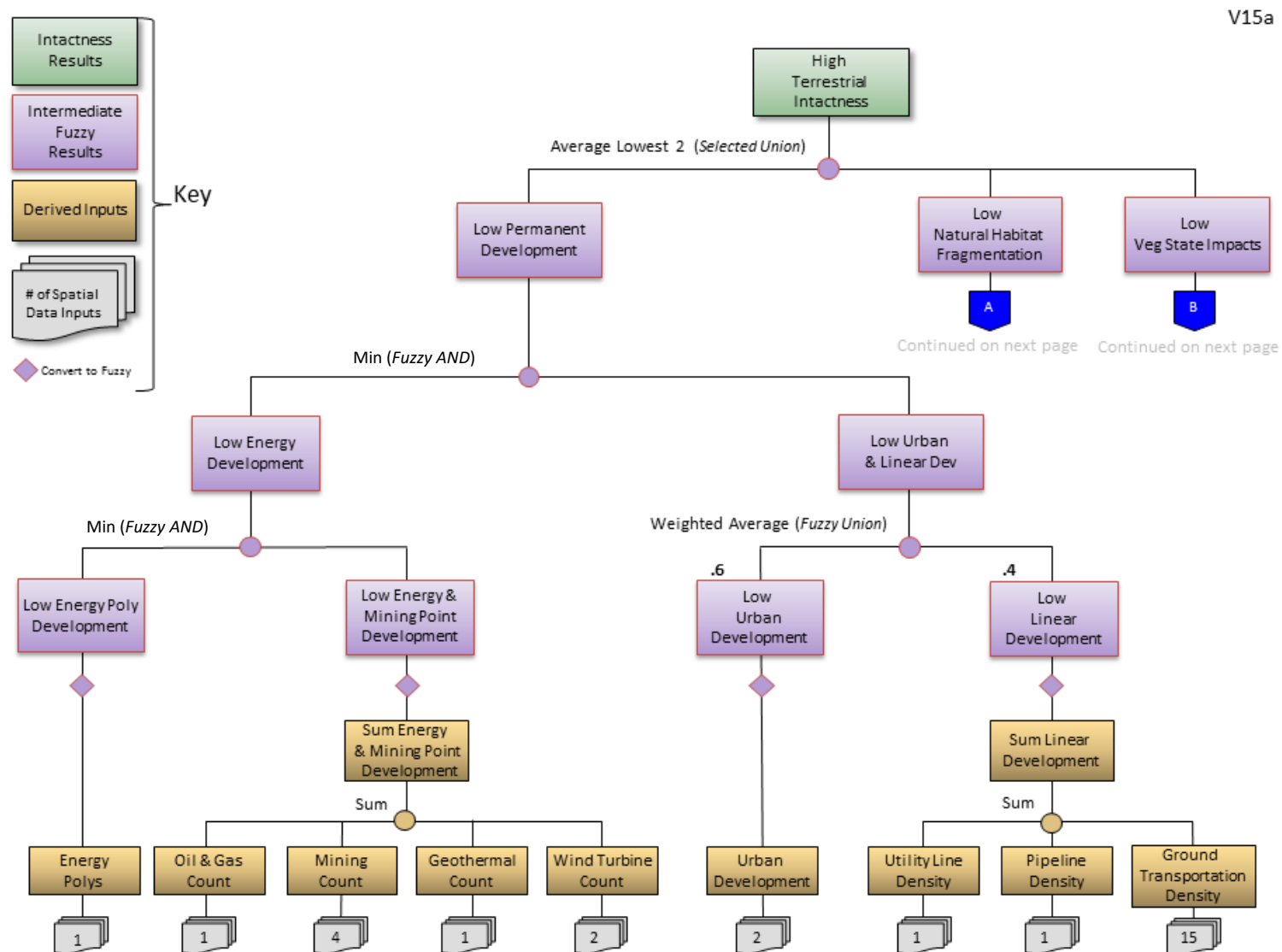
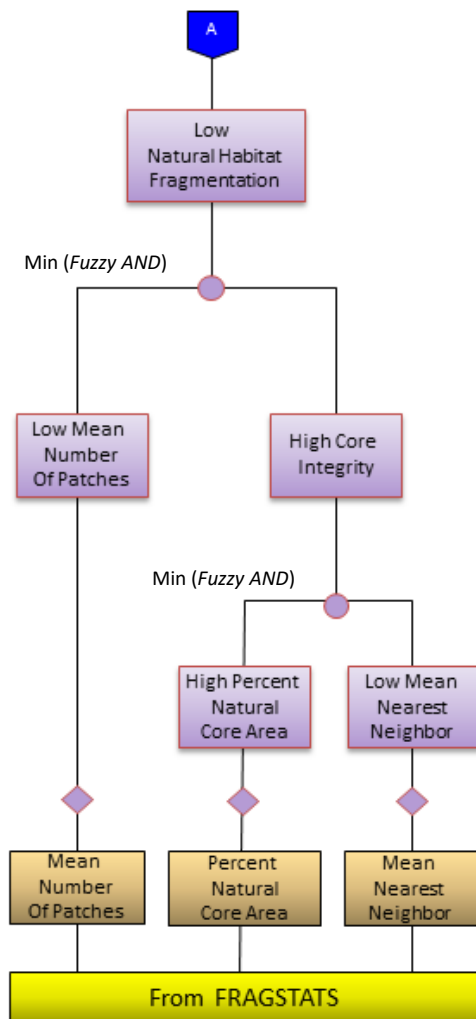


Figure 1-1. Logic model for terrestrial landscape intactness (Page 1 of 2)

Continued on previous page



Continued on previous page

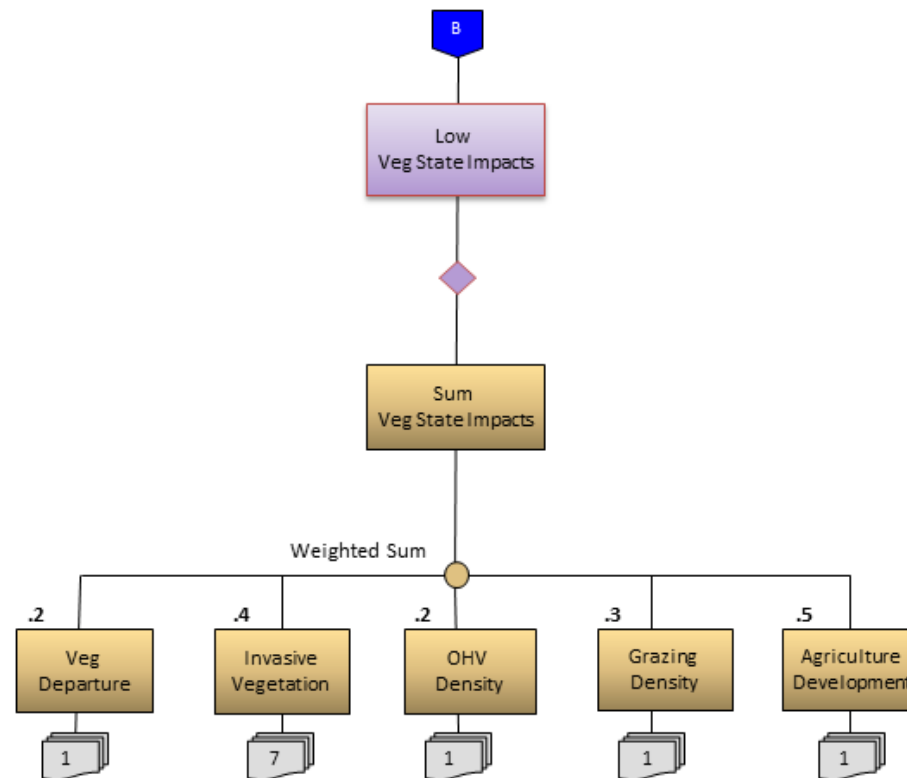


Figure 1-2. Logic model for terrestrial landscape intactness (Page 2 of 2)

Using fuzzy logic as the core modeling principle, evaluative logic model performance is achieved in several ways. For every spatial data input, the user determines how to assign the range of values along a truth continuum. For example, when trying to determine and map the most suitable habitat from the standpoint of road density for wildlife—the greater the road density, the greater is the risk to wildlife through habitat degradation and direct mortality. In our example, road density ranges from 0 km/km² to 24.5 km/km². To assign a fuzzy logic continuum for this range of values, one could assign a -1 to the high value (this value is totally harmful for wildlife or false) and a +1 to the lowest value (this value is totally beneficial for wildlife, or true; red line in Figure 1-3). However, mountain lion research has shown that mountain lion populations have a low probability of persistence in areas with road densities > 0.6 km/km² (Van Dyke et al. 1986). A more meaningful alternative then for setting fuzzy thresholds for this parameter would be that a road density of > 0.6 km/km² is totally false (-1) and 0 remains totally true (+1, green line in Figure 1-3). Of course, not all wildlife species have the same sensitivity to roads, but this example illustrates how the logic in the model can be altered for known thresholds.

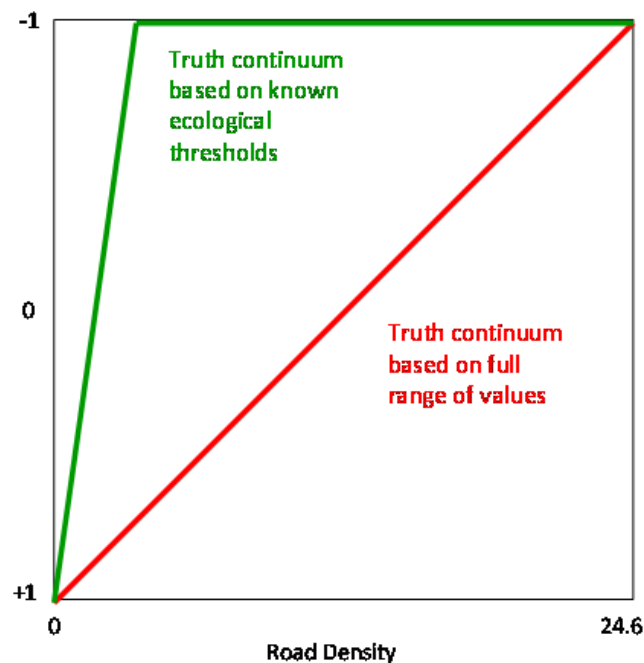


Figure 1-3. Diagram of two treatments of road density in fuzzy logic modeling illustrating important model control options, one based on a full range of values (red line) and the other based on a known threshold for road density (> 0.60 km/km² is totally false [-1], green line).

2.3 EEMS Tools (Operators)

EEMS can be implemented from within an ArcGIS Model Builder Model, or as a standalone program. The tables below list the EEMS tools available within the ArcGIS EEMS Toolbox as well as the corresponding EEMS command that the tool references. The last column describes the functionality of the tool.

The first table below lists the core set of EEMS tools which perform functions specific to the EEMS framework.

ARCGIS TOOL	EEMS COMMAND	DESCRIPTION
EEMS MODEL INITIALIZE	(N/A, applies only to Arc version)	Specifies the paths to the files needed to run EEMS (input reporting units & EEMS command file). Clears the contents of the EEMS command file if it exists
EEMS MODEL LOGIC CHECK	(N/A, applies only to Arc version)	Check for logical consistencies within the EEMS command file.
EEMS MODEL RUN	(N/A, applies only to Arc version)	Executes the commands in the EEMS command file on the input reporting units. Stores the results in the output reporting units.
EEMS READ	EEMSRead	Reads a variable from the input reporting units.

EEMS also comes with the set of logic and mathematical operators used to process or combine input data or derived data. These operators are described in the table below. The operators used and the order in which they are executed will depend on the question under evaluation. Note that some of the operators are intended to operate on raw values while others are intended to operate on fuzzy values.

ARCGIS TOOL	EEMS COMMAND	INPUT DATA	DESCRIPTION
CONVERT FROM FUZZY	CvtFromFuzzy	Fuzzy	Converts input fuzzy values into non-fuzzy values using linear interpolation
CONVERT TO BINARY	CvtToBinary	Raw	Converts input values into binary 0 or 1 based on threshold. Direction = LowToHigh for values below threshold to be false and above to be true. Direction = HighToLow for values below threshold to be true and above to be false.
CONVERT TO FUZZY	CvtToFuzzy	Raw	Converts input values into fuzzy values using linear interpolation.
CONVERT TO FUZZY CATEGORY	CvtToFuzzyCat	Raw	Converts integer input values into fuzzy based on user specification.
CONVERT TO FUZZY CURVE	CvtToFuzzyCurve	Raw	Converts input values into fuzzy based on user-defined curve.

CONVERT TO FUZZY MEAN TO MID	CvtToFuzzyMeanToMid	Raw	Converts input values into fuzzy based on the mean to mid-value normalization method. The mean is assigned the middle fuzzy value from the provided list. The mean of all the values below the mean (low mean) is assigned the low mid fuzzy value from the list. The same is done for the high mean. The remaining values are scaled linearly between these five points (-1, -0.5, 0, 0.5, and 1).
CONVERT TO FUZZY CURVE Z SCORE	CvtToFuzzyCurveZScore	Raw	Converts input values into fuzzy based on user-defined curve.
CONVERT TO FUZZY Z SCORE	CvtToFuzzyZScore	Raw	Converts input values into fuzzy values using linear interpolation based on Z Score.
FUZZY AND	FuzzyAnd	Fuzzy	Takes the fuzzy And (minimum) of fuzzy input variables.
FUZZY NOT	FuzzyNot	Fuzzy	Logical NOT for fuzzy modeling. Reverses the sign of values of the input field.
FUZZY OR	FuzzyOr	Fuzzy	Takes the fuzzy Or (maximum) of fuzzy input variables.
FUZZY SELECTED UNION	FuzzySelectedUnion	Fuzzy	Takes the fuzzy Union (mean) of N Truest or Falsest fuzzy input variables.
FUZZY UNION	FuzzyUnion	Fuzzy	Takes the fuzzy Union (mean) of fuzzy input variables.
FUZZY WEIGHTED UNION	FuzzyWeightedUnion	Fuzzy	Takes the weighted fuzzy Union (mean) of fuzzy input variables.
FUZZY X OR	FuzzyXOr	Fuzzy	Computes Fuzzy XOR: Truest - (Truest - 2nd Truest) * (2nd Truest - full False)/(Truest - full False).
MAXIMUM	Maximum	Raw	Takes the maximum of the input variables.
MEAN	Mean	Raw	Mean of input variables.
MINIMUM	Minimum	Raw	Takes the minimum input variables.
MULTIPLY	Multiply	Raw	Multiplies input variables
NORMALIZE	Normalize	Raw	Normalizes the data from another field to range (default 0:1).
SUM	Sum	Raw	Sums input variables.
UNION	Union	Fuzzy	Finds the union value of the inputs (mean value).
WEIGHTED MEAN	WeightedMean	Raw	Takes the weighted mean of input variables.

WEIGHTED SUM	WeightedSum	Raw	Takes the weighted sum of input variables. Multiplies each field by its weight before adding. Like a weighted mean without the division.
X DIVIDED BY Y	ADividedByB	Raw	Performs X / Y
X MINUS Y	AMinusB	Raw	Performs X - Y

Table 1.1: EEMS Tools (Operators)

Choosing Operators for Your Logic Model

EEMS presents the user with choices for many operators and finding the right one can be confusing at first. The guidelines presented here will help you choose the right operator, but remember, sometimes it is best to experiment with several choices to make sure the operator you choose is appropriate for your model.

EEMS has operators designed to work on data before they are converted into fuzzy numerical space (i.e. when they are still in *raw* space) and those designed to work on data after they are converted into fuzzy space (see the above table). A user should respect that distinction. Using a non-fuzzy operator on fuzzy data can produce a result that falls outside the -1 to +1 continuum of fuzzy space. Doing this produces an invalid model.

Weighted Sum

The operators used in raw space are for the most part pretty straightforward. However the Weighted Sum operator merits a discussion. A **Weighted Sum** takes two or more inputs, and multiplies each of them by a weight before adding them. It has proven especially valuable with combining data of very similar types into one result that is then converted into fuzzy space. For example, if you were evaluating a region for intactness, the negative impact of paved roads might be considered similar to but greater than that of dirt roads. Their effects are additive, but a sum operator is not available in fuzzy space. To apply the **Weighted Sum** operator you might provide a weight of 1 to the paved road density and a weight of 0.5 to the dirt road density. In models that have done this, the result has been labeled “Effective Road Density.”

And, Or, and Union

*Note that while the tools that operate on Fuzzy data are technically called “Fuzzy And”, “Fuzzy Or”, “Fuzzy Union”, etc. they will be generally be referred to simply as **And**, **Or**, **Union**, etc., for the remainder of this document.*

And, **Or** and **Union** are the most common EEMS operators used. The choice between **And**, **Or**, and **Union** depends on the relationship of the input data to the question you are asking. **Or** returns the highest fuzzy value of any of the inputs, it is appropriate when any of the inputs is sufficient for your

desired outcome. For example if you were evaluating a region in which three critically endangered species were present in some locations, you could use an **Or** to combine *presence of species A*, *presence of species B*, and *presence of species C* into *high preserve value*. The presence of any of the three species would cause a map reporting unit to have a high fuzzy value. **And** is used when all inputs are necessary for the result to be high. For instance, if both habitat for and presence of a species of interest were required to consider a location as a preserve, you could combine *species presence* and *habitat density* with an **And** to produce *high preservation value*. **And** chooses the lowest fuzzy value of the inputs so that high fuzzy values for both conditions are necessary to yield a high fuzzy value for the result. **Union** takes the mean of the input values. **Union** allows each input to exert an influence on the result. If all inputs have a high value, the result will have a high fuzzy value; if all have a low value, the result will be low. If some are high and some are low, the result will be somewhere in between. Going back to our preserve example, we know if the species is present, the location has value as a preserve. If the habitat is present there is some value, too. If they are both present then the value is the highest. Union will yield that result. A **Weighted Union** is similar to **Union**, except that it allows a weight to the inputs. In our preserve example, if *habitat density* is more important than *species presence* (for instance in an area where remnant populations are under stress and habitat has been restored in areas where the species has not been able to recolonize) then you could provide a greater weight to *habitat density*.

Selected Union

The Selected Union represents a combination of **Or** (or **And**) and **Union**. Consider a study area that includes many different types of habitat, for example, a basin and range terrain. Some species of concern are found in valleys, others inhabit the foothills, and others the high mountains. What if there are 30 species of concern? The more species of concern in a location, the more valuable the location, but nowhere are they all found together. The **Selected Union** allows for the evaluation of such a study area. With the **Selected Union**, you choose a number of the truest (or falsest) of inputs to evaluate. In the basin and range example, you might choose five. A location with a high density of five (or more) species of concern would have a high fuzzy value for *high species diversity*. As the density of species of concern falls, so does the fuzzy value for *high species diversity*. A **Selected Union** with a parameter of 5 Truest would do just that. It performs a **Union** operation on the five inputs with the highest fuzzy values.

Final Word

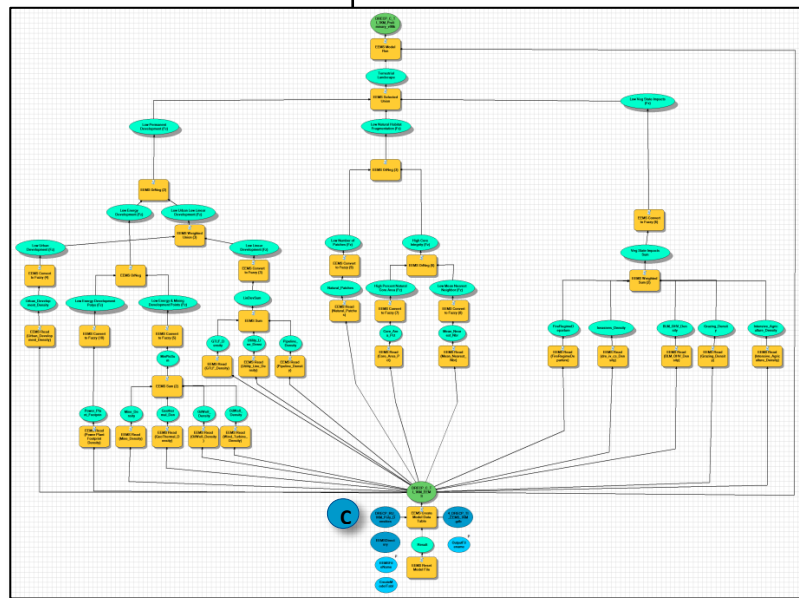
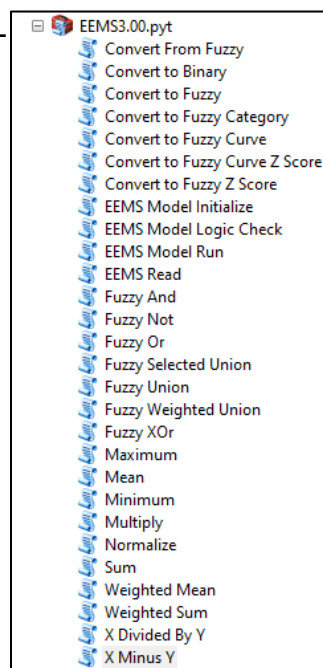
It may not seem so at first, but fuzzy logic evaluative models are not hard to understand, and they are not hard to construct. With some time, some trial and error, and some thought, you can easily manage the concepts, the operators, and the construction of these models. The key is to trust that with experience comes both understanding and expertise.

III. EEMS for ArcGIS

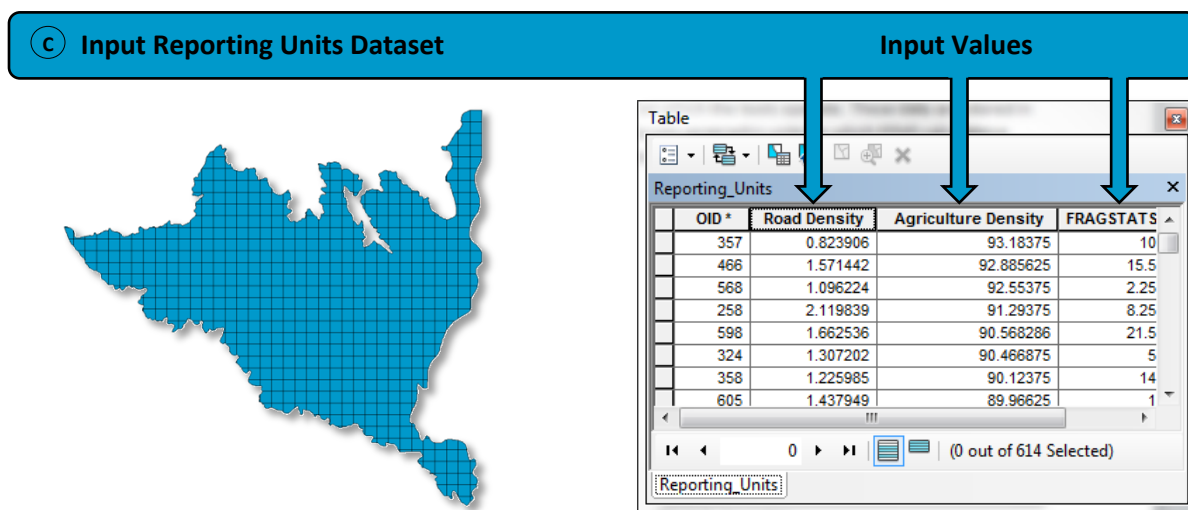
The following section describes the process of designing and building an EEMS model within the ArcGIS Desktop environment. It is organized into 3 parts. The first part (3.1) provides an overview of the EEMS for ArcGIS framework (The ABC's of EEMS for ArcGIS). The second part (3.2) describes the four phases of the EEMS modeling process. And the third part (3.3) consists of a tutorial which will walk you through the steps involved in building an EEMS model.

3.1 The ABC's of EEMS for ArcGIS

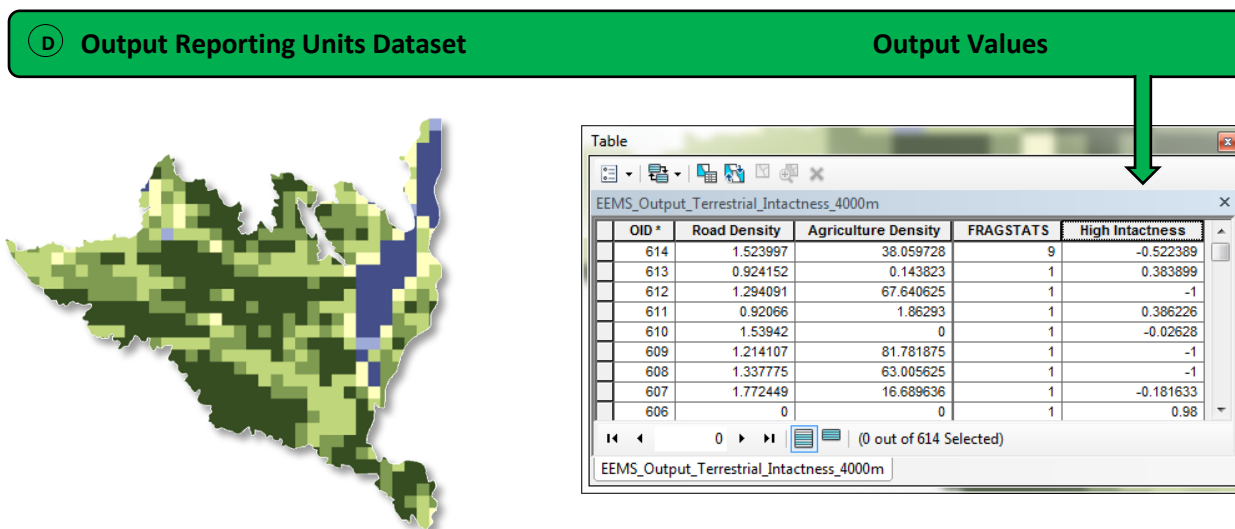
- A** **EEMS** is written in Python and deployed as a collection of script tools within an ArcGIS Toolbox.
- B** **EEMS models** are constructed by bringing these tools into an ArcGIS Model Builder model, where they are connected and arranged in a logical hierarchy. A full list of the EEMS Tools and a description of their functionality is presented in table 1.1.



EEMS tools operate on what is referred to as the **Input Reporting Units Dataset** **C**. All the values for the input variables (e.g., road density) need to be stored in a *single* vector dataset composed of reporting units. Reporting units are discrete geographic units containing quantitative measures of how much (or how little) of each variable is present within that reporting unit. Common measurements include densities, counts, and/or statistical measurements such as the mean or max. Reporting units may be regular geometric shapes (e.g., squares), or they may be in the form of irregular natural or political boundaries, such as watersheds or counties.



After the input reporting units dataset has been created and the EEMS model has been constructed, the model can be executed. When the model is executed, the EEMS tools read in the values in the input reporting units dataset (C), the calculations are performed, and an **output reporting unit dataset** **D** is created which contains a final output value for each reporting unit.



3.2 The EEMS Modeling Process

PRELIMINARY STEPS (BUILDING A CONCEPTUAL MODEL DIAGRAM):

Before preparing your input data and constructing an EEMS model, it is important to perform the following preliminary steps:

1. Think about the question you are trying to answer.
2. Think about what information (raw data) you will need to acquire in order to answer that question.
3. Construct a conceptual model diagram describing how you might fit these pieces of information together in order to answer your question (Figure 2-1)

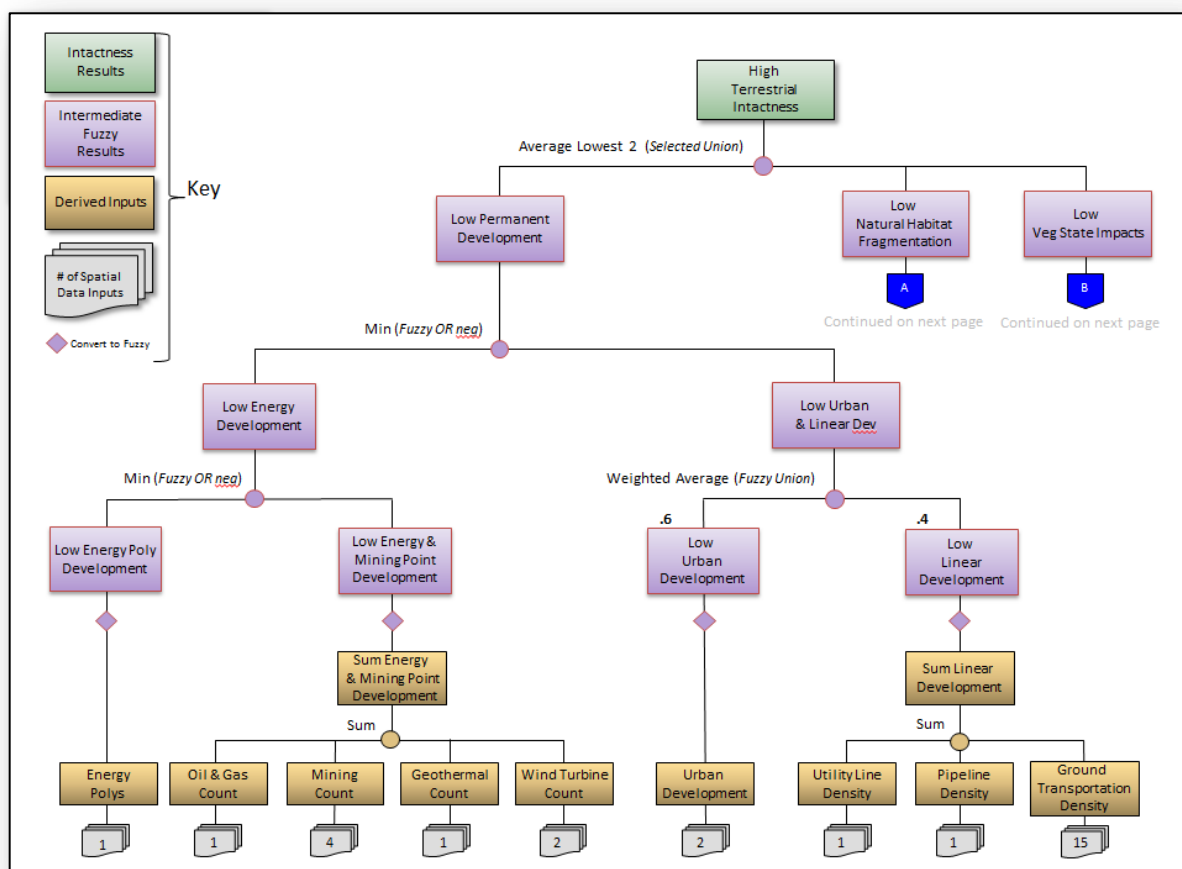


Figure 2-1: A conceptual model diagram (created in MS PowerPoint) describing the logic and operators to be used in constructing a Terrestrial Intactness logic model.

After you've created a conceptual representation of your model, you can proceed with the EEMS modeling process. There are typically four phases involved in this process:

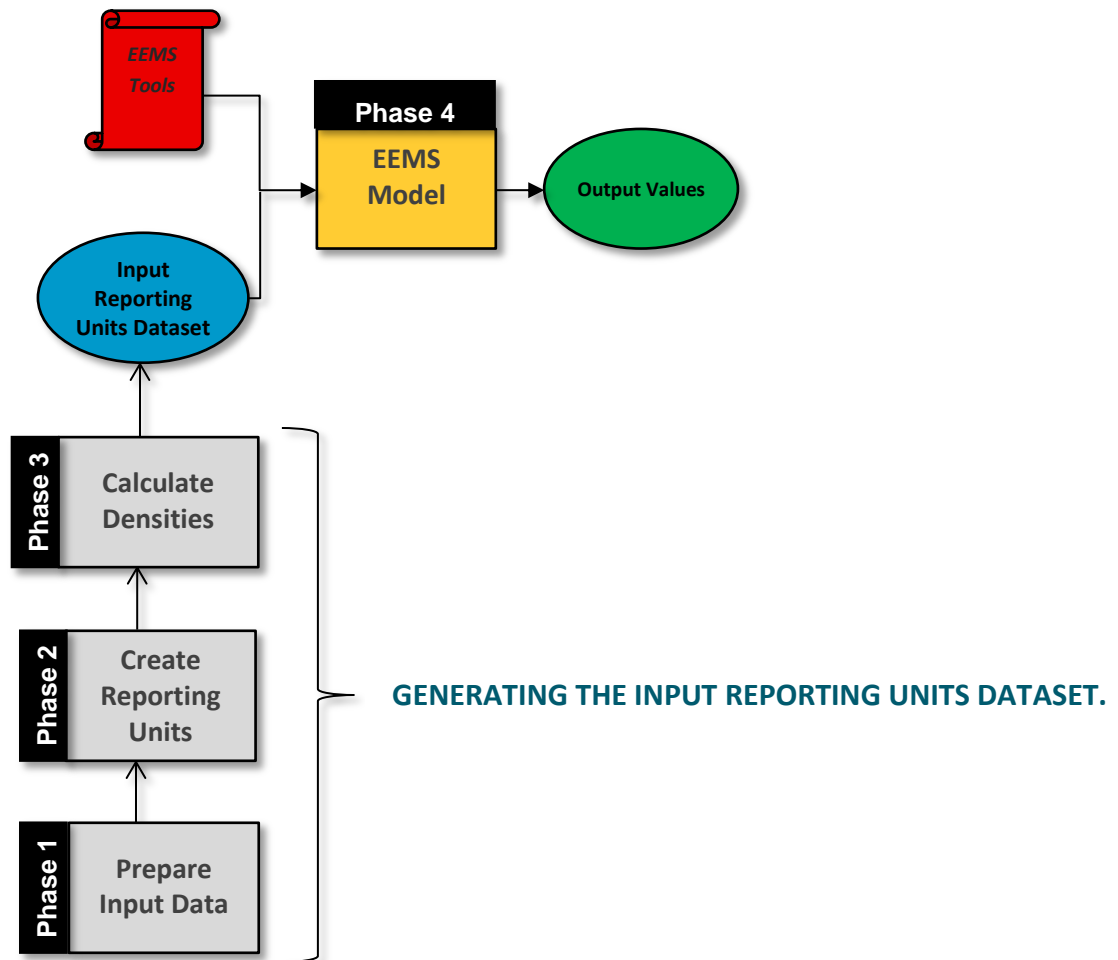
Phase 1: Preparing input data

Phase 2: Creating or defining reporting units

Phase 3: Calculating densities (quantitative evaluation by reporting unit)

Phase 4: Building and Running the EEMS model

As discussed in section 1.4, EEMS operates on a single input dataset (represented in blue below). Each field in the attribute table of this dataset corresponds to an input variable, and each record corresponds to a reporting unit (discrete geographic units for which EEMS calculates a final output value). Consequently, in order to build and execute an EEMS model, one must first generate this master input reporting units dataset which contains all of the input data. This is accomplished in phases 1-3.



Implementation of phases 1-4 is generally carried out within the ArcGIS Model Builder environment using standard ArcGIS geoprocessing tools in conjunction with custom Python scripts.

The analyst has the option of constructing their own models and scripts to perform the tasks required to create the input reporting units dataset, or, alternatively, he or she may use the EEMS support tools developed by CBI. These tools can be downloaded at the URL below.

EEMS Support Tools:

http://consbio.webfactional.com/EEMS/EEMS_Support_Tools.zip

Please note that these tools are not officially a part of the EEMS software package, and are provided *as-is*. CBI does not offer support for these tools at this time. Use of these tools is not required in order to use EEMS, but may be of value in assisting with the preparation and processing of input data.

The following section provides a detailed overview of each of the phases involved in the EEMS modeling process.

PHASE 1 (PREPARING INPUT DATA):

During this phase of the analysis, you will define your study area and pre-processes your input data. The purpose of this phase is to ensure that all of the input data are ready to be quantitatively evaluated within each reporting unit.

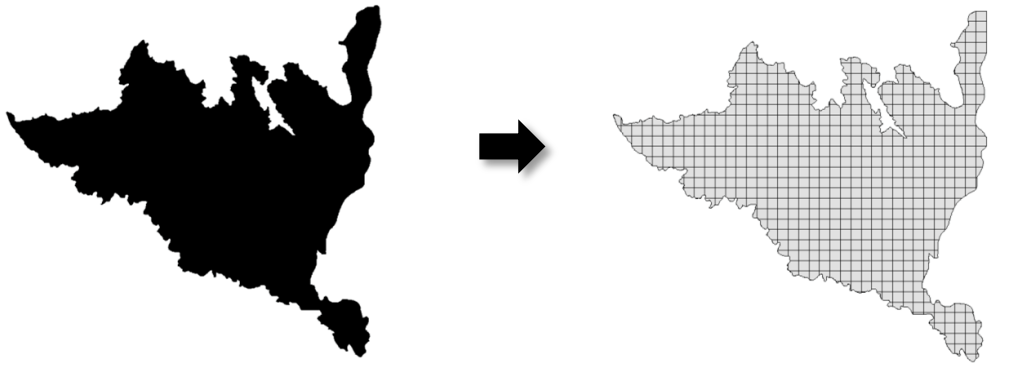
The process of preparing input data is unique to each analysis. Consequently, you will always need to construct your own models and scripts to conduct this phase of the process.

A common workflow for phase 1 is as follows:

1. Obtain or create a dataset which defines the study area.
2. Obtain the input data needed in order to answer the question under evaluation.
3. Project each dataset to a common projected coordinate system.
4. Select features of interest and generate new datasets as necessary.
5. Merge or combine existing datasets as necessary (e.g., roads from multiple counties).
6. Create points from x,y coordinates as necessary.
7. Dissolve input data as necessary.
8. Perform any necessary raster pre-processing
 - a. **NOTE:** Raster datasets which will be used in calculating percent cover by reporting unit should be classified to binary during this phase (1 for presence, 0 for absence).
9. Clip all of your input datasets to the study area.
10. Store all of your input data in a centralized location for input to phase 2.

PHASE 2 (CREATING OR DEFINING REPORTING UNITS):

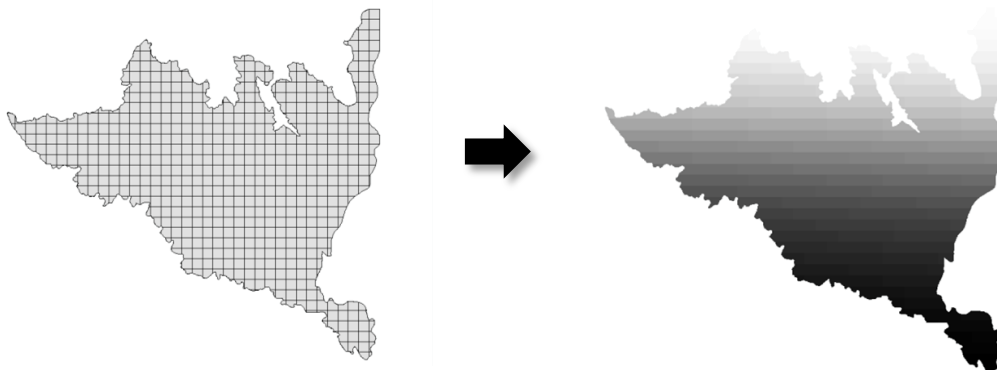
During phase 2, a reporting unit feature class is generated from the study area. The reporting units feature class is typically a vector based grid composed of regular square polygons (for example, 4km² or 1km²). However, as mentioned previously, it may also be in the form of irregular natural or political boundaries, such as watersheds or counties.



The general procedure for creating the input reporting units dataset within ArcGIS is to:

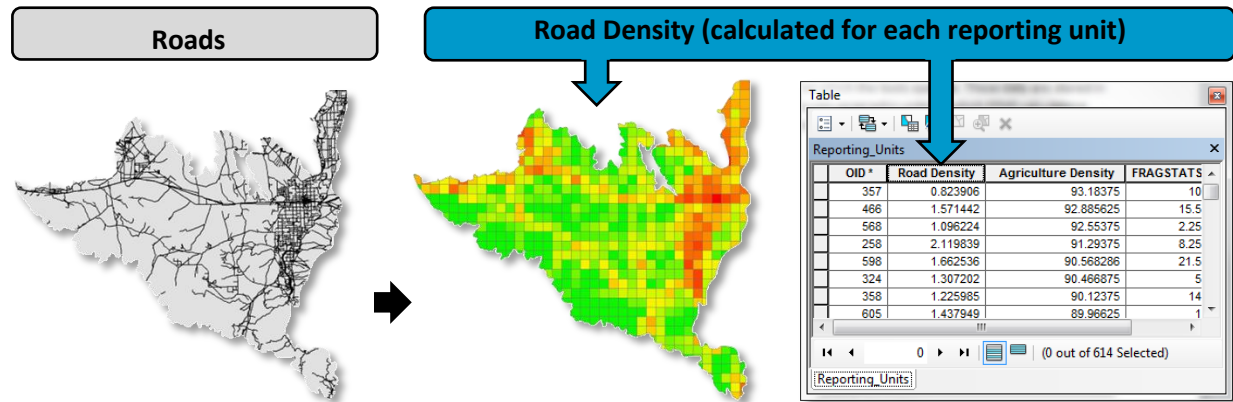
1. Use the **Fishnet** tool to create an empty reporting units dataset, setting the extent to match that of your study area boundary
2. **Clip** the fishnet to your study area boundary.

A corresponding raster version of the reporting units dataset should also be generated for processing any raster data inputs. The raster version of the reporting units dataset is used for generating zonal statistics (e.g., mean vegetation departure within each reporting unit). The cell size of the raster reporting units dataset should match the resolution of your input raster datasets (although ESRI's zonal statistics tool will automatically perform a resampling if this is not the case). When the raster version of the reporting units dataset is created, the output cell values should match the OBJECTIDs of the vector reporting units dataset within which they fall.



PHASE 3 (QUANTITATIVE EVALUATION, CALCULATING DENSITIES):

During this phase, the input datasets processed in phase 1 are quantitatively evaluated by reporting unit. That is, within each reporting unit, a value is calculated and stored which indicates *how much* (or how little) of each variable exists within that reporting unit.



There are a number of steps involved in this phase. While there is no universal method of conducting this phase of the analysis, there are a set of tools in the [EEMS Support Tools\EEMS Support Tools.tbx\Reporting Unit Tools](#) toolset that can help automate much of this process (e.g., *Calculate Feature Density by Reporting Unit* & *Calculate Raster Density By Reporting Unit*).

Common metrics calculated for vector input data are as follows:

- **Points:** a count of the number of points within each reporting unit.
- **Lines:** the linear density within each reporting unit (e.g., km/km²).
- **Polygons:** The percent of each reporting unit occupied by a polygon.

Common metrics calculated for raster input data are as follows:

- Percent coverage within each reporting unit
- Statistical values (e.g., mean, max, min) within each reporting unit.

The general procedure for conducting this phase of the analysis is as follows:

1. Make a copy of the empty reporting units dataset created in phase 2.
2. Perform the necessary operations to calculate the desired metric within each reporting unit (either using the reporting units tools in the [EEMS Support Tools](#) or the standard suite of tools in ArcToolbox).
3. Once you have the desired metric (e.g., road density) stored in the copy of the reporting units dataset, join this dataset back to the master reporting units dataset based on the OBJECTID field.
4. Repeat steps 2 and 3 for each of your input datasets.

PHASE 4 (BUILDING AND RUNNING AN EEMS MODEL):

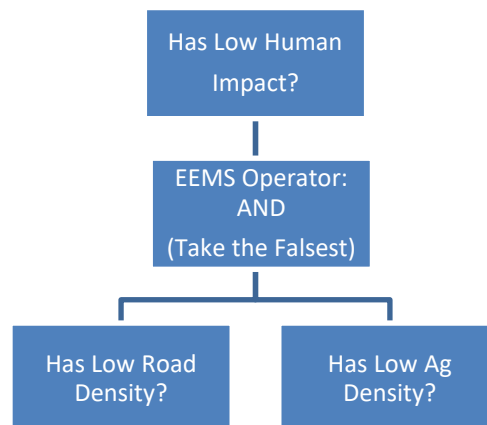
Phase 4 of the EEMS modeling process involves constructing your logic model according to the logical relationships outlined in your conceptual model diagram (figure 1-1) using the fuzzy logic operators in the EEMS toolbox (*EEMS3.1.x_ArcGIS\EEMS3.1.x\EEMS3.1.x.tbx*). Continue on with the tutorial below to learn how to build and execute a simple EEMS model.

3.3 Tutorial:

Building and Running an EEMS model in ArcGIS

The following section will walk you through the final phase (phase 4) of the EEMS modeling process in ArcGIS. This will involve building a simple logic model using an input reporting units dataset that has already been created and prepared for you. You want to be sure that you've copied the entire EEMS3.1_ArcGIS folder to a local drive for this exercise.

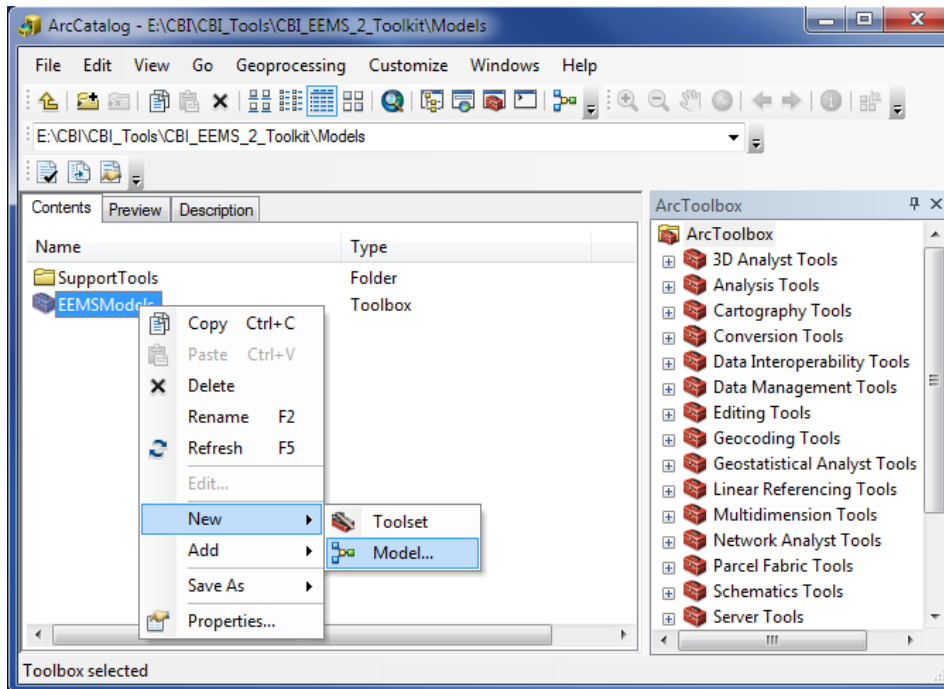
The purpose of the model you will be creating in this tutorial will simply be to determine the level of human impact across the landscape based on two input variables: road density and agricultural density. Below is our conceptual model diagram:



As described in section 1.5, the **AND** operator takes the lowest (falsest) of the inputs and is appropriate to use in situations where all of the input propositions must be true in order for the output proposition to be true. In other words, **there must be both low density and low agricultural density within a reporting unit in order for there to be a low human impact.**

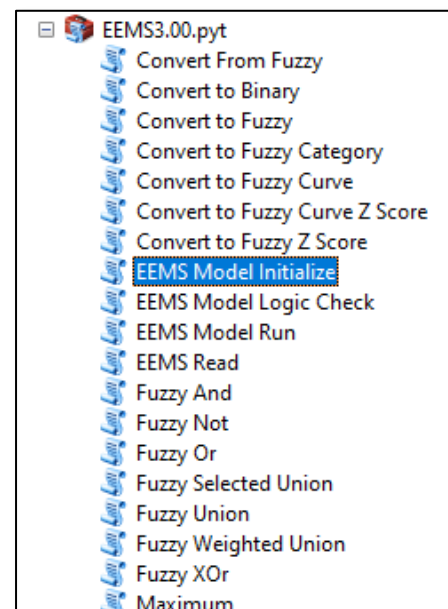
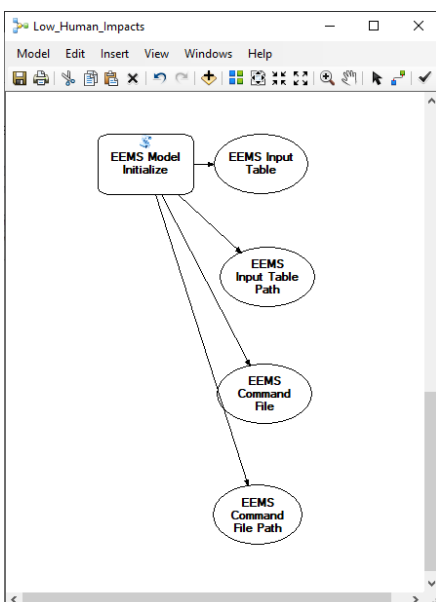
STEP 1

The first step in building your EEMS model is to create a new model within an ArcGIS Toolbox. Open up ArcCatalog and create a new toolbox in a directory on your local hard drive. Then create a model within that toolbox. This will be your EEMS Model.



STEP 2

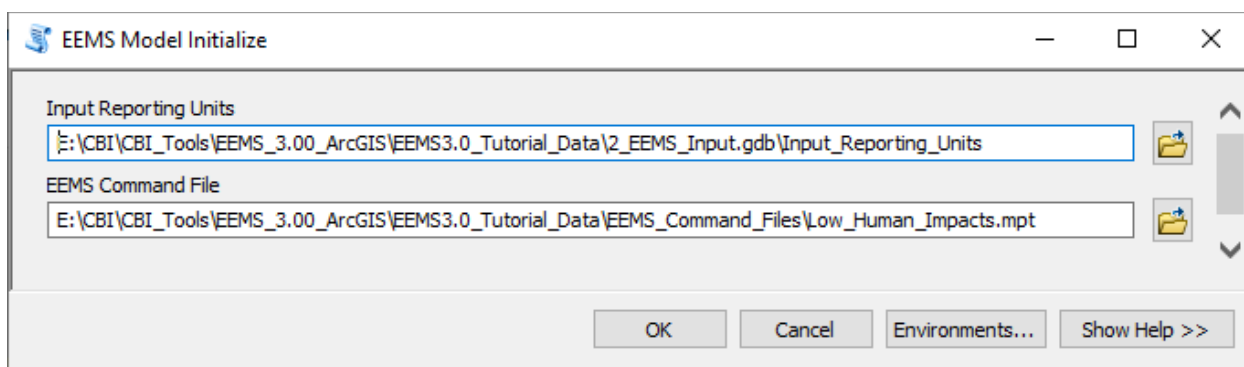
Next, bring the **EEMS Model Initialize** tool into your model. The EEMS Tools are all within the EEMS3.1Toolbox located in the `EEMS3.1.x_ArcGIS\EEMS3.1.x` directory.



The **EEMS Model Initialize** tool performs three functions:

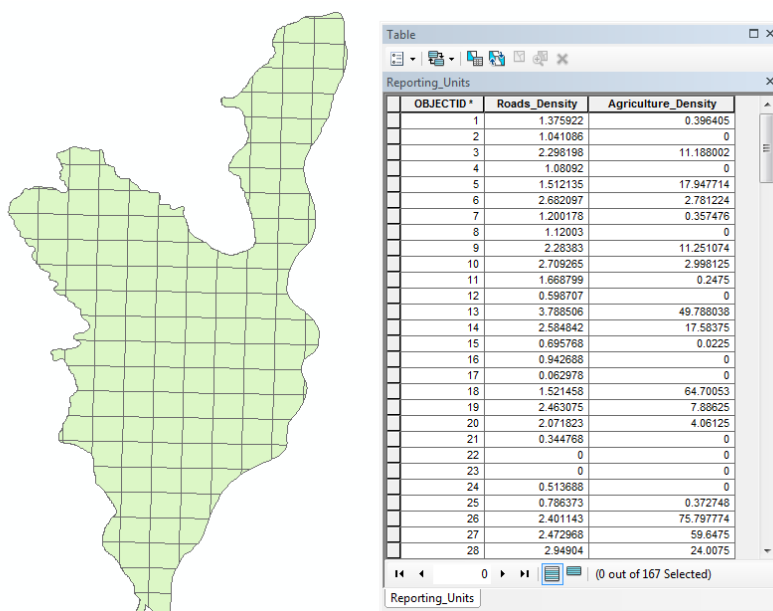
1. It specifies the name and location of the Input Reporting Units feature class, and makes the fields in this file accessible to EEMS as a table.
2. It specifies the name and location for the EEMS Command File (more on this later).
3. It stores the paths to these files as Model Builder variables with consistent names that can be used through the model (**EEMS Input Table Path**, and **EEMS Command File Path**). These variables names should not be changed.

Double click the **EEMS Model Initialize** tool and browse to the **Input Reporting Units** dataset located in EEMS3.1.x_ArcGIS\EEMS3.1.x_Tutorial_Data\3_EEMS_Input.gdb, and specify a name and output location for the **EEMS Command File**, as shown below. Note that the EEMS Command file must have a **.mpt** extension:



The input reporting units dataset has a field for **Road Density** and a field for **Ag Density**. There may be other fields in the attribute table as well, but we won't be using them in this tutorial.

INPUT REPORTING UNITS DATASET

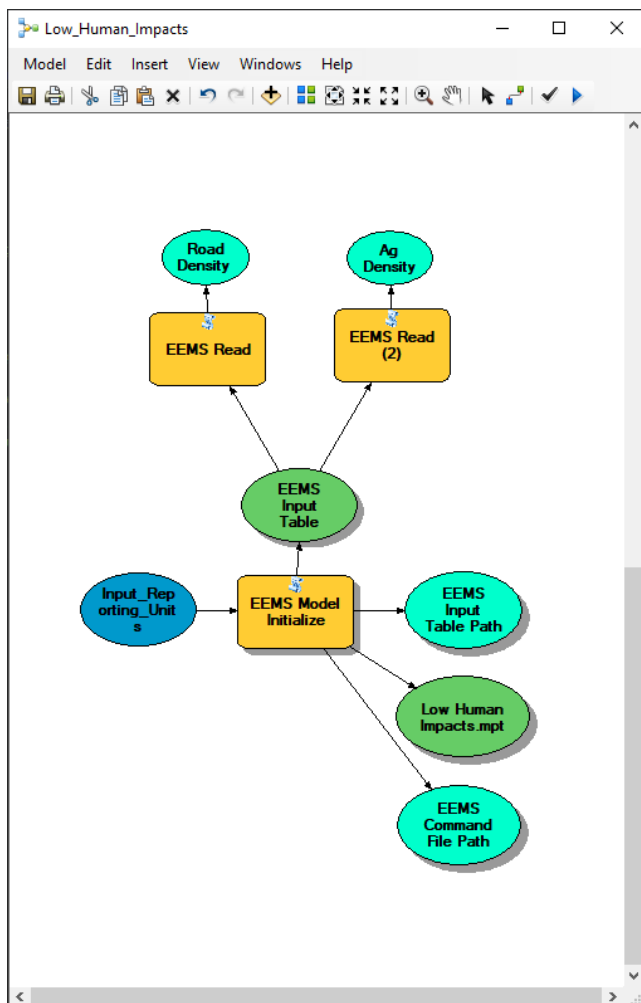


The **EEMS Command File** is a text file that will contain all the instructions to tell EEMS how to build and execute your EEMS model. Each time an EEMS tool in your model runs, an EEMS command gets written to the EEMS Command File. When all of the commands have been written to this file, EEMS (the set of Python scripts in the MPilot directory), will be able to read in and execute the lines of code in the EEMS command file to generate your results.

So, now, right click on the EEMS Model Initialize Tool in your model, and click **Run**.

STEP 4

Next, bring in two **EEMS Read** commands from the EEMS Toolbox and connect the **EEMS Input Table** to them. Double click each EEMS Read command and set the **Input Field** to “Road_Density” in one of them and “Agriculture_Density” in the other one.



At this point, you may want to organize your model as shown on the left. You should also rename the outputs from the EEMS Read commands to something more meaningful (e.g., “Road Density” & “Ag Density”). These are just labels and can have spaces, whereas the Input Field names cannot.

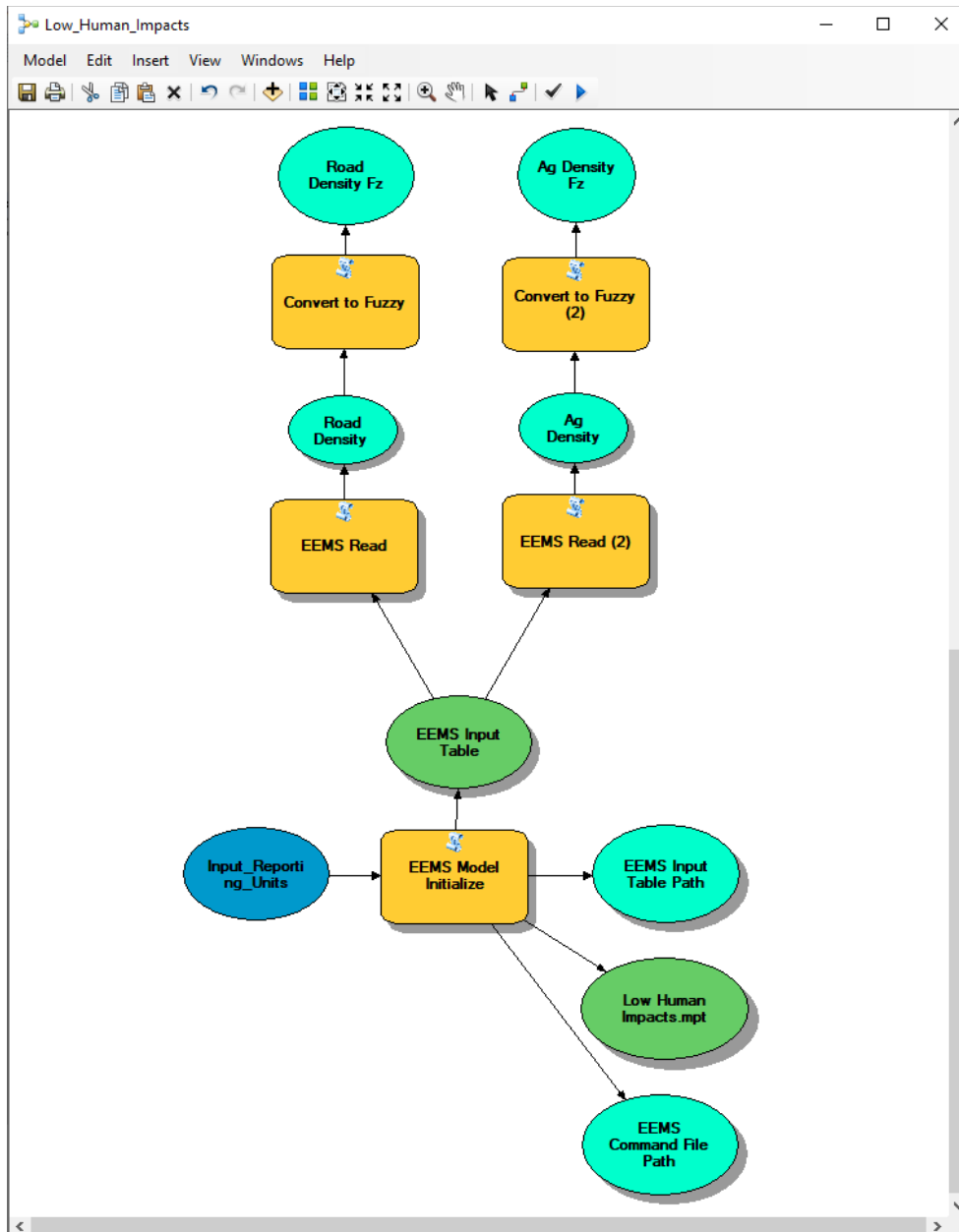
The **EEMS Read** tool is what you use to specify which of the fields in your input reporting units dataset should be read in and processed by your EEMS model.

The two bright blue bubbles are variable names. One of them stores a path to the Input Reporting Units dataset and the other one stores the path to the EEMS Command File. It is important that you don’t change these variable names as they will be used by EEMS tools throughout your model.

Click the **Run** button on your model. It is necessary to do this at this point in order to make the Road_Density and Agriculture_Density fields available for subsequent operations.

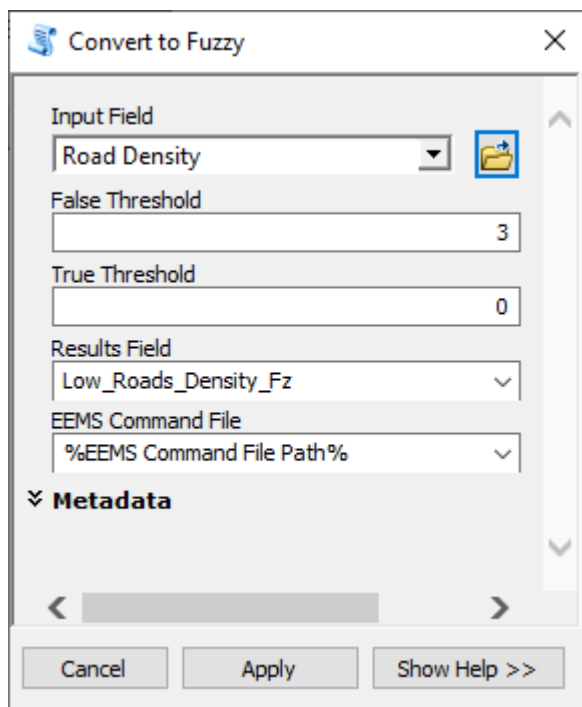
STEP 6

The next step is to convert the raw input data to fuzzy space. To do this, bring in two **Convert to Fuzzy** tools and connect them to the output from the **EEMS Read** tools as “Input Fields”. Rename the outputs from the EEMS Convert to Fuzzy commands to something more meaningful (e.g., “Road Density Fz” & “Ag Density Fz”, as shown below).



Next, double click the **EEMS Convert to Fuzzy** tools and set the true and false thresholds to the values specified in the tool dialogs below. You'll notice that the **Results Field** auto-populates with a default name with a directionality (High or Low) that reflects the values for the True and False thresholds (more on this later). You have the option of changing this field name but it's always good practice to name fuzzy fields with "Fz" at the end.

Roads



Convert to Fuzzy

Input Field: Road Density

False Threshold: 3

True Threshold: 0

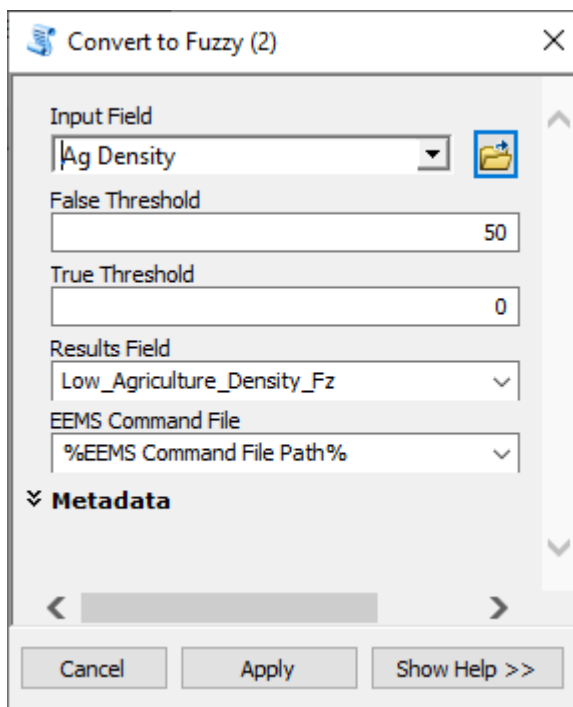
Results Field: Low_Roads_Density_Fz

EEMS Command File: %EEMS Command File Path%

Metadata

Cancel Apply Show Help >>

Agriculture



Convert to Fuzzy (2)

Input Field: Ag Density

False Threshold: 50

True Threshold: 0

Results Field: Low_Agriculture_Density_Fz

EEMS Command File: %EEMS Command File Path%

Metadata

Cancel Apply Show Help >>

As described earlier, these tools will convert the raw data in the Road Density and Ag Density Fields to Fuzzy space (values ranging from -1 to +1). By setting the thresholds specified above, we are saying that any reporting units that have a road density greater than 3 km² will exceed our false threshold, and consequently the resulting fuzzy value will be -1. A fuzzy value of -1 indicates that our proposition (Low Road Density) is totally false at that location.

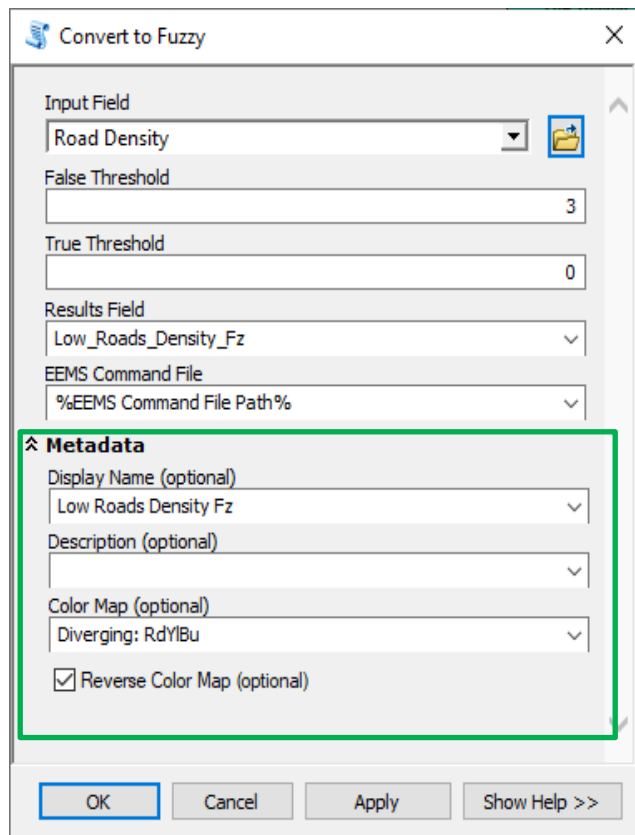
Likewise, any reporting units having an ag density greater than 50% will exceed our false threshold and the resulting fuzzy value of -1 will indicate that our proposition (Low Ag Density) is totally false.

Note that in both cases, there can't be any ag development or any roads present in order for our propositions to be totally true (i.e., our True Thresholds are both set at zero).

Input values that fall between the true and false thresholds will result in a fuzzy value somewhere between -1 and +1, meaning that the proposition is neither completely true nor completely false. But a fuzzy value of, say, 0.8, would indicate that our proposition is more true than false. The ability to produce results in the gray area in between a binary black and white classification is one of the characteristics of fuzzy logic that makes it so powerful. Another, as you will see later on, is that we can combine data with different units (km² and %) in a meaningful way.

Click the **Run** button on your model. Again, it is necessary to do this at this point in order to make these fields available for subsequent operations.

Metadata



You may have noticed a set of input options under the **Metadata** header in the tools described above (as well as in other tools in the EEMS Toolbox). These fields can be used to add auxiliary information to each node in your EEMS model. This information will be written to the EEMS Command file, where it can be accessed and used by [EEMS Online](#) and potentially other applications in the future.

Currently, there are three options available:

1. Display Name: Used to provide a human readable alias for the field name. For example, field names in ArcGIS must not have spaces, but the Display Name can. The Display Name is what will appear in the interactive model diagram in EEMS Online.

2. Description: Used to provide a brief description about the data stored in the Results Field. When a description is added, EEMS Online will add an “i” icon to the node which will display the description when a user hovers over it.

3. Color Map: Used by EEMS Online to control the colors used to render the data in the Results Field (plus an additional checkbox that can be used to reverse the selected color ramp). Visit the URL below for a visual representation of the color map options:

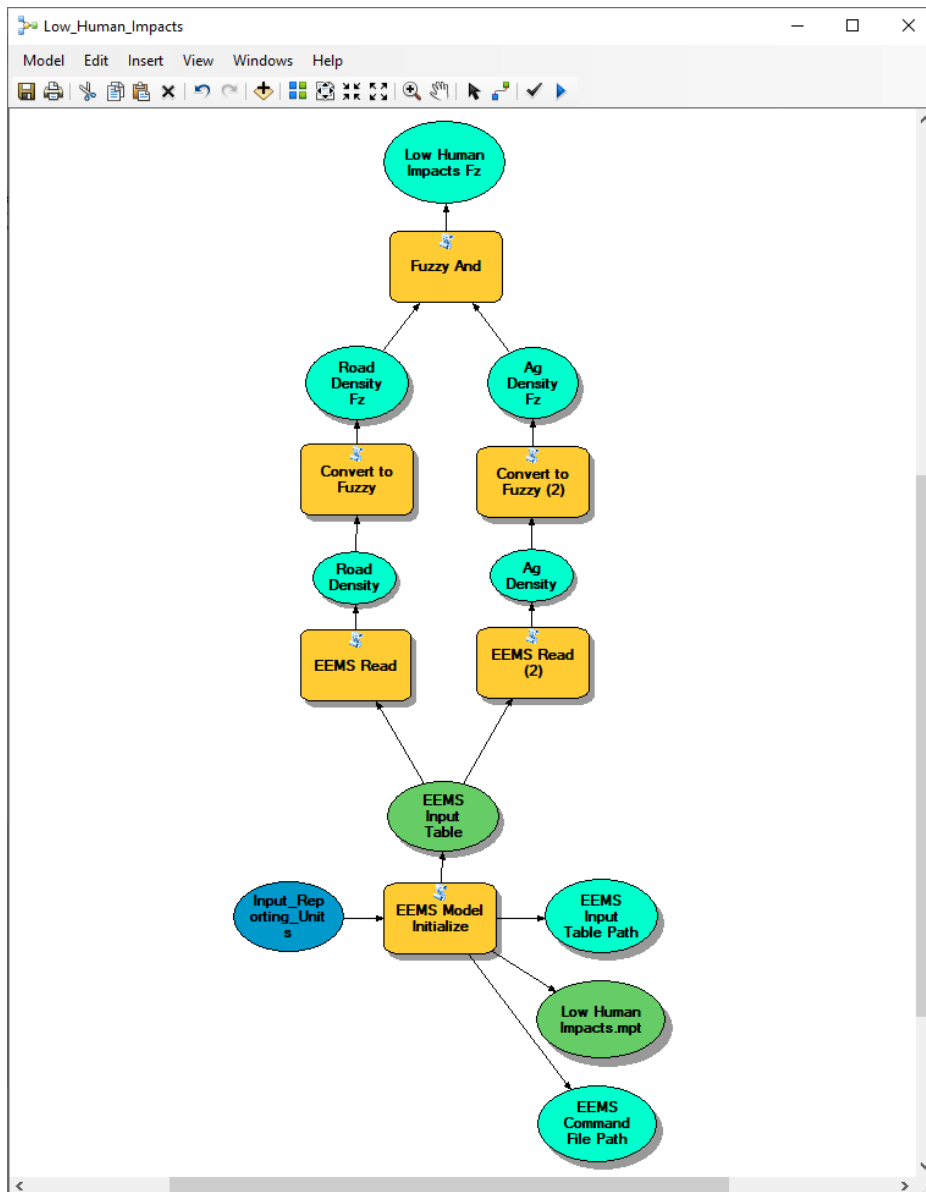
https://matplotlib.org/3.1.1/gallery/color/colormap_reference.html

STEP 7

Next, let's bring in the **Fuzzy And** tool and connect the two fuzzy nodes to it as "Input Fields", as shown below.

Double click the **EEMS And** tool and change the Results Field to "Low_Human_Impacts_Fz".

Rename the output from the **EEMS And** command to something more meaningful (e.g., "Low Human Impacts Fz").



Push **Run** on your model.

Note that up until this point we haven't created any new data and EEMS hasn't done any processing of the input fields. The only thing that has happened up to this point is that each time you pushed the run

button on your model, a line (or lines) of code have been written to the **EEMS Command File**. Let's open up this file and take a look inside. It can be opened with any text editor (e.g., Notepad or VIM).

Navigate to your EEMS Command File, which, if you followed the steps above, should be a file called *Low_Human_Impacts.eem* located in your EEMS3.1_Tutorial_Data\EEMS_Command_Files directory. In it you should see the following lines of code:

```
Roads_Density = EEMSRead(  
    InFileName = <your workspace path>\Input_Reporting_Units,  
    InFieldName = Roads_Density,  
    DataType = Float,  
    Metadata = [  
        DisplayName: Roads&nbsp;Density,  
        ColorMap: binary  
    ]  
)  
Low_Roads_Density_Fz = CvtToFuzzy(  
    InFieldName = Roads_Density,  
    FalseThreshold = 3.0,  
    TrueThreshold = 0.0,  
    Metadata = [  
        DisplayName: Low&nbsp;Roads&nbsp;Density&nbsp;Fz,  
        ColorMap: RdYlBu_r  
    ]  
)  
Agriculture_Density = EEMSRead(  
    InFileName = <your workspace path>\Input_Reporting_Units,  
    InFieldName = Agriculture_Density,  
    DataType = Float,  
    Metadata = [  
        DisplayName: Agriculture&nbsp;Density,  
        ColorMap: binary  
    ]  
)  
Low_Agriculture_Density_Fz = CvtToFuzzy(  
    InFieldName = Agriculture_Density,  
    FalseThreshold = 50.0,  
    TrueThreshold = 0.0,  
    Metadata = [  
        DisplayName: Low&nbsp;Agriculture&nbsp;Density&nbsp;Fz,  
        ColorMap: RdYlBu_r  
    ]  
)  
Low_Human_Impacts_Fz = FuzzyAnd(  
    InFieldNames = [Low_Roads_Density_Fz, Low_Agriculture_Density_Fz],  
    Metadata = [  
        DisplayName: Low&nbsp;Human&nbsp;Impacts&nbsp;Fz,  
        ColorMap: RdYlBu_r  
    ]  
)
```

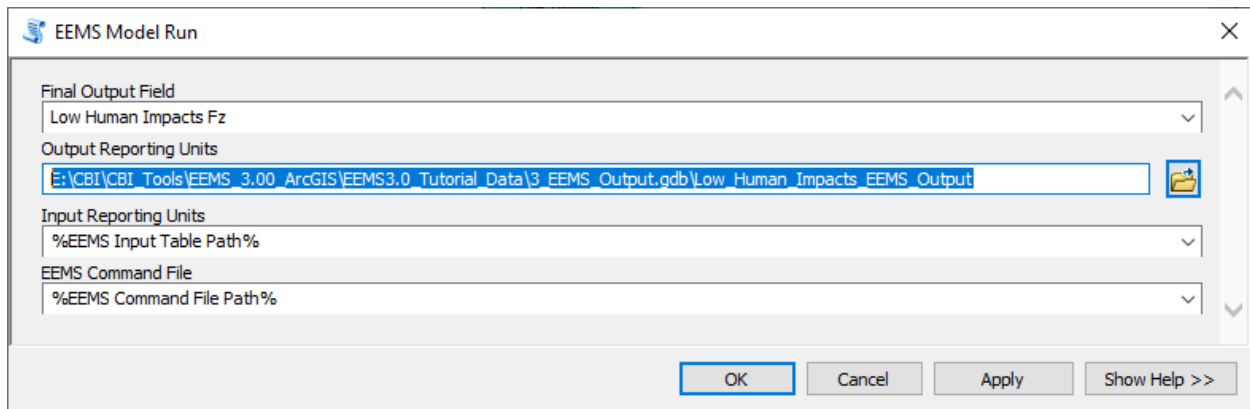
These are the commands that will instruct EEMS to read in the input data in the two input fields and generate new output data in the three fuzzy fields.

If you see more than the five commands listed above, that suggests that you ran one of the tools more than once. Duplicate commands can cause EEMS to fail. If this is the case, close your EEMS Command File and run your entire model again (Model -> Run Entire Model). By rerunning your tool from the beginning, the EEMS Model Initialize tool will re-run first, which will clear out the EEMS command file and all of the subsequent commands will be rewritten.

Close the EEMS Command File if you haven't already done so.

STEP 8

Now that we've constructed our model, bring the **EEMS Model Run** tool in and connect the Top Node (Low Human Impacts Fz) to it as the "Final Output Field". Double click the tool, and specify a path for the Output Reporting Units (you can send it to the **3_EEMS_Output.gdb** geodatabase in the Tutorial Data workspace. Name it **Low_Human_Impacts_EEMS_Output**):



The screenshot shows the 'EEMS Model Run' dialog box with the following settings:

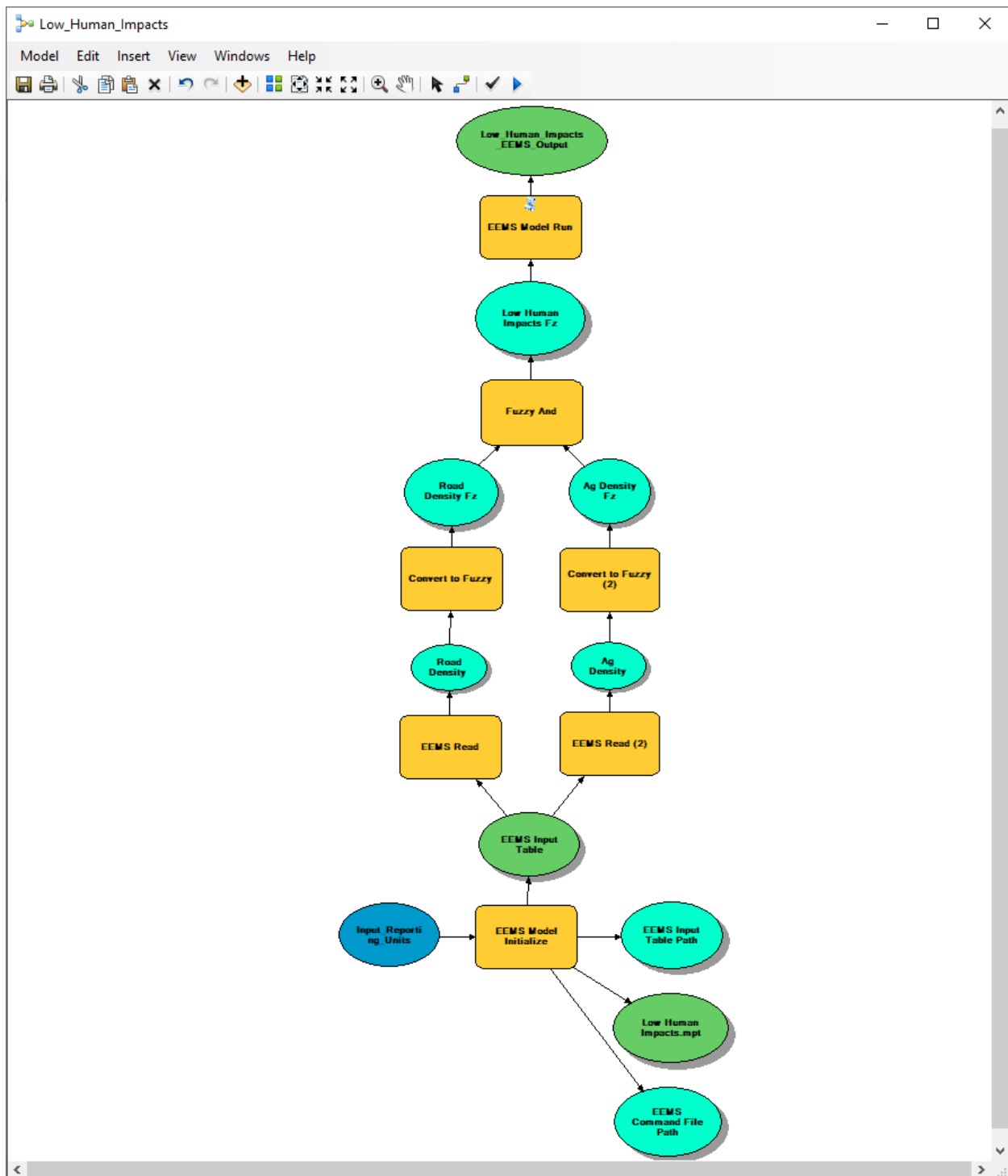
- Final Output Field:** Low Human Impacts Fz
- Output Reporting Units:** E:\CBI\CBI_Tools\EEMS_3.00_ArcGIS\EEMS3.0_Tutorial_Data\3_EEMS_Output.gdb\Low_Human_Impacts_EEMS_Output
- Input Reporting Units:** %EEMS Input Table Path%
- EEMS Command File:** %EEMS Command File Path%

The 'OK' button is highlighted in blue.

Click **Ok**.

The **EEMS Model Run** tool will pass the EEMS Command File to EEMS, along with the path to the input reporting units, and EEMS will execute the lines of code in the command file, writing the output values to the output reporting units.

At this point your model is complete. Please confirm that it appears as shown on the following page.



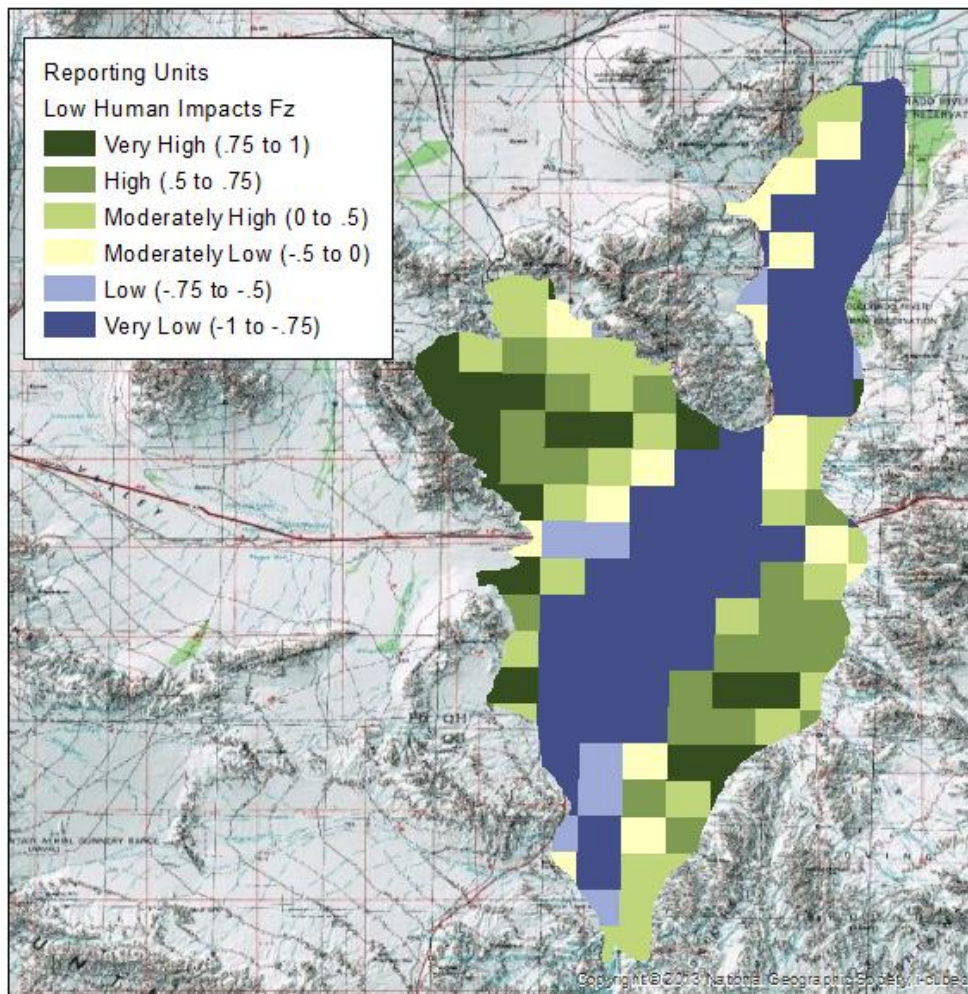
STEP 9

Finally, run your model by clicking the **Run** button on the Model Builder interface. Final output values will be stored in the attribute table of the feature class you specified as the Output File Name.

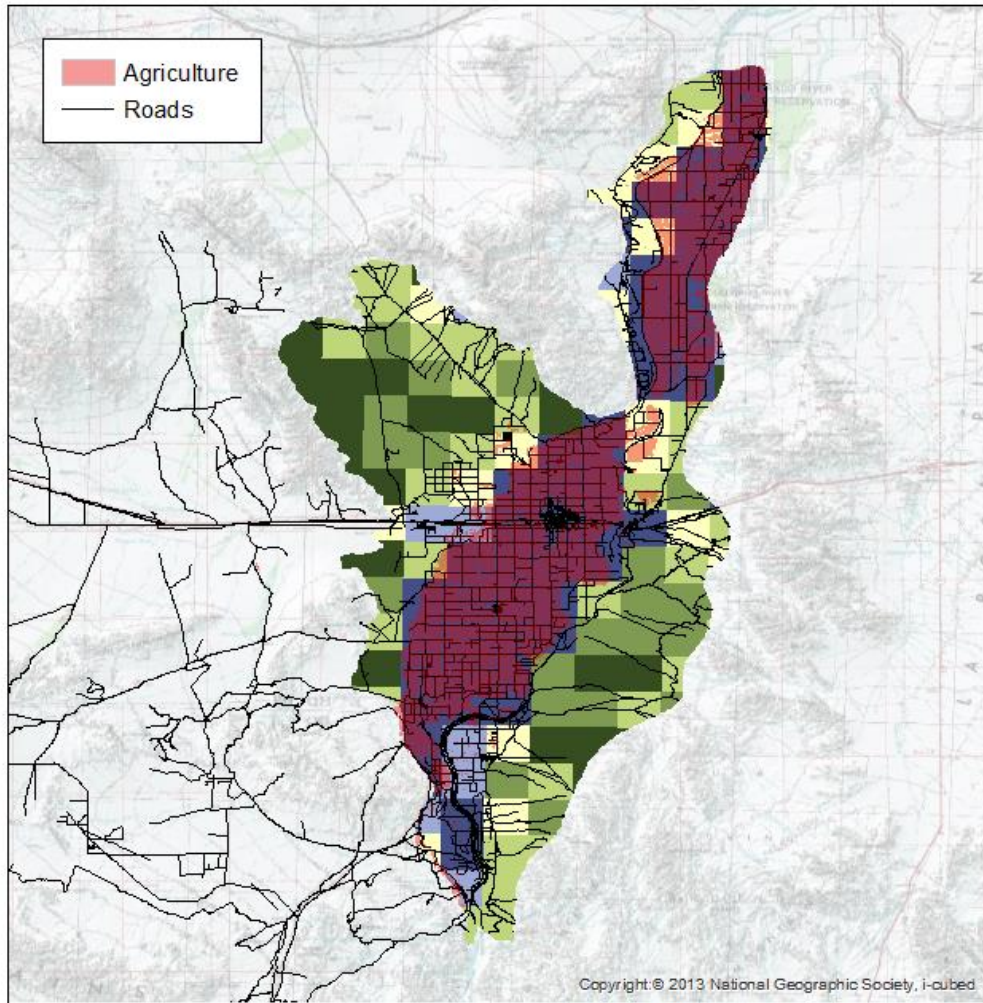
Bring your output into ArcMap and observe the results. Use the lyr file located in the EEMS3.1.x_Tutorial_Data\Layer_Files directory to symbolize your output using the same class breaks and color scheme used in the solution below. Do your results match?

Blue areas are places with Low or Very Low fuzzy values. These are areas that have been most affected by roads and agricultural development. In these areas, our **Low Human Impacts** proposition is closer to false than to true, and in some reporting units our proposition is totally false (-1).

Greener areas are those reporting units in which our proposition is closer to true than false. And in some areas, our proposition is totally true (+1). Since we set our **True Threshold** at zero, these will only occur in areas where there are no roads and no agricultural development.



The roads and agriculture data used to calculate the densities are in the \EEMS3.1.x_Tutorial_Data\1_RAW_Input.gdb geodatabase. If you bring them into your map, you can see that there are indeed areas in the Northwest side of the study area that don't have any roads or agricultural development. The output values in the Low_Human_Impacts_Fz field should be +1 in these areas.



This concludes the tutorial. At this point, you are encouraged to experiment with different thresholds and operators in order to observe the effects on the final output values. You are also encouraged to experiment with your own input data and study area boundary.

References Cited

- Jensen, M., K. Reynolds, U. Langner, and M. Hart. 2009. Application of logic and decision models in sustainable ecosystem management. 2009. Proceedings of the 42nd Hawaii International Conference on Systems Sciences. Waikoloa, Hawaii. 5-8 January 2009.
- Reynolds, K.M. 1999. NetWeaver for EMDS version 2.0 user guide: A knowledge base development system. U.S. Forest Service, General Technical Report PNW-GTR-471, U.S. Forest Service, Pacific Northwest Research Station, Portland, Oregon.
- Sheehan, T. and Gough, M. (2016) A platform-independent fuzzy logic modeling framework for environmental decision support. *Ecological Informatics*. 34:92-101