5

Implementación de los modelos cartográficos

Los modelos cartográficos se implementan en el *software* ArcGIS versión 10.2. Para la aplicación de la metodología descrita en el capítulo anterior se hará uso del lenguaje de programación Python (https://www.python.org/). Este lenguaje de programación es un ejemplo de un lenguaje de alto nivel y entre las ventajas de su uso están la facilidad para programar, la portabilidad (Downey, 2012); además es altamente escalable, es incorporable a ArcGIS, es estable y maduro y cuenta con una gran comunidad de usuarios (Environmental Systems Research Institute Inc., 2016).

En este capítulo se muestra un ejemplo de las funciones de análisis espacial presentadas y se describirán los *scripts* realizados para la automatización de los modelos cartográficos. Para las funciones de análisis, se hará referencia a su ubicación en el módulo de *Arctoolbox*, debido a que esta es la forma en la que se pueden integrar a los *scripts* para la implementación de los modelos.

5.1 Herramientas de análisis espacial en ArcGIS

A continuación, se presentan ejemplos de aplicación de las herramientas de análisis espacial en el *software* ArcGIS para el caso de los modelos descritos en el capítulo anterior.

5.1.1 Cálculo de atributos, expresión condicional

Para la capa *Amenaza Incendio*, se desea definir un valor numérico, de la viabilidad para la actividad minera, para cada nivel de amenaza. En el campo "amenaza", la capa tiene información cuantitativa que clasifica las amenazas en los niveles *alto, medio alto, medio, medio bajo* y *bajo*. La viabilidad que se asigna a cada nivel es la que se consigna en la Tabla 11.

Para convertir esta clasificación cualitativa en cuantitativa, se crea un nuevo campo de tipo numérico (*Float*) denominado *Viabilidad*. Posteriormente, con la ayuda de la herramienta para el cálculo de atributos (ubicada en la ruta: *ArcToolBox –Data Management Tools – Field – CalculateField*), se estiman los valores con la función que se muestra en la Figura 27.

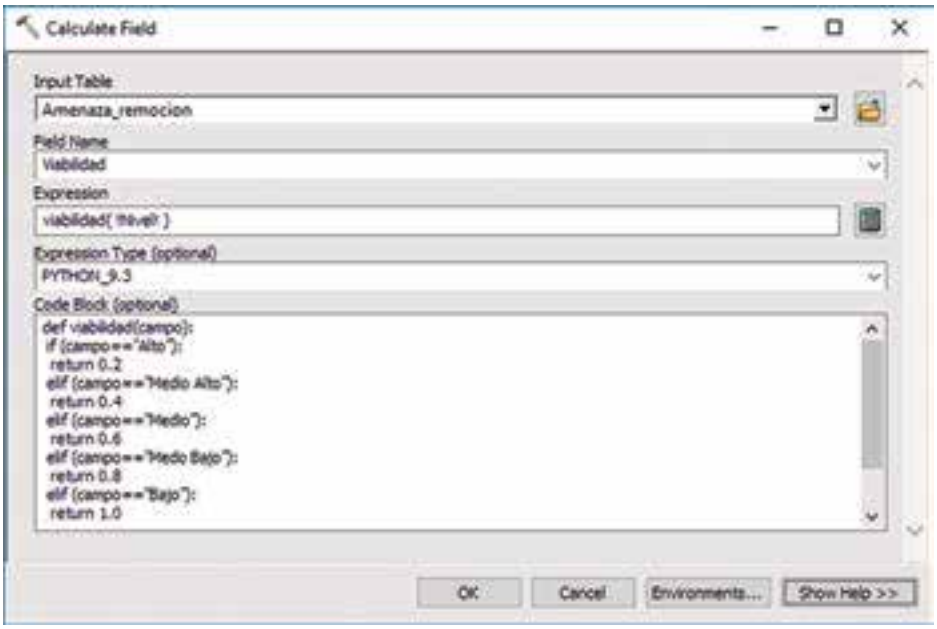

Figura 27. Interface herramienta *Calculate Field*, expresión condicional.

En la herramienta se introduce la tabla de atributos, se elige el campo sobre el cual se hará el cálculo, se escribe la expresión (que en este caso es la función viabilidad con el parámetro nivel de amenaza) y el código de la función en lenguaje Python. Como resultado se obtiene un valor numérico dentro del intervalo cero a uno, correspondiente al nivel de viabilidad. Dicho valor será utilizado como atributo principal en la creación de datos ráster, que se presentan en el siguiente capítulo.

5.1.2 Transformación de polígono a ráster

Una vez asignados los valores de viabilidad en el *Feature Class* de amenazas, se efectúa la transformación de tipo polígono a ráster con el apoyo de la herramienta de conversión ubicada en: *ArcToolBox - ConversionTools – To Raster – Polygon to Raster*. Se introduce la capa con geometría polígono, el campo que contiene los valores que serán asignados a cada uno de los píxeles, el archivo ráster de salida, el tipo de asignación de los valores a los píxeles, el campo prioritario y el tamaño de píxel (Figura 28).

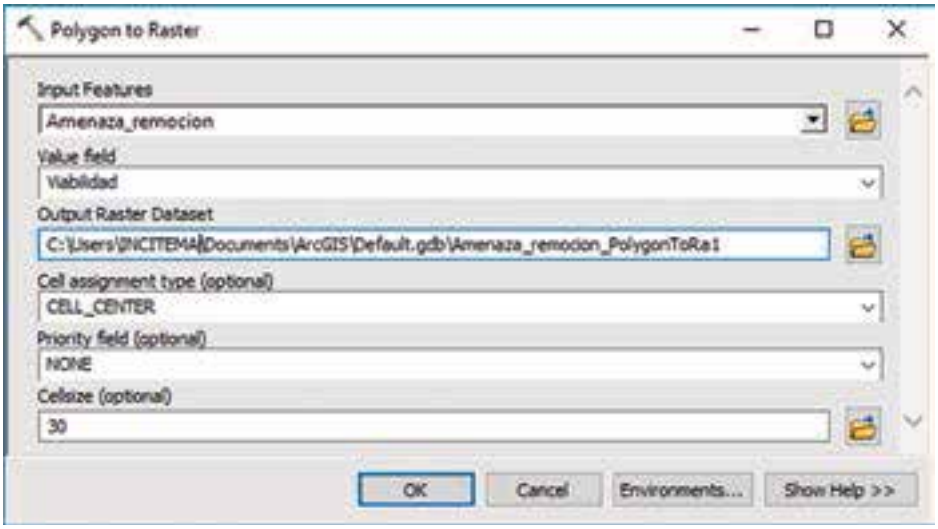

Figura 28. Interface de la herramienta *Polygon to Raster*

En la herramienta, la opción *Valor de campo* permite definir el valor que será asignado a cada píxel y corresponde a la información almacenada en el campo "Viabilidad". Los tipos de asignación de valores a los píxeles son los siguientes:

- CELL_CENTER: el polígono que se sobrepone en el centro del píxel será el que aporte el valor para este.
- MAXIMUM_AREA: el polígono con mayor área dentro del píxel será el que aporte del valor.
- MAXIMUM_COMBINED_AREA: cuando hay más de un polígono con el mismo valor, se combinan las áreas de estos. La mayor área después de la combinación será la que aporte el valor.

5.1.3 Distancia euclidiana

El cálculo de la distancia euclidiana puede ser aplicado en la delimitación del margen de protección de las corrientes hídricas. La herramienta *Euclidean Distance* en ArcGIS está disponible en: *ArcToolBox –Spatial Analyst Tools – Distance*. Se introducen a la capa sobre la cual se calculará la distancia euclidiana, el archivo ráster de salida, la distancia máxima del cálculo (más allá de esta distancia, el ráster tendrá valores nulos en sus píxeles), el tamaño de píxel y un ráster mostrando la dirección (ver Figura 29).

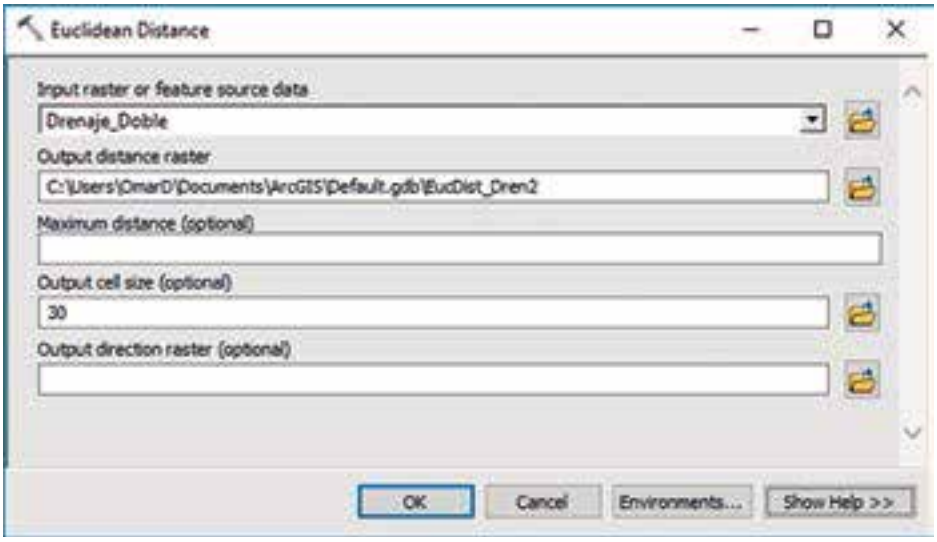

Figura 29. Interface de la herramienta *Euclidian Distance*.

5.1.4 Álgebra de mapas

Una vez determinada la distancia euclidiana de las corrientes hídricas (drenaje sencillo), se normalizan los resultados con el fin de obtener

un ráster con valores de cero a uno. La normalización se calcula estableciendo 2000 metros como la distancia máxima y utilizando la ecuación exponencial presentada en la Figura 24. La herramienta *Raster Calculator* puede ser encontrada en la ruta *ArcToolBox – Spatial Analyst Tools – Map Algebra*. La herramienta cuenta con los siguientes módulos para realizar el cálculo correspondiente: el módulo con el listado de capas que se pueden utilizar en la herramienta, el módulo de botones con caracteres numéricos y operadores matemáticos, el módulo en el que están las funciones (de tipo condicional, matemático, trigonométrico y de manejo de valores en píxeles) y el módulo para construir la expresión matemática (ver Figura 30).

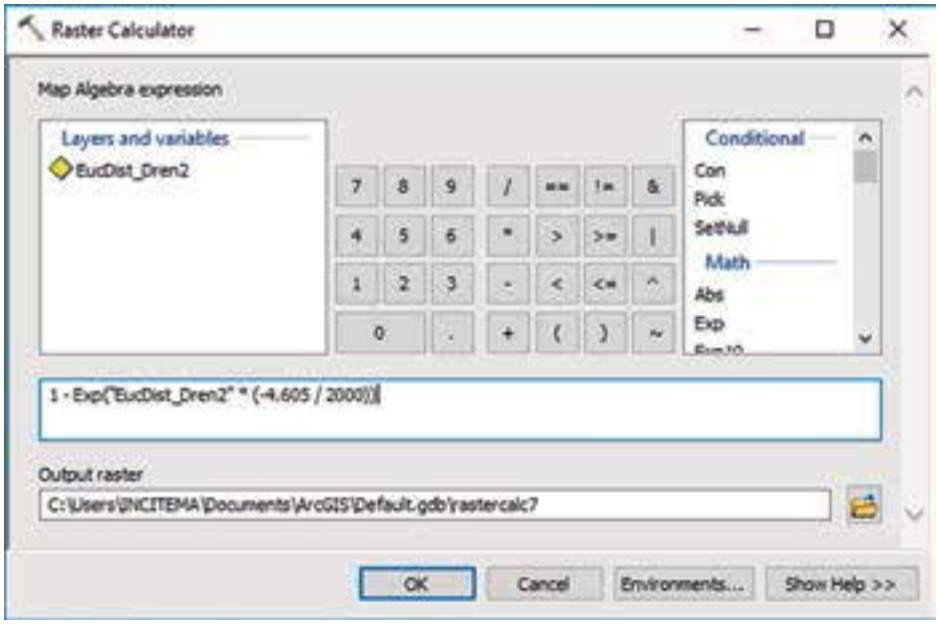

Figura 30. Interface de usuario de la herramienta *Raster Calculator*

5.1.5 Cálculo de atributos, expresión aritmética

Se desea calcular la densidad de los centros poblados para la capa de zonas urbanas a escala 1:100.000 descargadas de la página oficial del IGAC. Para ello se hace uso de la herramienta *Field Calculator* (Figura 31) disponible en: *ArcToolBox –Data Management Tools – Fields*. En el capítulo 5, se presenta la ecuación utilizada para calcular el número de puntos, la cual está incluida en un nuevo campo denominado *nu_puntos*.

Una vez determinado el número de puntos que corresponden a cada polígono que delimita las cabeceras municipales, se establece el número de habitantes que representa cada punto. Debido a que el *shapefile* originalmente descargado no cuenta con información de población, se debe buscar información espacial al respecto. Se crea un nuevo campo llamado *pob_cp* en el cual se incluirá la población proyectada a 2018 por el DANE. La información puede ser descargada en formato Excel en la página oficial del DANE siguiendo la ruta *DANE – Estadísticas por tema – Demografía y población – Proyecciones de Población – Estimación y proyección de población nacional, departamental y municipal total por área 1985-2020*.

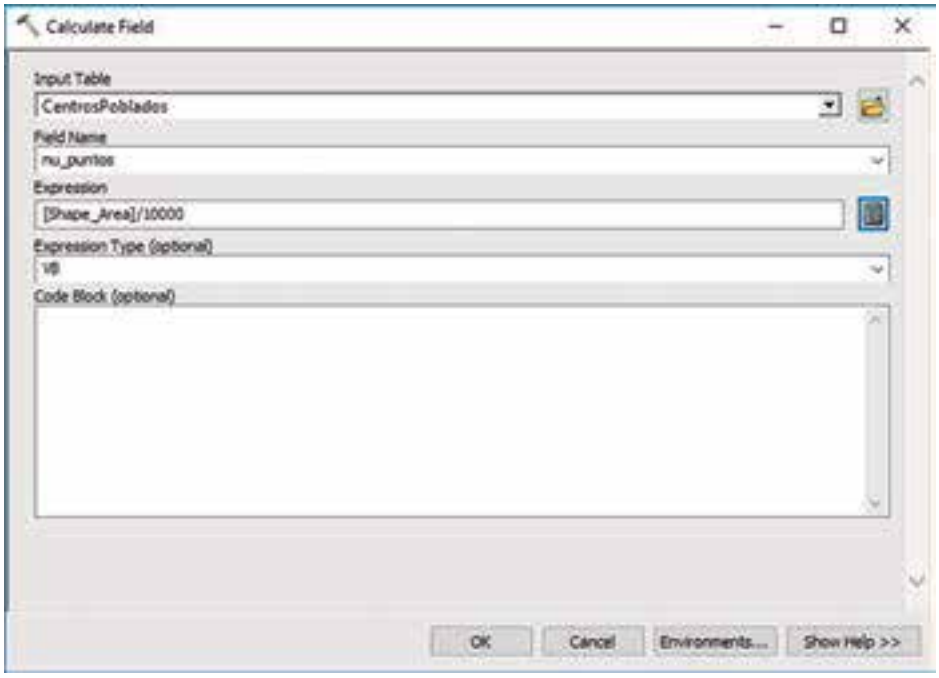

Figura 31. Interface de usuario herramienta *Calculate Field*, expresión aritmética.

El cálculo del número de habitantes por cada punto es registrado en un nuevo campo de tipo numérico denominado *pob_pto,* utilizando la ecuación:

$$Habitantes_{punto} = \frac{Población_{CentroPoblado}}{N_{puntos}} \quad (8)$$

5.1.6 Puntos aleatorios

Para generar puntos aleatorios en un área específica puede recurrirse a la herramienta *Create Random Points* (Figura 32) que se ubica en *ArcToolBox - Data Management Tools – Sampling*. Se debe ingresar la localización de la carpeta donde será creado el archivo y el nombre de salida, se adiciona la capa de zonas urbanas (cuyos polígonos servirán para delimitar las áreas en las que se crearán los puntos). El número de puntos por polígono está dado por el valor calculado en el campo *nu_puntos* calculado previamente. La separación mínima entre puntos también puede ser definida mediante un valor constante o utilizando el valor de un campo en la tabla de atributos. La herramienta también permite agrupar un determinado número de puntos (Mulitpoint Geometry).

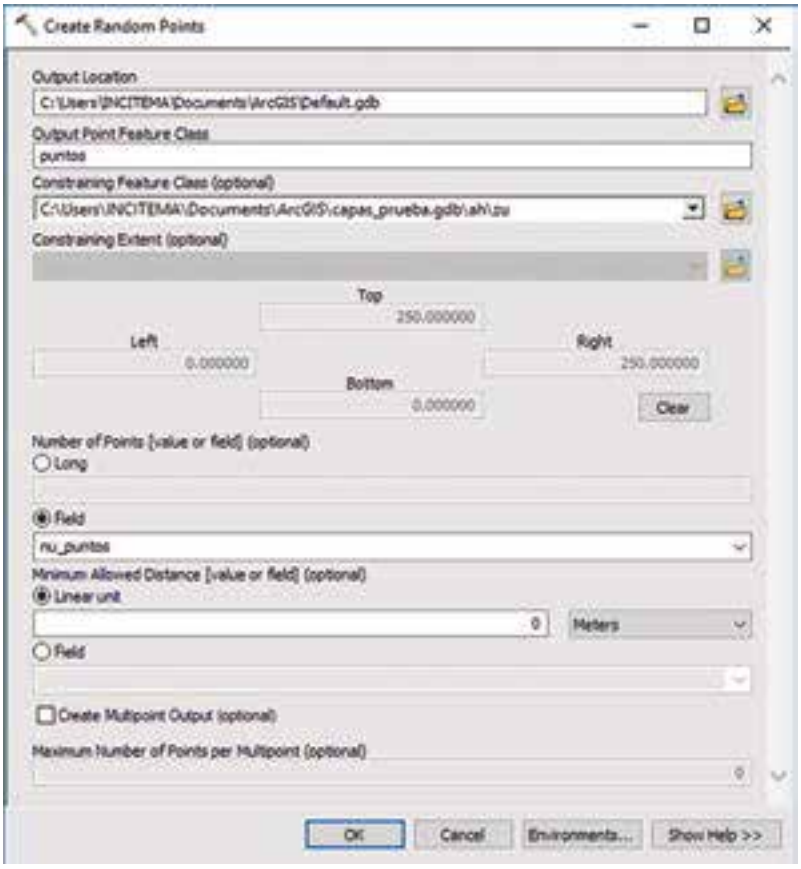

Figura 32. Interface de usuario para la herramienta *Create Random Points*.

5.1.7 Relación espacial entre tablas de atributos

Los puntos aleatorios generados en el ejemplo anterior tienen una tabla de atributos carente de información, razón por la cual se pretende relacionar la información de centros poblados a la capa de puntos con el fin de transferir la cantidad de habitantes que representa cada punto (la cual fue calculada en la tabla de atributos de las zonas urbanas). Este paso conlleva la creación de un nuevo *shapefile* que contendrá la combinación seleccionada de las tablas de atributos de los campos. La manera de relacionar la información es a través de la herramienta *Sptial Join*, disponible en *ArcToolBox – Analysis Tools – Overlay* (Figura 33).

Se introducen: la capa de puntos, la capa con los centros poblados, el archivo de salida (que será de geometría punto), la relación de atributos, los campos que se transferirán, la relación espacial (en este caso, intersección) y opcionalmente un radio de búsqueda (que incluirá puntos que estén a la distancia dada de los polígonos).

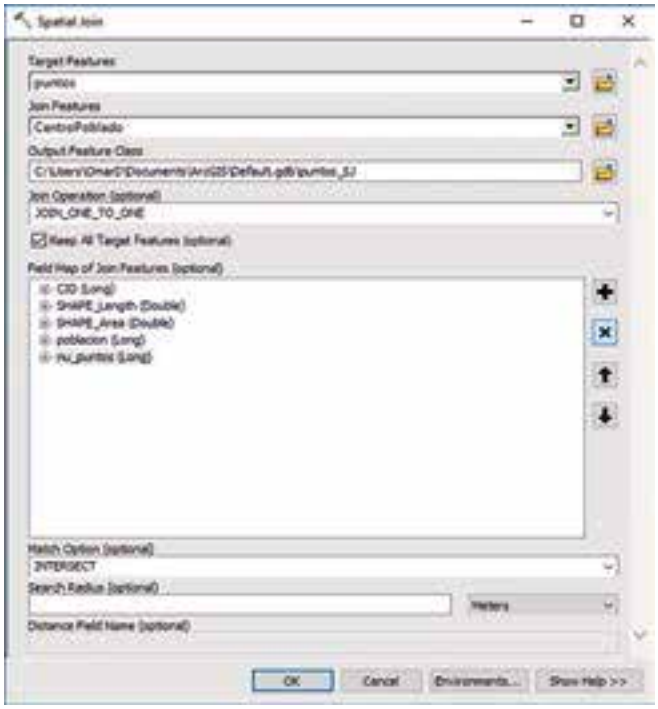

Figura 33. Interface de usuario para la herramienta *Spatial Join*.

5.1.8 Densidad Kernel

La información de población para cada uno de los puntos aleatorios generados, resultado de la relación espacial de atributos presentado en el numeral anterior, es utilizada para calcular la densidad de la población con la herramienta *Kernel Density* (Figura 34), la cual está disponible en: ArcToolBox –*Spatial Analyst Tools*.

Se introducen la capa de puntos, el campo en el que se almacena el valor de población para cada punto, el archivo de salida, el tamaño de pixel, el radio para la función Kernel y las unidades de los valores en los pixeles de salida.

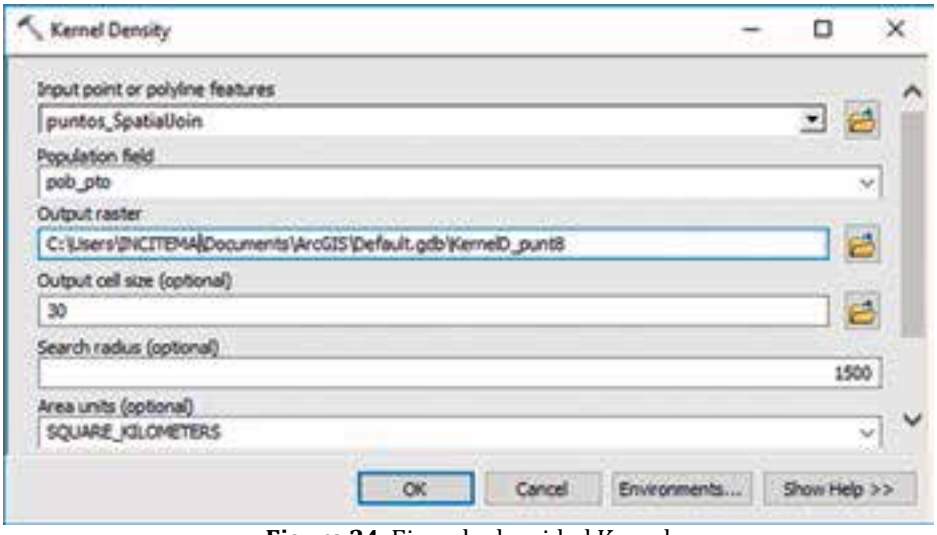

Figura 34. Ejemplo densidad Kernel

5.2 Script para el componente abiótico

Las primeras líneas del código son iguales para los cuatro *scripts* que se describirán más adelante.

```
1    import arcpy
2    arcpy.env.overwriteOutput = True
3
4    limite = arcpy.GetParameterAsText(0)
5    pixel = arcpy.GetParameterAsText (1)
6
7    arcpy.env.extent = limite
8
```

› Línea 1: importación del módulo arcpy, el cual permite el acceso a las herramientas y funciones del programa ArcGIS.

› Línea 2: configuración de entorno de procesamiento para permitir que se sobrescriba automáticamente un dato existente.
› Línea 4: asignación de las coordenadas norte y este, máximas y mínimas, a la variable *límite*, que definen el área en la cual se realizará el análisis. Es un dato de entrada.
› Línea 5: asignación a la variable *píxel* del tamaño de píxel necesario para la generación de datos tipo ráster. Es un dato de entrada.
› Línea 7: configuración de entorno de procesamiento para delimitar los análisis espaciales al área delimitada con las coordenadas almacenadas en la variable *límite.*

5.2.1 Asignación de viabilidad y transformación vector a ráster

Como se describió en el modelo cartográfico relacionado con los componentes abióticos, uno de los procesos es el cálculo de la viabilidad teniendo en cuenta una escala ordinal de la Tabla 11. A continuación, se define la función viabilidad para lograr este objetivo.

```
9     def viabilidad(capa):
10        expresion = """def reclass(Valor):
11           if (Valor=='Alto'):
12              return 0.2
13           elif (Valor=='Medio Alto'):
14    return 0.4
15           elif (Valor=='Medio'):
16              return 0.6
17    elif (Valor=='Medio Bajo'):
18              return 0.8
19    elif (Valor=='Bajo'):
20              return 1.0
21           else:
22              return 0"""
23        capaViab=arcpy.CreateScratchName("Viab_capa",\
24                                         workspace=arcpy.env.scratchGDB)
25        arcpy.AddField_management(capa, "Viabilidad", "FLOAT", "", "", "",\
26    "", "NULLABLE", "NON_REQUIRED", "")
27        arcpy.CalculateField_management(capa, "Viabilidad",\
28    "reclass(!Nivel!)", "PYTHON", expresion)
29        arcpy.PolygonToRaster_conversion(capa, "Viabilidad", capaViab,\
30    "CELL_CENTER", "NONE", pixel)
31        return capaViab
```

› Línea 9: definición de la función *viabilidad*, en la cual el parámetro es una capa en cuya tabla de atributos se encuentra un campo con nombre *Nivel*.
› Líneas 10 a 22: asignación de la función *reclass* a la variable *expresion*. Esta función identifica los valores de la escala ordinal en el campo *Nivel* y regresa como resultado el valor de viabilidad correspondiente.
› Línea 23: asignación a la variable *capaViab* de una ruta creada para el acceso exclusivo del dato especificado. La ubicación del dato

será una *File Geodatabase* de nombre *Scratch* que se creará en el directorio *Home*. Este dato es un resultado intermedio generado durante el procesamiento.

› Línea 25: creación de un campo de nombre *Viabilidad* en la tabla de atributos de la capa de entrada a la función.

› Línea 27: cálculo del campo *Viabilidad* utilizando la función *reclass* almacenada en la variable *expresion*.

› Línea 29: transformación de la capa de entrada de tipo vector polígono a tipo ráster. El valor de cada píxel en el ráster será el almacenado en el campo *Viabilidad*. El tamaño de píxel es el definido por el usuario y asignado a la variable *píxel* en la línea 5.

› Línea 27: la función regresa la capa ráster resultado del paso anterior.

5.2.2 Procesamiento de capas asociadas con el recurso hídrico superficial

En esta parte del *script* se obtiene el ráster que sobrepone las capas relacionadas con el componente del recurso hídrico superficial. Para cada capa se calcula la distancia euclidiana y se normaliza utilizando una función exponencial.

```
32  rhs = arcpy.GetParameterAsText(2)
33  d_rhs = int(arcpy.GetParameterAsText(3))
34
35  rhsMin = arcpy.CreateScratchName("Min_rhs",\
36   workspace=arcpy.env.scratchGDB)
37  rhsED = arcpy.CreateScratchName("ED_rhs", workspace=arcpy.env.
38  scratchGDB)
39
40  arcpy.gp.CreateConstantRaster_sa(rhsMin, 1, "INTEGER", pixel, "")
41  rhsMin1 = arcpy.sa.Raster(rhsMin)
42
43  rhsLista = rhs.split(";")
44
45  for rhsTipo in rhsLista:
46      count1 = int(arcpy.GetCount_management(rhsTipo).getOutput(0))
47      rhsNorm = arcpy.CreateScratchName("Norm_rhs",\
48  workspace=arcpy.env.scratchGDB)
49      rhsExpo = arcpy.CreateScratchName("Expo_rhs",\
50   workspace=arcpy.env.scratchGDB)
51      rhsE = arcpy.CreateScratchName("E_rhs",\
52   workspace=arcpy.env.scratchGDB)
53      if count1 > 0:
54          arcpy.gp.EucDistance_sa(rhsTipo, rhsED, "", pixel, "")
55          expo = arcpy.sa.Raster(rhsED)*(-4.605/d_rhs)
56  arcpy.gp.Exp_sa(expo,rhsExpo)
57  rhsNorm = 1-arcpy.sa.Raster(rhsExpo)
58      else:
59          arcpy.gp.CreateConstantRaster_sa(rhsNorm, 1, "INTEGER",\
60   pixel, "")
61      arcpy.gp.CellStatistics_sa([rhsMin1, rhsNorm], rhsE,\
62   "MINIMUM", "DATA")
63      rhsMin1 = arcpy.sa.Raster(rhsE)
```

- Línea 32: asignación a la variable *rhs* de la lista de capas que hacen parte de los recursos hídricos superficiales. Es un dato de entrada.
- Línea 33: asignación a la variable *d_rhs* de distancia máxima para ser utilizada en la normalización de las distancias calculadas desde cada elemento de la capa. Es un dato de entrada.
- Línea 35: asignación a la variable *rhsMin* de una ruta para el acceso al dato *Min_rhs* en la geodatabase Scratch. Este dato es un resultado intermedio durante el procesamiento.
- Línea 37: asignación a la variable *rhsED* de una ruta para el acceso al dato *ED_rhs* en la geodatabase Scratch. Este dato es un resultado intermedio durante el procesamiento.
- Línea 39: creación de una capa ráster con valores de uno para todos sus píxeles. La capa se almacena con el nombre y en la ruta de la variable *rhsMin*.
- Línea 40: almacena en la variable *rhsMin1* la capa *rhsMin* como objeto ráster para ser utilizado directamente en Python en una expresión algebraica.
- Línea 42: separa como elementos independientes en una lista, cada una de las capas ingresadas en la variable *rhs*.
- Líneas 44 a 63: ciclo en el cual se calculan las distancias euclidianas para cada una de las capas en la categoría recurso hídrico superficial, almacenadas en la variable *rhsLista*.
- Línea 44: para cada una de las capas dentro de la variable *rhsLista*.
- Línea 45: almacena en la variable *count1* el número de elementos que hacen parte de la capa. Solamente se contabilizan los elementos dentro de la extensión definida en la línea 7.
- Línea 46: asignación a la variable *rhsNorm* de una ruta para el acceso al dato *Norm_rhs* en la geodatabase Scratch. Este dato es un resultado intermedio durante el procesamiento.
- Línea 48: asignación a la variable *rhsExpo* de una ruta para el acceso al dato *Expo_rhs* en la geodatabase Scratch. Este dato es un resultado intermedio durante el procesamiento.
- Línea 50: asignación a la variable *rhsE* de una ruta para el acceso al dato *E_rhs* en la geodatabase Scratch. Este dato es un resultado intermedio durante el procesamiento.
- Línea 52: evalúa si la variable *count1* es mayor que cero. Es decir, evalúa si la capa correspondiente tiene elementos dentro de la extensión de análisis previamente definida.
- Línea 53: cálculo de la distancia euclidiana para la capa correspondiente. El resultado se almacena en la ruta y con el nombre de la variable *rhsED*.

› Línea 54: se asigna a la variable *expo* el resultado de aplicar las operaciones del exponente de la ecuación exponencial presentada en la Figura 17.
› Línea 55: se eleva la constante matemática *e* al valor almacenado en la variable *expo*.
› Línea 56: en la variable *rhsNorm* se almacena el cálculo necesario para completar las operaciones de la ecuación exponencial.
› Línea 57: se evalúa si no se cumple la condición de la línea 52.
› Línea 58: se crea un ráster en el cual todos los valores de sus píxeles son uno. Se almacena en la variable *rhsNorm*. En este caso no hay elementos en la extensión de análisis, por lo tanto es viable la actividad minera.
› Línea 60: de las variables (capas ráster) *rhsMin1* y *rhsNorm* se obtiene un nuevo ráster (*rhsE*) que almacena el valor mínimo al comparar cada píxel de las dos capas.
› Línea 62: se asigna como objeto ráster el resultado almacenado en *rhsE* a la variable *rhsMin1* para realizar el siguiente ciclo. El número de ciclos será igual al número de capas ingresadas por el usuario en la variable *rhs*

5.2.3 Procesamiento capas asociadas con las amenazas naturales

El objetivo de esta parte del *script* es calcular el promedio de las viabilidades asignadas para cada nivel de amenaza en cada capa de amenaza natural.

```
63  an = arcpy.GetParameterAsText(4)
64
65  anBase = arcpy.CreateScratchName("Base_an",\
66   workspace=arcpy.env.scratchGDB)
67  anPromedio = arcpy.CreateScratchName("Prom_an",\
68   workspace=arcpy.env.scratchGDB)
69
70  arcpy.gp.CreateConstantRaster_sa(anBase, 0, "INTEGER", pixel, "")
71  anBase1 = arcpy.sa.Raster(anBase)
72
73  anLista = an.split(";")
74
75  for anTipo in anLista:
76  anViab = viabilidad(anTipo)
77      anSum = anBase1 + arcpy.sa.Raster(anViab)
78  anBase1 = anSum
79
80  anProm = anBase1/int(len(anLista))
81
```

› Línea 63: asignación a la variable *an* de la lista de capas que hacen parte de amenazas naturales. Es un dato de entrada.

- Línea 65: asignación a la variable *anBase* de una ruta para el acceso al dato *Base_an* en la geodatabase Scratch. Este dato es un resultado intermedio durante el procesamiento.
- Línea 67: asignación a la variable *anPromedio* de una ruta para el acceso al dato *Prom_an* en la geodatabase Scratch. Este dato es un resultado intermedio durante el procesamiento.
- Línea 70: creación de un dato tipo ráster, en el cual el dato para cada uno de sus píxeles es igual a cero.
- Línea 71: almacena en la variable *anBase1* la capa *anBase* como objeto ráster para ser utilizado directamente en Python en una expresión algebraica.
- Línea 73: separa como elementos independientes en una lista cada una de las capas ingresadas en la variable *an*.
- Línea 75 a 78: ciclo en el cual se calcula la sumatoria de los datos de viabilidad asignados a los tipos de amenazas naturales y se almacena en la variable *anBase1* al finalizar el último ciclo.
- Línea 75: para cada una de las capas dentro de la variable *anLista*.
- Línea 76: almacena en la variable *anViab* el resultado de la función *viabilidad,* incorporando como parámetro cada una de las capas de la lista *anLista* en cada ciclo.
- Línea 77: se asigna en la variable *anSuma* el resultado de sumar la variable *anBase1* con el resultado de la función *viabilidad*.
- Línea 78: se asigna el valor de la variable *anSuma* a la variable *anBase1*, para almacenar la sumatoria y continuar con el siguiente ciclo.
- Línea 80: se almacena en la variable *anPromedio* el valor de dividir el valor de la variable *anBase1* (sumatoria de los valores de viabilidad asignados a cada tipo de amenaza), entre el total de capas ingresadas por el usuario.

5.2.4 Procesamiento capas asociadas con el potencial hídrico superficial y la capacidad agrológica

En estas líneas se asignan los valores de viabilidad a las capas de potencial hídrico superficial y capacidad agrológica y se transforman en datos tipo ráster.

```
82  phs = arcpy.GetParameterAsText(5)
83  phsViab = viabilidad(phs)
84
85  ca = arcpy.GetParameterAsText(6)
86  caViab = viabilidad(ca)
87
```

- Línea 82: asignación a la variable *phs* de la capa de potencial hídrico subterráneo. Es un dato de entrada.
- Línea 83: se asigna a la variable *phsViab* del resultado de la función viabilidad, siendo el parámetro de entrada la capa almacenada en la variable *phs*.
- Línea 85: asignación a la variable *ca* de la capa de potencial hídrico subterráneo. Es un dato de entrada.
- Línea 86: se asigna a la variable *caViab* del resultado de la función viabilidad, siendo el parámetro de entrada la capa almacenada en la variable *ca*.

5.2.5 Cálculo de la viabilidad para el componente abiótico

Finalmente, se realiza la sumatoria ponderada para los resultados del procesamiento de las capas de recurso hídrico superficial, amenazas naturales, potencial hídrico subterráneo y capacidad agrológica. Los factores de ponderación fueron calculados como se indica en el capítulo 2.

```
88  abioRes = arcpy.CreateScratchName("Res_abio",\
89   workspace=arcpy.env.scratchGDB)
90  abio = 0.41*rhsMin1 + \
91  0.09*anProm + \
92   0.15*arcpy.sa.Raster(phsViab) + \
93  0.31*arcpy.sa.Raster(caViab)
94  abio.save(abioRes)
```

- Línea 88: asignación a la variable *abioRes* de una ruta para el acceso al dato *Res_abio* en la geodatabase *Scratch*. Este dato es el resultado final del *script*.
- Líneas 90 a a93: sumatoria ponderada con los datos calculados en cada una de las partes descritas del *script*. Este resultado se almacena en la variable *abio*.
- Línea 94: el resultado de la sumatoria ponderada se guarda en la variable *abioRes* en la geodatabase *Scratch*.

5.3 Script para el componente sociocultural

5.3.1 Función para la normalización de densidades y distancias

Con esta función se lleva a cabo la normalización de las capas ráster obtenidas después de aplicar las herramientas de distancia euclidiana

o densidad Kernel. Para el caso de la normalización de la distancia se puede elegir entre las tres formas propuestas en la Figura 24, y para el caso de la densidad Kernel solamente la opción de normalización lineal.

```
8   def normaliza(capaN,funN,dN,tipoN):
9   if funN == "Lineal":
10          capLin2 = arcpy.CreateScratchName("Lin_cap",\
11   workspace=arcpy.env.scratchGDB)
12          capLin1 = arcpy.sa.Raster(capaN)/dN
13  arcpy.gp.Con_sa(capLin1, "1", capLin2, capLin1, """"VALUE" > 1""")
14  if tipoN == "Distancia":
15              capNorm = arcpy.sa.Raster(capLin2)
16          if tipoN == "Densidad":
17              capNorm = 1- arcpy.sa.Raster(capLin2)
18      if funN == "Exponencial" and tipoN == "Distancia":
19          capExpo = arcpy.CreateScratchName("Expo_cap",\
20   workspace=arcpy.env.scratchGDB)
21          expo = arcpy.sa.Raster(capaN)*(-4.605/dN)
22  arcpy.gp.Exp_sa(expo,capExpo)
23  capNorm = 1-arcpy.sa.Raster(capExpo)
24  if funN == "Valor Medio" and tipoN == "Distancia":
25  capNorm = \
26  arcpy.sa.Raster(capaN)**2/(arcpy.sa.Raster(capaN)**2 + dN**2)
27      return capNorm
28
```

- Línea 8: definición de la función *normaliza,* cuyos parámetros son la capa ráster de distancia o densidad, la función de normalización, la distancia o densidad máxima de normalización y el tipo de capa a normalizar.
- Línea 9: se evalúa si la función de normalización es de tipo lineal.
- Línea 12: normalización aplicando la ecuación lineal.
- Línea 13: la normalización lineal da como resultado píxeles con valores mayores que uno, por lo que estos valores se cambian por uno utilizando la herramienta condicional.
- Línea 14: se evalúa si el tipo de capa es de distancia.
- Línea 15: el resultado de la normalización se transforma en un objeto ráster para su posterior uso en operaciones algebraicas.
- Línea 16: se evalúa si el tipo capa es de densidad.
- Línea 17: el cálculo de la normalización se completa y el resultado se almacena como un objeto ráster.
- Línea 18: se evalúa si la normalización es exponencial y el ráster es de distancia.

Línea 21: cálculo del exponente de la función (ver Figura 24).
Línea 22: cálculo de la función exponencial con base *e* para cada píxel en la capa ráster.
Línea 23: el cálculo de la normalización se completa y el resultado se almacena como un objeto ráster.

Línea 24: se evalúa si la función de normalización es valor medio y el dato ráster es de distancia.
Línea 25: se aplica la ecuación para la normalización.
Línea 27: la función retorna como resultado la capa ráster *capNorm*.

5.3.2 Función para el cálculo de distancias o densidades

Esta función aplica las herramientas de análisis para calcular los ráster de distancia euclidiana o densidad Kernel.

```
29  def dist_den(capaDD,campoD,h,funD,dD,tipoD):
30      count1 = int(arcpy.GetCount_management(capaDD).getOutput(0))
31      if count1 > 0:
32          capD = arcpy.CreateScratchName("D_cap",\
33   workspace=arcpy.env.scratchGDB)
34          if tipoD == "Distancia":
35              arcpy.gp.EucDistance_sa(capaDD, capD, "", pixel, "")
36              capND = normaliza(capD,funD,dD,tipoD)
37          if tipoD == "Densidad":
38  arcpy.gp.KernelDensity_sa(capaDD,campoD,capD,\
39  pixel,h,"SQUARE_KILOMETERS")
40              capND = normaliza(capD,"Lineal",dD,tipoD)
41      else:
42          capD = arcpy.CreateScratchName("D_cap",\
43   workspace=arcpy.env.scratchGDB)
44          arcpy.gp.CreateConstantRaster_sa(capD, 1, "INTEGER", pixel, "")
45          capND = arcpy.sa.Raster(capD)
46      return capND
```

- Línea 29: definición de la función *dist_den,* cuyos parámetros son la capa sobre la cual se va a aplicar la distancia euclidiana o densidad Kernel, el campo con los datos de población para el cálculo de la densidad, el radio para la aplicación de la función Kernel, la función de normalización, la distancia o densidad máxima de normalización y el tipo de análisis por realizar (distancia o densidad).
- Línea 30: conteo del número de elementos en la capa de entrada para la zona de análisis definida en la extensión de análisis.
- Línea 31: se evalúa si hay más de un elemento en la extensión de análisis.
- Línea 34: se evalúa si el tipo de análisis es de distancia euclidiana.
- Línea 35: aplicación de la herramienta de análisis para obtener un ráster de distancia euclidiana. El tamaño de píxel del ráster de salida es el definido por el usuario en la línea 5.
- Línea 36: se aplica la función *normaliza*.
- Línea 37: se evalúa si el tipo de análisis por realizar es de densidad.
- Línea 38: aplicación de la herramienta de análisis para obtener un ráster de densidad Kernel. El tamaño de píxel del ráster de salida será el definido por el usuario en la línea 5.

› Línea 40: se aplica la función *normaliza*.
› Línea 41: se evalúa si no hay elementos en la extensión de análisis.
› Línea 44: creación de una capa en la cual los píxeles tienen un valor igual a uno. Es decir que la actividad minera es viable en esa extensión de análisis para la capa analizada.
› Línea 46: la función regresa la capa de distancia o densidad normalizada.

5.3.3 Función para el análisis de múltiples capas

Para los casos de las categorías de provisión de servicios públicos, población y áreas de importancia cultural o arqueológica, en las cuales se tiene más de una capa de entrada, se propone una función para calcular la densidad o distancia una por una y después calcular el valor mínimo entre los datos ráster estimados.

```
47  def cicloDD(capasC,campoC,h,funC,dC,tipoC):
48  capMin = arcpy.CreateScratchName("Min_psp",\
49   workspace=arcpy.env.scratchGDB)
50
51      arcpy.gp.CreateConstantRaster_sa(capMin, 1, "INTEGER", pixel, "")
52      capMin1 = arcpy.sa.Raster(capMin)
53
54      for CapaTipo in capasC:
55          capaE = arcpy.CreateScratchName("E_capa",\
56   workspace=arcpy.env.scratchGDB)
57  cNorm = dist_den(CapaTipo,campoC,h,funC,dC,tipoC)
58  arcpy.gp.CellStatistics_sa([capMin1, cNorm], capaE, \
59  "MINIMUM", "DATA")
60          capMin1 = arcpy.sa.Raster(capaE)
61      return capMin1
```

› Línea 47: definición de la función *cicloDD,* cuyos parámetros son la lista de las capas que hacen parte de cualquiera de las categorías provisión de servicios públicos, población o áreas de importancia cultural y arqueológica, el campo con los datos de población para el cálculo de la densidad, el radio para la aplicación de la función Kernel, la función de normalización, la distancia o densidad máxima de normalización y el tipo de análisis por realizar (distancia o densidad).
› Línea 51: creación de una capa ráster con píxeles iguales a uno, la cual servirá como base para calcular el valor mínimo utilizando las capas normalizadas de los datos de entrada.
› Línea 54: ciclo que se realiza para cada una de las capas en la lista.
› Línea 57: función para el cálculo de la distancia euclidiana o densidad y su correspondiente normalización. En la primera iteración se realiza para la primera capa de la categoría.

› Línea 58: cálculo de estadística del mínimo valor entre el ráster con valor uno y la primera capa normalizada.
› Línea 60: la capa con los valores mínimos reemplaza la capa creada inicialmente con valores de uno para todos sus píxeles. Luego se lleva a cabo con la siguiente capa el mismo procedimiento.
› Línea 61: el resultado de la función será la capa en la que se tengan los valores mínimos para cada píxel, considerando la normalización de todos los datos de la lista de entrada para la categoría de análisis.

5.3.4 Procesamiento de las capas para población

En esta parte del *script* se incorporan los datos necesarios para procesar la categoría de población. Se incluye la transformación de los polígonos ingresados, asociados con centros poblados, en puntos aleatorios para poder realizar el cálculo de la densidad Kernel.

```
62  ah = arcpy.GetParameterAsText(2)
63  d_ah = int(arcpy.GetParameterAsText(3))
64  h_ah = int(arcpy.GetParameterAsText(4))
65
66  ahLista = ah.split(";")
67
68  for ahTipo in ahLista:
69      desc = arcpy.Describe(ahTipo)
70      if desc.shapeType == "Polygon":
71          count = int(arcpy.GetCount_management(ahTipo).getOutput(0))
72          if count > 0:
73              arcpy.AddField_management(ahTipo,"nu_puntos", "LONG", "", "",\
74   "", "", "NULLABLE", "NON_REQUIRED", "")
75              arcpy.AddField_management(ahTipo,"poblacion","LONG","",\
76  "","","","NULLABLE","NON_REQUIRED","")
77  arcpy.CalculateField_management(ahTipo, "nu_puntos",\
78  "[Shape_Area]/10000", "", "")
79  calc = """def calculo(hab,nu):
80  hab1 = hab/nu
81                  return hab1"""
82              arcpy.env.workspace = arcpy.env.scratchGDB
83  arcpy.CalculateField_management(ahTipo, "poblacion",\
84  "calculo(!pob_cp!,!nu_puntos!)", "PYTHON", calc)
85  arcpy.CreateRandomPoints_management(arcpy.env.scratchGDB,"puntos",ahTipo,\
86  "","nu_puntos","0 Meters","POINT","")
87  puntos_ah = arcpy.CreateScratchName("ah_zu_p",\
88  workspace=arcpy.env.scratchGDB)
89  arcpy.SpatialJoin_analysis("puntos",ahTipo,puntos_ah,\
90  "JOIN_ONE_TO_ONE","KEEP_ALL","","INTERSECT","#","#")
91  ahLista.remove(ahTipo)
92  ahLista.insert(0,puntos_ah)
93
94  ahNorm = cicloDD(ahLista,"poblacion",h_ah,"Lineal",d_ah,"Densidad")
95
```

› Línea 62: asignación a la variable *ah* de las capas de población. Es un dato de entrada.
› Línea 63: asignación a la variable *d_ah* del valor máximo de densidad para la normalización. Este valor es incorporado por el usuario.

- Línea 64: asignación a la variable *h_ah* del valor del radio de análisis para generar el ráster de densidad Kernel. Este valor es introducido por el usuario.
- Línea 66: separa como elementos independientes en una lista, cada una de las capas incorporadas en la variable *ah*.
- Línea 68: ciclo que se ejecuta para las capas en la lista.
- Línea 69: asignación a la variable *desc* de la descripción de la capa en la primera iteración del bucle.
- Línea 70: evalúa si el tipo de geometría almacenado en la variable *desc* es polígono.
- Línea 71: almacena en la variable *count* el número de elementos que hacen parte de cada una de las capas. Solamente se contabilizan los elementos dentro de la extensión definida en la línea 7.
- Línea 72: evalúa si la variable *count* es mayor que cero, es decir, si hay elementos en el área de análisis para la capa correspondiente.
- Línea 73 y 75: se adicionan dos campos en la tabla de atributos de la capa tipo polígono. En el campo *nu_puntos* se calcula para cada polígono la cantidad de puntos que se crearán dentro del polígono. En el campo *población* se calcula para cada polígono la población total asociada al polígono, divida en el número de puntos calculados en la capa *nu_puntos*.
- Línea 77: cálculo del número de puntos que se generarán de forma aleatoria para cada uno de los polígonos en la capa.
- Líneas 79 a 81: definición de la función para calcular el número de habitantes que representa cada punto dentro de cada polígono de la capa.
- Línea 82: asignación de la ruta de la geodatabase *Scratch* como espacio de trabajo, a fin de evitar incorporar la ruta completa de los datos al ejecutar las herramientas de análisis de ArcGIS.
- Línea 83: cálculo del campo *población* utilizando la función definida en las líneas 79 a 81.
- Línea 85: creación de puntos aleatorios dentro de cada polígono de la capa en el área de análisis. Se creará una capa de nombre *puntos,* en la cual el número de puntos por polígono será el almacenado en el campo *nu_puntos*.
- Línea 89: se utiliza la herramienta de unión espacial para transferir los atributos de la capa de polígono a la nueva capa de puntos.
- Línea 91: se elimina de la lista de capas, la capa de polígono que se acaba de procesar.

› Línea 92: se adiciona a la lista, la capa de puntos que se acaba de generar. Después de esto, se continúa con la siguiente iteración.
› Línea 94: se utiliza la función *cicloDD* para procesar cada una de las capas de la lista y obtener una capa en la que se almacenen los valores mínimos de viabilidad al sobreponer los resultados de densidad normalizados.

5.3.5 Procesamiento de las capas para provisión de servicios públicos y áreas de interés arqueológico y cultural

Teniendo en cuenta que ya se definieron las funciones necesarias, en esta parte del *script* se hace la asignación de los datos incorporados por el usuario a las variables correspondientes y se aplica la función respectiva. Estos dos componentes incluyen el ingreso de varias capas a la variable de entrada.

```
96    psp = arcpy.GetParameterAsText(5)
97    d_psp = int(arcpy.GetParameterAsText(6))
98    fn_psp = arcpy.GetParameterAsText(7)
99
100   pspLista = psp.split(";")
101   pspNorm = cicloDD(pspLista,"","",fn_psp,d_psp,"Distancia")
102
103   ica = arcpy.GetParameterAsText(8)
104   d_ica = int(arcpy.GetParameterAsText(9))
105   fn_ica = arcpy.GetParameterAsText(10)
106
107   icaLista = ica.split(";")
108   icaNorm = cicloDD(icaLista,"","",fn_ica,d_ica,"Distancia")
109
```

› Líneas 96 y 103: asignación de las capas correspondientes a provisión de servicios públicos a la variable *psp,* y de las capas con datos culturales y arqueológicos a la variable *ica*.
› Líneas 97 y 104: asignación a las variables *d_psp* y *d_ica* de los valores máximos de distancia para la normalización de las distancias euclidianas.
› Líneas 98 y 105: asignación a las variables *fn_psp* y *fn_ica* del nombre de las funciones de normalización por aplicar (lineal, exponencial o valor medio).
› Líneas 100 y 107: separación, como elementos independientes en la lista, de cada una de las capas incorporadas en las variables *psp* e *ica*.
› Líneas 101 y 108: aplicación de la función *cicloDD* y asignación del resultado en las variables *pspNorm* e *icaNorm*.

5.3.6 Cálculo de densidades y normalización de las capas de carreteras y de equipamientos públicos

Las capas de entrada para las carreteras y equipamientos públicos son una para cada caso. Las carreteras en geometría línea y los equipamientos en geometría punto. Por lo tanto, se utiliza directamente la función para calcular la densidad Kernel y su correspondiente normalización.

```
110  car = arcpy.GetParameterAsText(11)
111  d_ica = int(arcpy.GetParameterAsText(12))
112  h_car = int(arcpy.GetParameterAsText(13))
113
114  carNorm = dist_den(car,"transito",h_car,"Lineal",d_ica,"Densidad")
115
116  ep = arcpy.GetParameterAsText(14)
117  d_ep = int(arcpy.GetParameterAsText(15))
118  h_ep = int(arcpy.GetParameterAsText(16))
119
120  epNorm = dist_den(ep,"poblacion",h_ep,"Lineal",d_ep,"Densidad")
121
```

- Líneas 110 y 116: asignación de la capa de carreteras a la variable *car* y de la capa de equipamientos públicos a la variable *ep*.
- Líneas 111 y 117: asignación a las variables *d_car* y *d_ep* de los valores máximos de densidad para la normalización.
- Líneas 112 y 118: asignación a las variables *h_car* y *h_ep* de los valores para el radio de análisis de la densidad Kernel.
- Líneas 114 y 120: aplicación de la función *dist_den* y asignación del resultado en las variables *carNormyepNorm*.

5.3.7 Cálculo de la viabilidad para el componente sociocultural

Finalmente, se hace la sumatoria ponderada para los resultados del procesamiento de las capas de provisión de servicios públicos, población, áreas de interés arqueológico o cultural, carreteras y equipamientos públicos. Los factores de ponderación fueron calculados en el capítulo 2.

```
122  socioRes = arcpy.CreateScratchName("Res_sc",\
123   workspace=arcpy.env.scratchGDB)
124  socio = 0.19*pspNorm + \
125  0.39*ahNorm + \
126  0.09*icaNorm + \
127  0.10*carNorm + \
128  0.23*epNorm
129  socio.save(socioRes)
```

- Línea 122: asignación a la variable *socioRes* de una ruta para el acceso al dato *Res_sc* en la geodatabase *Scratch*. Este dato es el resultado final.

› Líneas124 a 128: sumatoria ponderada con los datos calculados en cada una de las partes descritas del *script*. Este resultado se almacena en la variable *socio*.
› Línea 129: el resultado de la sumatoria ponderada se guarda en la variable *socioRes* en la geodatabase *Scratch*.

5.4 Script para el componente biótico

El objetivo de este *script* es combinar las capas de cobertura vegetal y sensibilidad ambiental. Para la capa de cobertura vegetal se efectúa la conversión de dato vector polígono a ráster, en donde el valor del píxel será el código de la leyenda Corine Land Cover, y después se reclasifica con los valores de viabilidad. Para el caso de la sensibilidad ambiental, se agrega el nivel de viabilidad en la tabla de atributos y después se transforma a dato ráster.

```
8   cob = arcpy.GetParameterAsText (2)
9   tab_viab = arcpy.GetParameterAsText (3)
10
11  cobRaster = arcpy.CreateScratchName("Raster_cob", \
12  workspace=arcpy.env.scratchGDB)
13  arcpy.PolygonToRaster_conversion(cob, "CODIGO", cobRaster, \
14  "MAXIMUM_AREA", "", pixel)
15  cobViab = arcpy.CreateScratchName("Viab_cob", \
16  workspace=arcpy.env.scratchGDB)
17  arcpy.gp.ReclassByTable_sa(cobRaster,tab_viab,"CODIGO",\
18  "CODIGO","viabilidad",cobViab,"NODATA")
19  cobViab1 = arcpy.sa.Raster(cobViab)*0.1
20
21  sens = arcpy.GetParameterAsText (4)
22
23  arcpy.AddField_management(sens,"viabilidad", "FLOAT", \
24  "", "", "", "", "NULLABLE", "NON_REQUIRED", "")
25
26  expresion = """def reclass(Valor):
27      if (Valor=='Muy Alta'):
28  return 0.2
29      elif (Valor=='Alta'):
30          return 0.4
31  elif (Valor=='Media'):
32          return 0.6
33  elif (Valor=='Baja'):
34          return 0.8
35      elif (Valor=='Muy Baja'):
36  return 1.0
37      else:
38          return 0"""
39
40  arcpy.CalculateField_management(sens, "viabilidad",\
41   "reclass(!sensible!)", "PYTHON", expresion)
42  sensViab=arcpy.CreateScratchName("Viab_sens",\
43   workspace=arcpy.env.scratchGDB)
44  arcpy.PolygonToRaster_conversion(sens, "viabilidad",\
45   sensViab, "CELL_CENTER", "NONE", pixel)
46
47  bioRes = arcpy.CreateScratchName("Res_bio",\
48   workspace=arcpy.env.scratchGDB)
49  bio = 0.7*cobViab1 + 0.3*arcpy.sa.Raster(sensViab)
50  bio.save(bioRes)
```

- Líneas 8 y 9: asignación de la capa de cobertura del suelo a la variable *cob* y de la tabla para la reclasificación del código de cobertura utilizando los valores de viabilidad (ver Tabla 14) en la variable *tab_viab*.
- Línea 13: transforma la capa de polígonos de cobertura de suelo en una capa ráster, en la cual los valores de los píxeles son los códigos de la leyenda de la metodología CORINE Land Cover aplicada para Colombia.
- Línea 17 y 19: reclasificación de la capa ráster con los códigos de cobertura del suelo, utilizando los datos de la tabla incorporada en la variable *tab_viab*. Para la reclasificación, solo se pueden utilizar números enteros, por lo que en la línea 19 multiplica la capa ráster por 0.1, para tener valores en el rango cero a uno.
- Línea 21: asignación de la capa de sensibilidad ambiental a la variable *sens*.
- Línea 23: Creación de un nuevo campo en la tabla de atributos de la capa de sensibilidad ambiental. Es este campo se calculará el valor de viabilidad.
- Líneas 26 a 38: definición de la función para calcular la viabilidad en la capa de sensibilidad ambiental, a partir de los niveles de sensibilidad.
- Línea 40: cálculo del campo viabilidad en la capa de sensibilidad ambiental.
- Línea 44: transformación de la capa de polígonos de sensibilidad ambiental en una capa ráster con valores de viabilidad.
- Línea 47: asignación a la variable *bioRes* de una ruta para el acceso al dato *Res_bio* en la geodatabase *Scratch*. Este dato es el resultado final del *script*.
- Línea 49: sumatoria ponderada con los datos calculados para cobertura del suelo y sensibilidad ambiental. Este resultado se almacena en la variable *bio*.
- Línea 50: el resultado de la sumatoria ponderada se guarda en la variable *bioRes* en la geodatabase *Scratch*.

5.5 Script para la sumatoria ponderada de los componentes abiótico, biótico y sociocultural

Con este *script* se combinan las capas obtenidas en *scripts* descritos en los numerales anteriores. Se utiliza como dato de entrada una tabla

de atributos en la que se consignan los valores de ponderación. Este sería un ejemplo de cómo incorporar los valores de ponderación por el usuario, lo que daría flexibilidad para el cálculo de las sumatorias ponderadas.

```
1   import arcpy
2   arcpy.env.overwriteOutput = True
3
4   abioViab = arcpy.GetParameterAsText(0)
5   bioViab = arcpy.GetParameterAsText(1)
6   socioViab = arcpy.GetParameterAsText(2)
7   pond = arcpy.GetParameterAsText(3)
8   ambViab = arcpy.GetParameterAsText(4)
9
10  filas = arcpy.SearchCursor(pond)
11  for fila in filas:
12      if fila.getValue("componente") == "biotico":
13          bio = arcpy.sa.Raster(bioViab)*fila.getValue("ponderador")
14      if fila.getValue("componente") == "abiotico":
15          abio = arcpy.sa.Raster(abioViab)*fila.getValue("ponderador")
16  if fila.getValue("componente") == "sociocultural":
17          socio = arcpy.sa.Raster(abioViab)*fila.getValue("ponderador")
18
19  amb = bio + abio + socio
20  amb.save(ambViab)
```

- Línea 4: asignación a la variable *abioViab* de la capa ráster de viabilidad del componente abiótico, introducida por el usuario.
- Línea 5: asignación a la variable *bioViab* de la capa ráster de viabilidad del componente biótico, incorporada por el usuario.
- Línea 6: asignación a la variable *socioViab* de la capa ráster de viabilidad del componente sociocultural, incorporada por el usuario.
- Línea 7: asignación a la variable *pond* de la tabla con los valores de ponderación para cada componente.
- Línea 8: asignación a la variable *ambViab* de la ruta y nombre de archivo para almacenar el resultado del *script*.
- Línea 10: lectura de la tabla almacenada en la variable *pond* para extraer los valores de ponderación de cada componente.
- Línea 11: se realiza el ciclo para cada fila en la tabla *pond*.
- Línea 12: evalúa si el valor en el campo *componente* para la fila de análisis es *biótico*.
- Línea 13: asigna a la variable *bio* el resultado de la multiplicación de la capa ráster de viabilidad biótica con su correspondiente ponderador.
- Línea 14: evalúa si el valor en el campo *componente* para la fila de análisis es *abiótico*.
- Línea 15: asigna a la variable *abio* el resultado de la multiplicación de la capa ráster de viabilidad abiótica con su correspondiente ponderador.

› Línea 16: evalúa si el valor en el capo *componente* para la fila de análisis es *sociocultural*.
› Línea 17: asigna a la variable *socio* el resultado de la multiplicación de la capa ráster de viabilidad sociocultural con su correspondiente ponderador.
› Línea 19: asigna a la variable *amb* el valor de la sumatoria de las variables *abio*, *bio* y *socio*. Este es el resultado de la sumatoria ponderada para calcular la viabilidad ambiental.
› Línea 20: se almacena el resultado en la ruta y nombres especificados por el usuario en la variable *ambViab*.

5.6 Procedimiento general para incluir un script en un Toolbox de ArcGIS Desktop

Una vez se han creado los *scripts*, se adicionan como un conjunto de herramientas en el módulo de Arctoolbox de ArcGIS Desktop. En este apartado se muestra el procedimiento general para añadir la herramienta correspondiente al componente sociocultural a ArcGIS y poder hacer uso de ella. Lo primero es dar clic derecho sobre un directorio elegido, en el menú se seleccionan la opción *New* y la opción *Toolbox*. Para este caso se ha asignado el nombre de "Planificación Minera" al *Toolbox* creado (Figura 35).

Figura 35. Creación de un Toolbox.

Una vez creado el *Toolbox*, se da clic derecho sobre este y en el menú se selecciona *Add* y finalmente *Script* (Figura 36).

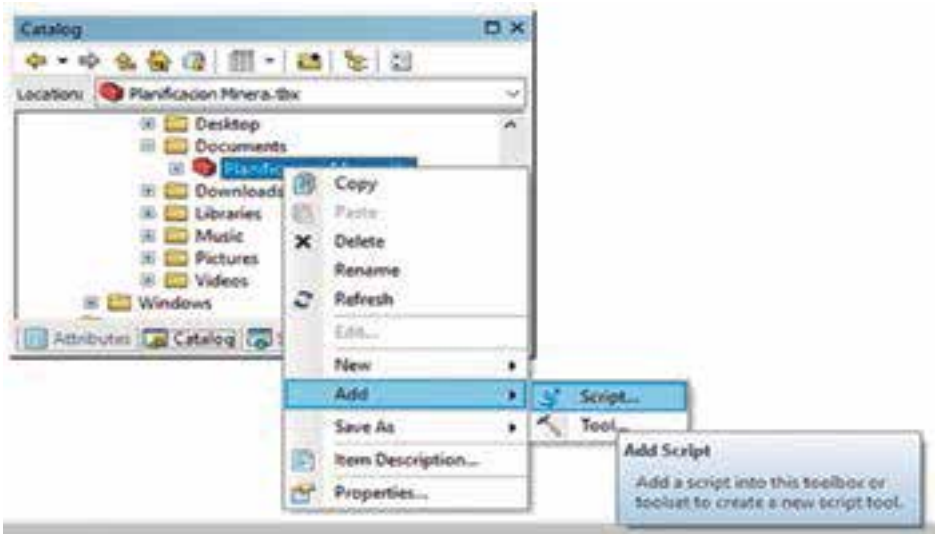

Figura 36. Agregar un script al Toolbox.

Al realizar la anterior instrucción, se abre una ventana en la cual se ingresa la información básica sobre el *script*. El nombre no pude incluir espacios, solamente se aceptan caracteres alfanuméricos y el *label* es el nombre con el que se visualizará en el *Toolbox*. Se pueden almacenar rutas relativas o absolutas para el *script* (Figura 37).

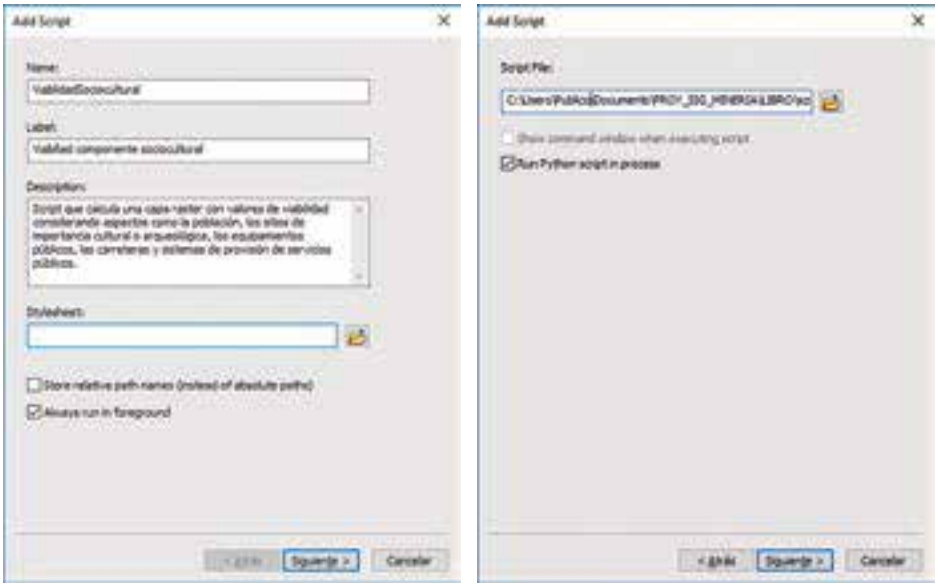

Figura 37. Asignación de información y directorio del modelo

Finalmente, se agregan los parámetros del modelo. Para cada parámetro se asigna el nombre, el tipo de dato y las propiedades. Estos dependen de los parámetros definidos en el *script*, los cuales se corresponden con las líneas de código en las que se utiliza la función arcpy.GetParameterAsText(0). Se deben incorporar en el orden dado por el valor del paréntesis de la función. Para el caso de este ejemplo, las líneas del *script* son las siguientes:

```
4       limite = arcpy.GetParameterAsText(0)
5       pixel = arcpy.GetParameterAsText (1)
…
55      ah = arcpy.GetParameterAsText(2)
56      d_ah = int(arcpy.GetParameterAsText(3))
57      h_ah = int(arcpy.GetParameterAsText(4))
…
82      psp = arcpy.GetParameterAsText(5)
83      d_psp = int(arcpy.GetParameterAsText(6))
84      fn_psp = arcpy.GetParameterAsText(7)
…
89      ica = arcpy.GetParameterAsText(8)
90      d_ica = int(arcpy.GetParameterAsText(9))
91      fn_ica = arcpy.GetParameterAsText(10)
…
96      car = arcpy.GetParameterAsText(11)
97      d_car = int(arcpy.GetParameterAsText(12))
98      h_car = int(arcpy.GetParameterAsText(13))
…
102     ep = arcpy.GetParameterAsText(14)
103     d_ep = int(arcpy.GetParameterAsText(15))
104     h_ep = int(arcpy.GetParameterAsText(16))
```

Las ventanas para la incorporación y la validación de la información de los parámetros son las que se presentan a continuación (Figura 38):

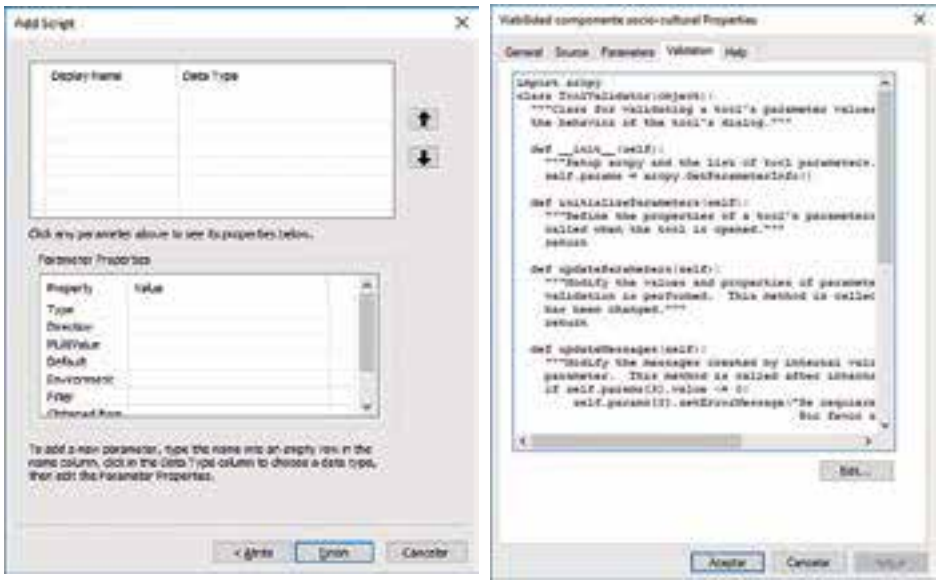

Figura 38. Ventana para la incorporación de los parámetros del modelo

En la Tabla 15 se resumen los valores que se han de incorporar para cada parámetro. Después de incorporados, se da clic en finalizar y ya se puede ejecutar el *script*. Todos los parámetros son de tipo *Required* y la dirección es *Input*.

Tabla 15. Tabla resumen para los parámetros del *script*

N.º	Nombre de Visualización	Tipo de dato	Propiedades del parámetro		
			Valor múltiple	Valor por defecto	Filtro
0	Área de análisis	Extent	No	DEFAULT	None
1	Tamaño de píxel	Cell Size	No	30	None
2	Población	Feature Class	Si		Feature Class: Point/Multipoint/ Polygon
3	Población: densidad de normalización	Long	No		None
4	Población: radio de análisis función Kernel	Long	No		None
5	Provisión de servicios públicos	Feature Class	Si		Feature Class: Point/Multipoint/ Polygon
6	Provisión de servicios públicos: distancia de normalización	Long	No		None
7	Provisión de servicios públicos: función de normalización	String	No	Lineal	Value List Values: Lineal/ Exponencial/ Valor Medio
8	Áreas de importancia cultural o arqueológica	Feature Class	Sí		Feature Class: Point/Multipoint/ Polygon
9	Áreas de importancia cultural o arqueológica: distancia de normalización	Long	No		None
10	Áreas de importancia cultural o arqueológica: función de normalización	String	No	Lineal	Value List Values: Lineal/ Exponencial/ Valor Medio
11	Carreteras	Feature Class	No		Feature Class: Polyline
12	Carreteras: densidad de normalización	Long	No		None
13	Carreteras: radio de análisis función Kernel	Long	No		None
14	Equipamientos públicos	Feature Class	No		Feature Class: Point/Multipoint/ Polygon
15	Equipamientos públicos: densidad de normalización	Long	No		None
16	Equipamientos públicos: radio de análisis función Kernel	Long	No		None

Se presenta a continuación un ejemplo de validación, en el cual se mostrará un mensaje de error cuando se introduzcan valores de normalización, para la distancia o la densidad, iguales o menores que cero. En la ventana se edita el código y se modifica la función correspondiente a la actualización de mensajes. En la Figura 39, se muestra el ejemplo del código de validación.

Figura 39. Ejemplo de validación de parámetros.