# Documentation of tools used in nrip-jamaica GitHub repository

Contents

[Documentation of tools used in nrip-jamaica github repository 1](#_Toc87352415)

[1.0 Library installation guide 2](#_Toc87352416)

[2.0 Tool set-up guide 2](#_Toc87352417)

[3.0 Open-source tools 4](#_Toc87352418)

[3.1 Geomorphological Fluvial Flood Hazard Areas 4](#_Toc87352419)

[3.2 Steep Areas 7](#_Toc87352420)

[3.3 Clipping By Elevation 8](#_Toc87352421)

[3.4 Hydrological Routing 8](#_Toc87352422)

[3.5 Inundation extents 12](#_Toc87352423)

[3.6 Intersect 13](#_Toc87352424)

[3.7 Zonal Statistics 13](#_Toc87352425)

[3.8 Distance from points 14](#_Toc87352426)

[3.9 Distance from lines 15](#_Toc87352427)

[3.10 Distance from polygons 16](#_Toc87352428)

[3.11 Distance from raster 16](#_Toc87352429)

[3.12 Hotspots from points 16](#_Toc87352430)

[3.13 Hotspots from lines 17](#_Toc87352431)

[3.14 Gaussian filter 17](#_Toc87352432)

[3.15 Hotspots from polygons 18](#_Toc87352433)

[3.16 Hotspots from raster 18](#_Toc87352434)

[3.17 Interpolate points 18](#_Toc87352435)

[3.18 Summarize within 19](#_Toc87352436)

[4.0 Arcpy Tools 19](#_Toc87352437)

[4.1 Zonal statistics 19](#_Toc87352438)

[4.2 Summarize within 19](#_Toc87352439)

[4.3 Calculate proximity 19](#_Toc87352440)

[4.4 Calculate hotspots 20](#_Toc87352441)

[4.5 Buffer vector 22](#_Toc87352442)

[4.6 Intersect 22](#_Toc87352443)

[4.7 Calculate slope 23](#_Toc87352444)

[4.8 Calculate aspect 23](#_Toc87352445)

[4.9 Steep areas 24](#_Toc87352446)

[4.10 Calculate fill 25](#_Toc87352447)

[4.11 Calculate flow direction 25](#_Toc87352448)

[4.12 Calculate flow accumulation 26](#_Toc87352449)

[4.13 Calculate flow network 26](#_Toc87352450)

[4.14 Complete hydrological routing 27](#_Toc87352451)

[4.15 SetNull below 28](#_Toc87352452)

[4.16 SetNull between 28](#_Toc87352453)

[4.17 SetNull above 28](#_Toc87352454)

[4.18 Inundation extents 28](#_Toc87352455)

This document aims to describe all tools created for the NRIP Jamaica works, with a focus on the underlying geoprocessing functions that run each tool.

# Library installation guide

To run the tools there are a few required python libraries that must be installed. These are Geopandas and WhiteboxTools. GeoPandas is an open-source library that extends the datatypes used by pandas to allow spatial operations on geometric types. WhiteboxTools is an open-source geospatial data analysis platform.

Before installing these libraries, it is recommended to have the conda package manager. This can be obtained by installing the Anaconda Distribution (a free Python distribution for data science), or through miniconda (minimal distribution only containing Python and the conda package manager).

Once set up, you can then use the Miniconda/Anaconda shell to install GeoPandas. GeoPandas and all its dependencies are available on the conda-forge channel. Simply enter the following to the shell:

*conda install --channel conda-forge geopandas*

WhiteboxTools are installed in the same manner. Simply enter the following to the Miniconda/Anaconda shell:

*conda install -c conda-forge whitebox*

To run the arcpy tools, arcpy must be installed on the local computer. It comes pre-installed with ArcGIS so only has to be imported in a notebook within ArcGIS.

# Tool set-up guide

These tools were built to function in a python notebook or a standalone python script. There are two groups of tools created. These are the open-source tools and the arcpy tools.

The open-source tools are designed to be run by any individual user using free, open-source GIS software. The underlying software running the functions is WhiteboxTools, an advanced open-source geospatial data analysis platform developed by the University of Guelph. The open-source component is made up of six python files, with the bespoke tools written within these scripts as callable functions. The five python files are as follows:

1. *geometric\_analysis.py*
2. *geomorphological\_analysis.py*
3. *hydrological\_analysis.py*
4. *inundation\_analysis.py*
5. *wbt\_utils.py*

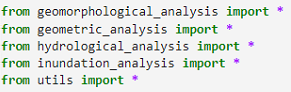
The arcpy tools are designed to be run within the ESRI environment using ESRI products. The underlying software running the functions is arcpy, ESRI’s Python site package. The arcpy component is made up of five python files with the bespoke tools written within these scripts as callable functions. The six python files are as follows:

1. *conditional\_eval.py*
2. *geometric\_analysis.py*
3. *geomorphological\_analysis.py*
4. *hydrological\_analysis.py*
5. *inundation\_analysis.py*
6. *utils.py*

To help with the familiarisation of the product, we have created an example notebook for both open-source and arcpy tools. These detail the necessary imports and show working examples of specific functions. The notebook files are as follows:

1. *examples\_whitebox\_toolset.ipynb*
2. *examples\_arcpy\_toolset.ipynb*

The first steps in using these tools will be to clone the needed **nrip-jamaica** repository from our online Síor GitHub to your local device. The link for the generic repository which contains a project summary and necessary links is <https://github.com/siorconsulting/nrip-jamaica>. The links for the open-source and ArcGIS toolsets are <https://github.com/siorconsulting/nrip-jamaica-open-source> and <https://github.com/siorconsulting/nrip-jamaica-arcgis>. This can be done through GIT or by downloading the repository as a ZIP file. If using the open-source tools, a new notebook can be created within the same directory. Within this notebook you can import the files needed for analysis. Optionally, you can import all files for full functionality as follows:



It is important to note that whether you are importing one file, or all, that utils always be imported. This also goes for the arcpy tools. Once the necessary files have been imported, the functions within them must be called from the notebook. To call a specific function, for example Geomorphological Fluvial Flood Hazard Areas, simply type the first few letters of the function and press Tab. This will open a list of all functions beginning with the same letters. From here you can select the function and insert it into the notebook. After it is inserted, it can be analysed by pressing Shift-Tab. This will show any information related to the function such as inputs, outputs, and file location. Once you have this, you can begin to fill out the function with the required parameters.

The arcpy tools run in much the same manner, however the notebook can be opened within ArcGIS, meaning arcpy only has to be imported within the notebook. If using a third-party IDE, the python environment will have to be changed to the ArcGIS environment to run arcpy. To import all files for full functionality, use the following imports:

Text

Description automatically generated

# Open-source tools

## Geomorphological Fluvial Flood Hazard Areas

**Outline:** calculates flood hazard areas from a raster and outputs these areas as polygons.

**Source:** this tool is part of *wbt\_geomorphological\_analysis.py*.

**Description:**

This tool is composed of 6 whitebox tools with the following workflow:

1. fill\_depressions\_planchon\_and\_darboux()
2. d8\_pointer()
3. d8\_flow\_accumulation()
4. conditional\_evaluation()
5. buffer\_raster()
6. raster\_to\_vector\_polygons()

**Fill depressions planchon and darboux *[****[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/hydrological_analysis.html" \l "FillDepressionsPlanchonAndDarboux)****]***

This tool can be used to fill all of the depressions in a digital elevation model (DEM) and to remove the flat areas using the Planchon and Darboux (2002) method. This is a common pre-processing step required by many flow-path analysis tools to ensure continuous flow from each grid cell to an outlet located along the grid edge. This tool is currently not the most efficient depression-removal algorithm available in WhiteboxTools; FillDepressions and BreachDepressionsLeastCost are both more efficient and often produce better, lower-impact results.

The user may optionally specify the size of the elevation increment used to solve flats (--flat\_increment), although it is best not to specify this optional value and to let the algorithm determine the most suitable value itself.

**D8 pointer *[****[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/hydrological_analysis.html" \l "D8Pointer)****]***

This tool is used to generate a flow pointer grid using the simple D8 (O'Callaghan and Mark, 1984) algorithm. The user must specify the name (--dem) of a digital elevation model (DEM) that has been hydrologically corrected to remove all spurious depressions and flat areas. DEM pre-processing is usually achieved using either the BreachDepressions or FillDepressions tool. The local drainage direction raster output (--output) by this tool serves as a necessary input for several other spatial hydrology and stream network analysis tools in the toolset. Some tools will calculate this flow pointer raster directly from the input DEM.

By default, D8 flow pointers use the following clockwise, base-2 numeric index convention:

|  |  |  |
| --- | --- | --- |
| . | . | . |
| 64 | 138 | 1 |
| 32 | 0 | 2 |
| 16 | 8 | 4 |

Notice that grid cells that have no lower neighbours are assigned a flow direction of zero. In a DEM that has been pre-processed to remove all depressions and flat areas, this condition will only occur along the edges of the grid. If the pointer file contains ESRI flow direction values instead, the --esri\_pntr parameter must be specified.

Grid cells possessing the NoData value in the input DEM are assigned the NoData value in the output image.

**D8 flow accumulation *[****[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/hydrological_analysis.html" \l "D8FlowAccumulation)****]***

This tool is used to generate a flow accumulation grid (i.e., catchment area) using the D8 (O'Callaghan and Mark, 1984) algorithm. This algorithm is an example of single-flow-direction (SFD) method because the flow entering each grid cell is routed to only one downslope neighbour, i.e., flow divergence is not permitted. The user must specify the name of the input digital elevation model (DEM) or flow pointer raster (--input) derived using the D8 or Rho8 method (D8Pointer, Rho8Pointer). If an input DEM is used, it must have been hydrologically corrected to remove all spurious depressions and flat areas. DEM pre-processing is usually achieved using the BreachDepressionsLeastCost or FillDepressions tools. If a D8 pointer raster is input, the user must also specify the optional --pntr flag. If the D8 pointer follows the Esri pointer scheme, rather than the default WhiteboxTools scheme, the user must also specify the optional --esri\_pntr flag.

In addition to the input DEM/pointer, the user must specify the output type. The output flow-accumulation can be 1) cells (i.e., the number of inflowing grid cells), catchment area (i.e., the upslope area), or specific contributing area (i.e., the catchment area divided by the flow width. The default value is cells. The user must also specify whether the output flow-accumulation grid should be log-transformed (--log), i.e., the output, if this option is selected, will be the natural-logarithm of the accumulated flow value. This is a transformation that is often performed to better visualize the contributing area distribution. Because contributing areas tend to be very high along valley bottoms and relatively low on hillslopes, when a flow-accumulation image is displayed, the distribution of values on hillslopes tends to be 'washed out' because the palette is stretched out to represent the highest values. Log-transformation provides a means of compensating for this phenomenon. Importantly, however, log-transformed flow-accumulation grids must not be used to estimate other secondary terrain indices, such as the wetness index, or relative stream power index.

Grid cells possessing the NoData value in the input DEM/pointer raster are assigned the NoData value in the output flow-accumulation image.

**Conditional evaluation *[****[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/mathand_stats_tools.html" \l "ConditionalEvaluation)****]***

The ConditionalEvaluation tool can be used to perform an if-then-else style conditional evaluation on a raster image on a cell-to-cell basis. The user specifies the names of an input raster image (--input) and an output raster (--output), along with a conditional statement (--statement). The grid cell values in the output image will be determined by the TRUE and FALSE values and conditional statement. The conditional statement is a logical expression that must evaluate to either a Boolean, i.e., TRUE or FALSE. Then depending on how this statement evaluates for each grid cell, the TRUE or FALSE values will be assigned to the corresponding grid cells of the output raster. The TRUE or FALSE values may take the form of either a constant numerical value or a raster image (which may be the same image as the input). These are specified by the --true and --false parameters, which can be either a file name pointing to existing rasters, or numerical values.

The conditional statement is a single-line logical condition. In addition to the common comparison and logical operators, i.e., < > <= >= == (EQUAL TO) != (NOT EQUAL TO) || (OR) && (AND), conditional statements may contain a number of valid mathematical functions. For example:

\* log(base=10, val) -- Logarithm with optional 'base' as first argument.

If not provided, 'base' defaults to '10'.

Example: log(100) + log(e(), 100)

\* e() -- Euler's number (2.718281828459045)

\* pi() -- π (3.141592653589793)

\* int(val)

\* ceil(val)

\* floor(val)

\* round(modulus=1, val) -- Round with optional 'modulus' as first argument.

Example: round(1.23456) == 1 && round(0.001, 1.23456) == 1.235

\* abs(val)

\* sign(val)

\* min(val, ...) -- Example: min(1, -2, 3, -4) == -4

\* max(val, ...) -- Example: max(1, -2, 3, -4) == 3

\* sin(radians) \* asin(val)

\* cos(radians) \* acos(val)

\* tan(radians) \* atan(val)

\* sinh(val) \* asinh(val)

\* cosh(val) \* acosh(val)

\* tanh(val) \* atanh(val)

Notice that the constants Pi and e must be specified as functions, pi() and e(). A number of global variables are also available to build conditional statements. These include the following:

Special variable names for use in conditional statements:

|  |  |
| --- | --- |
| **Name** | **Description** |
| value | The grid cell value. |
| nodata | The input raster's NoData value. |
| null | Same as nodata. |
| minvalue | The input raster's minimum value. |
| maxvalue | The input raster's maximum value. |
| rows | The input raster's number of rows. |
| columns | The input raster's number of columns. |
| row | The grid cell's row number. |
| column | The grid cell's column number. |
| rowy | The row's y-coordinate. |
| columnx | The column's x-coordinate. |
| north | The input raster's northern coordinate. |
| south | The input raster's southern coordinate. |
| east | The input raster's eastern coordinate. |
| west | The input raster's western coordinate. |
| cellsizex | The input raster's grid resolution in the x-direction. |
| cellsizey | The input raster's grid resolution in the y-direction. |
| cellsize | The input raster's average grid resolution. |

The following are examples of valid conditional statements:

value != 300.0

row > (rows / 2)

value >= (minvalue + 35.0)

(value >= 25.0) && (value <= 75.0)

tan(value \* pi() / 180.0) > 1.0

value == nodata

Any grid cell in the input raster containing the NoData value will be assigned NoData in the output raster, unless a NoData grid cell value allows the conditional statement to evaluate to True (i.e., the conditional statement includes the NoData value), in which case the True value will be assigned to the output.

**Buffer raster *[****[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/gis_analysis_distance_tools.html" \l "BufferRaster)****]***

This tool can be used to identify an area of interest within a specified distance of features of interest in a raster data set.

The Euclidean distance (i.e., straight-line distance) is calculated between each grid cell and the nearest 'target cell' in the input image. Distance is calculated using the efficient method of Shih and Wu (2004). Target cells are all non-zero, non-NoData grid cells. Because NoData values in the input image are assigned the NoData value in the output image, the only valid background value in the input image is zero.

The user must specify the input and output image names, the desired buffer size (--size), and, optionally, whether the distance units are measured in grid cells (i.e. --gridcells flag). If the --gridcells flag is not specified, the linear units of the raster's coordinate reference system will be used.

**Raster to vector polygons *[****[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/data_tools.html" \l "RasterToVectorPolygons)****]***

Converts a raster data set to a vector of the POLYGON geometry type. The user must specify the name of a raster file (--input) and the name of the output (--output) vector. All grid cells containing non-zero, non-NoData values will be considered part of a polygon feature. The vector's attribute table will contain a field called 'VALUE' that will contain the cell value for each polygon feature, in addition to the standard feature ID (FID) attribute.

## Steep Areas

**Outline:** it creates a mask of areas with slope equal or higher than a threshold, and export as raster and vector files.

**Source:** this tool is part of *wbt\_geomorphological\_analysis.py*.

**Description:**

This tool is composed of two whitebox tools with the following workflow:

1. slope()
2. condtional\_evaluation()

**Slope *[****[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/geomorphometric_analysis.html?highlight=slope#slope)****]***

This tool calculates slope gradient (i.e., slope steepness in degrees, radians, or percent) for each grid cell in an input digital elevation model (DEM). The user must specify the name of the input DEM (--dem) and the output raster image. The Z conversion factor is only important when the vertical and horizontal units are not the same in the DEM. When this is the case, the algorithm will multiply each elevation in the DEM by the Z conversion factor. If the DEM is in the geographic coordinate system (latitude and longitude), the following equation is used:

zfactor = 1.0 / (111320.0 x cos(mid\_lat))

where mid\_lat is the latitude of the centre of each raster row, in radians.

The tool uses Horn’s (1981) 3rd-order finite difference method to estimate slope. Given the following clock-type grid cell numbering scheme (Gallant and Wilson, 2000):

| 7 | 8 | 1 |

| 6 | 9 | 2 |

| 5 | 4 | 3 |

slope = arctan(fx2 + fy2)0.5

where,

fx = (z3 – z5 + 2(z2 – z6) + z1 – z7) / 8 \* Δx

and,

fy = (z7 – z5 + 2(z8 – z4) + z1 – z3) / \* Δy

Δx and Δy are the grid resolutions in the x and y direction respectively

**Conditional evaluation**

See [Conditional Evaluation](#Conditional_evaluation)

## Clipping By Elevation

**Outline:** i

**Source:** this tool is part of *wbt\_hydrological\_analysis.py.*

**Description:**

This tool is composed of one whitebox tool with the following workflow:

1. conditional\_evaluation()

**Conditional evaluation**

See [Conditional evaluation](#_Conditional_evaluation)

## Hydrological Routing

**Outline:**

it performs a Hydrological Analysis on the input raster and calculates flow lines, basins and sinks derived from input raster.

**Source:** t

**Description:**

This tool is composed of 10 whitebox tools with the following workflow:

1. fill\_depressions\_planchon\_and\_darboux()
2. d8\_pointer()
3. d8\_flow\_accumulation()
4. conditional\_evaluation()
5. raster\_to\_vector\_lines()
6. basins()
7. raster\_to\_vector\_polygons()
8. raster\_calculator()
9. conditional\_evaluation()
10. raster\_to\_vector\_polygons()

**Fill depressions planchon and darboux**

See [Fill depressions planchon and darboux](#fill_depressions_planchon_and_darboux)

**D8 Pointer**

See [D8 pointer](#d8_pointer)

**D8 flow accumulation**

See [D8 flow accumulation](#d8_flow_accumulation)

**Conditional evaluation**

See [Conditional evaluation](#Conditional_evaluation)

**Raster to vector lines [**[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/data_tools.html?highlight=raster%20to%20vector%20lines#rastertovectorlines)**]**

This tool converts raster lines features into a vector of the POLYLINE ShapeType. Grid cells associated with line features will contain non-zero, non-NoData cell values. The algorithm requires three passes of the raster. The first pass counts the number of line neighbours of each line cell; the second pass traces line segments starting from line ends (i.e., line cells with only one neighbouring line cell); lastly, the final pass traces any remaining line segments, which are likely forming closed loops (and therefore do not have line ends).

If the line raster contains streams, it is preferable to use the RasterStreamsToVector instead. This tool will use knowledge of flow directions to ensure connections between stream segments at confluence sites, whereas RasterToVectorLines will not.

**Basins [***[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/hydrological_analysis.html?highlight=basins#basins)***]**

This tool can be used to delineate all of the drainage basins contained within a local drainage direction, or flow pointer raster (--d8\_pntr), and draining to the edge of the data. The flow pointer raster must be derived using the D8Pointer tool and should have been extracted from a digital elevation model (DEM) that has been hydrologically pre-processed to remove topographic depressions and flat areas, e.g., using the BreachDepressions tool. By default, the flow pointer raster is assumed to use the clockwise indexing method used by WhiteboxTools:

|  |  |  |
| --- | --- | --- |
| . | . | . |
| 64 | 138 | 1 |
| 32 | 0 | 2 |
| 16 | 8 | 4 |

If the pointer file contains ESRI flow direction values instead, the --esri\_pntr parameter must be specified.

The Basins and Watershed tools are similar in function but while the Watershed tool identifies the upslope areas that drain to one or more user-specified outlet points, the Basins tool automatically sets outlets to all grid cells situated along the edge of the data that do not have a defined flow direction (i.e., they do not have a lower neighbour). Notice that these edge outlets need not be situated along the edges of the flow-pointer raster, but rather along the edges of the region of valid data. That is, the DEM from which the flow-pointer has been extracted may incompletely fill the containing raster, if it is irregular shaped, and NoData regions may occupy the peripherals. Thus, the entire region of valid data in the flow pointer raster will be divided into a set of mutually exclusive basins using this tool.

**Raster to vector polygons**

See [Raster to vector polygons](#raster_to_vector_polygons)

**Raster calculator [***[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/mathand_stats_tools.html#RasterCalculator)***]**

The RasterCalculator tool can be used to perform a complex mathematical operations on one or more input raster images on a cell-to-cell basis. The user specifies the name of the output raster (--output) and a mathematical expression, or statement (--statement). Rasters are treated like variables (that change value with each grid cell) and are specified within the statement with the file name contained within either double or single quotation marks (e.g., "DEM.tif" > 500.0). Raster variables may or may not include the file directory. If unspecified, a raster is assumed to exist within the working directory. Similarly, if the file extension is unspecified, it is assumed to be '.tif'. \*\*Note, all input rasters must share the same number of rows and columns and spatial extent. Use the Resample tool if this is not the case to convert the one raster's grid resolution to the others.

The mathematical expression supports all of the standard algebraic unary and binary operators (+ - \* / ^ %), as well as comparisons (< <= == != >= >) and logical operators (&& ||) with short-circuit support. The order of operations, from highest to lowest is as follows.

Listed in order of precedence:

|  |  |  |
| --- | --- | --- |
| **Order** | **Symbol** | **Description** |
| (Highest Precedence) | ^ | Exponentiation |
|  | % | Modulo |
|  | / | Division |
|  | \* | Multiplication |
|  | - | Subtraction |
|  | + | Addition |
|  | == != < <= >= > | Comparisons (all have equal precedence) |
|  | && and | Logical AND with short-circuit |
| (Lowest Precedence) | || or | Logical OR with short-circuit |

Several common mathematical functions are also available for use in the input statement. For example:

*\* log(base=10, val) -- Logarithm with optional 'base' as first argument.*

*If not provided, 'base' defaults to '10'.*

*Example: log(100) + log(e(), 100)*

*\* e() -- Euler's number (2.718281828459045)*

*\* pi() -- π (3.141592653589793)*

*\* int(val)*

*\* ceil(val)*

*\* floor(val)*

*\* round(modulus=1, val) -- Round with optional 'modulus' as first argument.*

*Example: round(1.23456) == 1 && round(0.001, 1.23456) == 1.235*

*\* abs(val)*

*\* sign(val)*

*\* min(val, ...) -- Example: min(1, -2, 3, -4) == -4*

*\* max(val, ...) -- Example: max(1, -2, 3, -4) == 3*

*\* sin(radians) \* asin(val)*

*\* cos(radians) \* acos(val)*

*\* tan(radians) \* atan(val)*

*\* sinh(val) \* asinh(val)*

*\* cosh(val) \* acosh(val)*

*\* tanh(val) \* atanh(val)*

Notice that the constants pi and e must be specified as functions, pi() and e(). A number of global variables are also available to build conditional statements. These include the following:

Special variable names for use in conditional statements:

|  |  |
| --- | --- |
| **Name** | **Description** |
| nodata | An input raster's NoData value. |
| null | Same as nodata. |
| minvalue | An input raster's minimum value. |
| maxvalue | An input raster's maximum value. |
| rows | The input raster's number of rows. |
| columns | The input raster's number of columns. |
| row | The grid cell's row number. |
| column | The grid cell's column number. |
| rowy | The row's y-coordinate. |
| columnx | The column's x-coordinate. |
| north | The input raster's northern coordinate. |
| south | The input raster's southern coordinate. |
| east | The input raster's eastern coordinate. |
| west | The input raster's western coordinate. |
| cellsizex | The input raster's grid resolution in the x-direction. |
| cellsizey | The input raster's grid resolution in the y-direction. |
| cellsize | The input raster's average grid resolution. |

The special variable names are case-sensitive. If there are more than one raster inputs used in the statement, the functional forms of the nodata, null, minvalue, and maxvalue variables should be used, e.g., nodata("InputRaster"), otherwise the value is assumed to specify the attribute of the first raster in the statement. The following are examples of valid statements:

*"raster" != 300.0*

*"raster" >= (minvalue + 35.0)*

*("raster1" >= 25.0) && ("raster2" <= 75.0) -- Evaluates to 1 where both conditions are true.*

*tan("raster" \* pi() / 180.0) > 1.0*

*"raster" == nodata*

Any grid cell in the input rasters containing the NoData value will be assigned NoData in the output raster, unless a NoData grid cell value allows the statement to evaluate to True (i.e., the mathematical expression includes the nodata value).

**Conditional Evaluation**

See [Conditional Evaluation](#Conditional_evaluation)

**Raster to vector polygons**

See [Raster to vector to polygons](#raster_to_vector_polygons)

## Inundation extents

**Outline:** it calculates inundation extents based on a specific threshold.

**Source:** this tool is part of *wbt\_inundation\_analysis.py.*

**Description:**

This tool is composed of two whitebox tools with the following workflow:

1. conditional\_evaluation()
2. raster\_to\_vector\_polygons()

**Conditional evaluation**

See [Conditional evaluation](#Conditional_evaluation)

**Raster to vector polygons**

See [Raster to vector polygons](#raster_to_vector_polygons)

## Intersect

**Outline:** function that returns all feature parts that occur in both input layers.

**Source:** this tool is part of wbt\_geometric\_analysis.py

**Description:**

This tool is composed of one whitebox tool with the following workflow:

1. Intersect()

**Intersect [***[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/gis_analysis_overlay_tools.html?highlight=intersect#intersect)***]**

The result of the Intersect vector overlay operation includes all the feature parts that occur in both input layers, excluding all other parts. It is analogous to the OR logical operator and multiplication in arithmetic. This tool is one of the common vector overlay operations in GIS. The user must specify the names of the input and overlay vector files as well as the output vector file name. The tool operates on vector points, lines, or polygon, but both the input and overlay files must contain the same ShapeType.

The Intersect tool is similar to the Clip tool. The difference is that the overlay vector layer in a Clip operation must always be polygons, regardless of whether the input layer consists of points or polylines.

The attributes of the two input vectors will be merged in the output attribute table. Note, duplicate fields should not exist between the inputs layers, as they will share a single attribute in the output (assigned from the first layer). Multipoint ShapeTypes will simply contain a single output feature identifier (FID) attribute. Also, note that depending on the ShapeType (polylines and polygons), Measure and Z ShapeDimension data will not be transferred to the output geometries. If the input attribute table contains fields that measure the geometric properties of their associated features (e.g., length or area), these fields will not be updated to reflect changes in geometry shape and size resulting from the overlay operation.

## Zonal Statistics

**Outline:** calculates zonal statistics based on an input raster, using raster or polygon zones.

**Source:** this tool is part of *wbt\_geometric\_analysis.py*

**Description:**

This tool is composed of two whitebox tools and a geopandas with the following workflow:

1. gpd.read\_file()
2. vector\_polygons\_to\_raster()
3. zonal\_statistics()

**Read files [***[link](https://geopandas.org/en/stable/docs/reference/api/geopandas.read_file.html)***]**

This geopandas tool reads the vector file and is used to create a new index. This new index is then subsequently used as the field from attribute table, which is used to convert the vector polygon to a raster.

**Vector polygons to raster [***[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/data_tools.html?highlight=vectorpolygons#vectorpolygonstoraster)***]**

Converts a vector containing polygons into a raster.

**Zonal statistics [***[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/mathand_stats_tools.html?highlight=zonalstatistics#zonalstatistics)***]**

This tool can be used to extract common descriptive statistics associated with the distribution of some underlying data raster based on feature units defined by a feature definition raster. For example, this tool can be used to measure the maximum or average slope gradient (data image) for each of a group of watersheds (feature definitions). Although the data raster can contain any type of data, the feature definition raster must be categorical, i.e., it must define area entities using integer values.

The --stat parameter can take the values, 'mean', 'median', 'minimum', 'maximum', 'range', 'standard deviation', or 'total'.

If an output image name is specified, the tool will assign the descriptive statistic value to each of the spatial entities defined in the feature definition raster. If text output is selected, an HTML table will be output, which can then be readily copied into a spreadsheet program for further analysis. This is a very powerful and useful tool for creating numerical summary data from spatial data which can then be interrogated using statistical analyses. At least one output type (image or text) must be specified for the tool to operate.

NoData values in either of the two input images are ignored during the calculation of the descriptive statistic.

## Distance from points

**Outline:** it creates a raster showing distances between input points.

**Source:** this tool is part of *wbt\_geometric\_analysis.py*.

**Description:**

This tool is composed of two whitebox tools with the following workflow:

1. vector\_points\_to\_raster()
2. convert\_nodata\_to\_zero()
3. euclidean\_distance()

**Vector points to raster [***[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/data_tools.html?highlight=vectorpoints#vectorpointstoraster)***]**

This tool can be used to convert a vector points file into a raster grid. The user must specify the name of the input vector and the output raster file. The field name (--field) is the field from the attributes table from which the tool will retrieve the information to assign to grid cells in the output raster. The field must contain numerical data. If the user does not supply a field name parameter, each feature in the raster will be assigned the record number of the feature. The assignment operation determines how the situation of multiple points contained within the same grid cell is handled. The background value is zero by default but can be set to NoData optionally using the --nodata value.

If the user optionally specifies the grid cell size parameter (--cell\_size) then the coordinates will be determined by the input vector (i.e., the bounding box) and the specified cell size. This will also determine the number of rows and columns in the output raster. If the user instead specifies the optional base raster file parameter (--base), the output raster's coordinates (i.e., north, south, east, west) and row and column count will be the same as the base file.

In the case that multiple points are contained within a single grid cell, the output can be assigned (--assign) the first, last (default), min, max, or sum of the contained points.

**Convert nodata to zero [***[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/data_tools.html?highlight=convertnoda#convertnodatatozero)***]**

This tool can be used to change the value within the grid cells of a raster file (--input) that contain NoData to zero. The most common reason for using this tool is to change the background region of a raster image such that it can be included in analysis since NoData values are usually ignored by most tools. This change, however, will result in the background no longer displaying transparently in most GIS. This change can be reversed using the SetNoDataValue tool.

**Euclidean distance [***[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/gis_analysis_distance_tools.html?highlight=euclidean#euclideandistance)***]**

This tool will estimate the Euclidean distance (i.e., straight-line distance) between each grid cell and the nearest 'target cell' in the input image. Target cells are all non-zero, non-NoData grid cells. Distance in the output image is measured in the same units as the horizontal units of the input image.

Algorithm description:

The algorithm is based on the highly efficient distance transform of Shih and Wu (2003). It makes four passes of the image; the first pass initializes the output image; the second and third passes calculate the minimum squared Euclidean distance by examining the 3 x 3 neighbourhood surrounding each cell; the last pass takes the square root of cell values, transforming them into true Euclidean distances, and deals with NoData values that may be present. All NoData value grid cells in the input image will contain NoData values in the output image. As such, NoData is not a suitable background value for non-target cells. Background areas should be designated with zero values.

## Distance from lines

**Outline:**

it creates a raster showing distances between input lines.

**Source:** t

**Description:**

This tool is composed of two whitebox tools with the following workflow:

1. vector\_lines\_to\_raster()
2. convert\_nodata\_to\_zero()
3. euclidean\_distance()

**Vector lines to raster [***[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/data_tools.html?highlight=vectorlines#vectorlinestoraster)***]**

This tool can be used to convert a vector lines or polygon file into a raster grid of lines. If a vector of one of the polygon ShapeTypes is selected, the resulting raster will outline the polygons without filling these features. Use the VectorPolygonToRaster tool if you need to fill the polygon features.

The user must specify the name of the input vector (--input) and the output raster file (--output). The Field Name (--field) is the field from the attributes table, from which the tool will retrieve the information to assign to grid cells in the output raster. Note that if this field contains numerical data with no decimals, the output raster data type will be INTEGER; if it contains decimals, it will be of a FLOAT data type. The field must contain numerical data. If the user does not supply a Field Name parameter, each feature in the raster will be assigned the record number of the feature. The assignment operation determines how the situation of multiple points contained within the same grid cell is handled. The background value is the value that is assigned to grid cells in the output raster that do not correspond to the location of any points in the input vector. This value can be any numerical value (e.g., 0) or the string 'NoData', which is the default.

If the user optionally specifies the --cell\_size parameter, then the coordinates will be determined by the input vector (i.e., the bounding box) and the specified Cell Size. This will also determine the number of rows and columns in the output raster. If the user instead specifies the optional base raster file parameter (--base), the output raster's coordinates (i.e., north, south, east, west) and row and column count will be the same as the base file. If the user does not specify either of these two optional parameters, the tool will determine the cell size automatically as the maximum of the north-south extent (determined from the shapefile's bounding box) or the east-west extent divided by 500.

**Convert nodata to zero**

See [Convert nodata to zero](#convert_nodata_to_zero)

**Euclidean distance**

See [Euclidean distance](#euclidean_distance)

## Distance from polygons

**Outline:** it creates a raster showing distances between input polygons.

**Source:** this tool is part of *wbt\_geometric\_analysis.py.*

**Description:**

This tool is composed of two whitebox tools with the following workflow:

1. vector\_polygons\_to\_raster()
2. convert\_nodata\_to\_zero()
3. euclidean\_distance()

**Vector polygons to raster [***[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/data_tools.html?highlight=vectorpol" \l "vectorpolygonstoraster)***]**

Converts a vector containing polygons into a raster.

**Convert nodata to zero**

See [Convert nodata to zero](#convert_nodata_to_zero)

**Euclidean distance**

See [Euclidean distance](#_Euclidean_distance)

## Distance from raster

**Outline:** it creates a raster showing distances within raster.

**Source:** this tool is part of *wbt\_geometric\_analysis.py.*

**Description:**

This tool is composed of one whitebox tool with the following workflow:

1. convert\_nodata\_to\_zero()
2. euclidean\_distance()

**Convert nodata to zero**

See [Convert nodata to zero](#convert_nodata_to_zero)

**Euclidean distance**

See [Euclidean distance](#euclidean_distance)

## Hotspots from points

**Outline:**

it creates a hotspot raster from an input point shapefile.

**Source:** t

**Description:**

This tool is composed of two whitebox tools and has the following workflow:

1. vector\_points\_to\_raster()
2. convert\_nodata\_to\_zero()
3. gaussian\_filter()

**Vector points to raster**

See [Vector points to raster](#vector_points_to_raster)

**Convert nodata to zero**

See [Convert nodata to zero](#convert_nodata_to_zero)

**Gaussian filter [***[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/image_processing_tools_filters.html?highlight=gaus#gaussianfilter)***]**

This tool can be used to perform a Gaussian filter on a raster image. A Gaussian filter can be used to emphasize the longer-range variability in an image, effectively acting to smooth the image. This can be useful for reducing the noise in an image. The algorithm operates by convolving a kernel of weights with each grid cell and its neighbours in an image. The weights of the convolution kernel are determined by the 2-dimensional Gaussian (i.e., normal) curve, which gives stronger weighting to cells nearer the kernel centre. It is this characteristic that makes the Gaussian filter an attractive alternative for image smoothing and noise reduction than the MeanFilter. The size of the filter is determined by setting the standard deviation parameter (--sigma), which is in units of grid cells; the larger the standard deviation the larger the resulting filter kernel. The standard deviation can be any number in the range 0.5-20.

GaussianFilter works with both greyscale and red-green-blue (RGB) colour images. RGB images are decomposed into intensity-hue-saturation (IHS) and the filter is applied to the intensity channel. NoData values in the input image are ignored during processing.

Like many low-pass filters, Gaussian filtering can significantly blur well-defined edges in the input image. The EdgePreservingMeanFilter and BilateralFilter offer more robust feature preservation during image smoothing. GaussianFilter is relatively slow compared to the FastAlmostGaussianFilter tool, which offers a fast-running approximatation to a Gaussian filter for larger kernel sizes.

## Hotspots from lines

**Outline:**

it creates a hotspot raster from an input line shapefile.

**Source:** t

**Description:**

This tool is composed of two whitebox tools and has the following workflow:

1. vector\_lines\_to\_raster()
2. convert\_nodata\_to\_zero()
3. gaussian\_filter()

**Vector lines to raster**

See [Vector lines to raster](#vector_lines_to_raster)

**Convert nodata to zero**

See [Convert nodata to zero](#convert_nodata_to_zero)

## Gaussian filter

See [Gaussian filter](#gaussian_filter)

## Hotspots from polygons

**Outline:**

it creates a hotspot raster from an input polygons shapefile.

**Source:** t

**Description:**

This tool is composed of two whitebox tools and has the following workflow:

1. vector\_polygons\_to\_raster()
2. convert\_nodata\_to\_zero()
3. gaussian\_filter()

**Vector polygon to raster**

See [Vector polygon to raster](#_Vector_polygons_to)

**Convert nodata to zero**

See [Convert nodata to zero](#convert_nodata_to_zero)

**Gaussian filter**

See [Gaussian filter](#gaussian_filter)

## Hotspots from raster

**Outline:**

it creates a hotspot raster from an input raster.

**Source:** t

**Description:**

This tool is composed of one whitebox tool and has the following workflow:

1. convert\_nodata\_to\_zero()
2. gaussain\_filter()

**Gaussian filter**

See [Gaussian filter](#gaussian_filter)

## Interpolate points

**Outline:**

it interpolates point into a raster surface.

**Source:** t

**Description:**

This tool is composed of one whitebox tool and has the following workflow:

1. radial\_basis\_function\_interpolation()

**Radial basis function interpolation [***[link](https://www.whiteboxgeo.com/manual/wbt_book/available_tools/gis_analysis.html?highlight=radial#radialbasisfunctioninterpolation)***]**

This tool interpolates vector points into a raster surface using a radial basis function (RBF) scheme.

## Summarize within

**Outline:**

it summarizes vector data relative to existing polygons.

**Source:** t

**Description:**

This tool first uses geopandas to perform a join between the vector data and polygons. From there it uses a dissolve to aggregate the joined shapefile based on a specified field and has the following workflow:

1. input\_vector\_gdf.join()
2. input\_vector\_join.drop()
3. input\_vector\_join.dissolve()

**Summarize within [***[link](https://geopandas.org/en/stable/docs/user_guide/aggregation_with_dissolve.html)***]**

Spatial data are often more granular than we need. For example, we might have data on sub-national units, but we’re actually interested in studying patterns at the level of countries.

In a non-spatial setting, when all we need are summary statistics of the data, we aggregate our data using the groupby() function. But for spatial data, we sometimes also need to aggregate geometric features. In the *geopandas* library, we can aggregate geometric features using the dissolve() function.

# Arcpy Tools

## Zonal statistics

**Outline:**

it calculates statistics on values of a raster within the zones of another dataset.

**Source:** this

**Description:**

This tool is composed of one arcpy tool and has the following workflow:

1. ZonalStatistics(in\_zone\_data, zone\_field, in\_value\_raster, {statistics\_type})

**Zonal statistics [***[link](https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/zonal-statistics.htm)***]**

This tool calculates a statistic for each zone define by a zone dataset, based on values from another dataset (a value raster). A single output value is computed for every zone in the input zone dataset.

The Zonal Statistics tool, a statistic is calculated for each zone defined by a zone dataset, based on values from another dataset (a value raster). A single output value is computed for every zone in the input zone dataset.

A zone is all the cells in a raster that have the same value, whether or not they are contiguous. The input zone layer defines the shape, values, and location of the zones. An integer field in the zone input is specified to define the zones. A string field can also be used. Both raster and feature datasets can be used as the zone dataset.

The input value raster contains the input values used in calculating the output statistic for each zone.

## Summarize within

**Outline:**

it overlays a polygon layer with another layer to summarize the number of points, length of the lines, or area of the polygons within each polygon, and calculate attribute field statistics about those features within the polygons.

**Source:** this

**Description:**

This tool is composed of one arcpy tool and has the following workflow:

1. arcpy.analysis.SummarizeWithin(in\_polygons, in\_sum\_features, out\_feature\_class, {keep\_all\_polygons}, {sum\_fields})

**Summarize within [***[link](https://pro.arcgis.com/en/pro-app/latest/tool-reference/analysis/summarize-within.htm)***]**

This tool works by using two input features and stacking them on top of each other. After stacking these layers, you peer down through the stack and count the number input summary features that fall within the input polygons. Not only can you count the number of features, but you can also calculate simple statistics about the attributes of the input summary features, such as sum, mean, minimum, maximum, and so on.

## Calculate proximity

**Outline:**

it calculates, for each cell, the Euclidean distance to the closest source.

**Source:** tis

**Description:**

This tool is composed of one arcpy tool and has the following workflow:

1. EucDistance(inSourceData, maxDistance, cellSize, outDirectionRaster)

**Euclidean distance [***[link](https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/euclidean-distance.htm)***]**

This tool works with both feature class and raster. So, the input source data can be a feature class or raster.

When the input source data is a raster, the set of source cells consists of all cells in the source raster that have valid values. Cells that have NoData values are not included in the source set. The value 0 is considered a legitimate source. A source raster can be created using the extraction tools. When the input source data is a feature class, the source locations are converted to a raster internally before performing the analysis.

The Maximum distance is specified in the same map units as the input source data. The Output cell size can be defined by a numeric value or obtained from an existing raster dataset. If the cell size hasn’t been explicitly specified as the parameter value, it is derived from the Cell Size environment if it has been specified. If the parameter cell size or the environment cell size have not been specified, the default output cell size is determined based on the type of input dataset as follows:

* If the input data is a raster, the cell size of the dataset is used.
* If the input dataset is a feature and the Snap Raster environment has been set, the cell size of the snap raster is used. If no snap raster is set, the cell size is calculated from the shorter of the width or height of the extent divided by 250, in which the extent is in the Output Coordinate System specified in the environment.

## Calculate hotspots

**Outline:**

it calculates hotspots based on a layer of points using a kernel function to fit a smoothly tapered surface.

**Source:** tis

**Description:**

This tool is composed of following arcpy tools and has the following workflow:

1. KernelDensity(in\_features, population\_field, cell\_size = 500)
2. ExtractByMask(outKernelDensity, masking\_polygon)
3. arcpy.RasterToNumPyArray(outExtractByMask)
4. numpy.percentile(array, q)
5. Reclassify(outExtractByMask, "Value",

RemapRange([[p[0],p[1],1],

[p[1],p[2],2],

[p[2],p[3],3],

[p[3],p[4],4],

[p[4],p[5],5], ]))

**Kernel density [***[link](https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/kernel-density.htm)***]**

This tool calculates a magnitude-per-unit area from point or polyline features using a kernel function to fit a smoothly tapered surface to each point or polyline. Its usage is summarized below.

* Larger values of the search radius parameter produce a smoother, more generalized density raster. Smaller values produce a raster that shows more detail.
* Only the points or portions of a line that fall within the neighbourhood are considered in calculating density. If no points or line sections fall within the neighbourhood of a particular cell, that cell is assigned NoData.
* If the area unit scale factor units are small relative to the features (distance between points or length of line sections, depending on feature type), the output values may be small. To obtain larger values, select the area unit scale factor for larger units (for example, square kilometres versus square meters).
* The default search radius (bandwidth) is calculated based on the spatial configuration and number of input points. This approach corrects for spatial outliers—input points that are very far away from the rest—so they will not make the search radius unreasonably large.
* Very large or very small values in the Population field can give results that may seem unintuitive. If the mean of the population field is much bigger than 1 (for example, as with city populations), the default search radius might be very small, resulting in small rings around the input points. If the mean of the population field is much smaller than 1, the calculated bandwidth might seem unreasonably large. In these cases, you may want to enter your own search radius.

**Extract by mask [***[link](https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/extract-by-mask.htm)***]**

This tool extracts the cells of a raster that correspond to the areas defined by a mask. Its usage is summarized below.

* The result of using the Extract by Mask tool is similar to that of setting the Mask environment, except that the input mask is only used on the immediate instance, while a mask set in the environment is applied to all tools until it is changed or disabled.
* When a multiband raster is specified as input, a new multiband raster will be created as output. Each individual band in the input multiband raster will be analysed accordingly.
* If the input is a layer created from a multiband raster with more than three bands, the extraction operation will only consider the bands that were loaded (symbolized) by the layer. As a result, the output multiband raster can only have three bands, corresponding to those used in the display of the input layer.
* When a multiband raster is specified for the input raster mask, only the first band will be used in the operation.
* If the input raster is integer, the output raster will be integer. If the input is floating point, the output will be floating point.

**Raster to numpy array [***[link](https://pro.arcgis.com/en/pro-app/latest/arcpy/functions/rastertonumpyarray-function.htm)***]**

Converts a raster to a NumPy array. A NumPy array is designed to deal with large arrays. If the array definition (the lower left corner and the number of rows and columns) exceeds the extent of the **in\_raster**, the array values will be assigned NoData.

If the **lower\_left\_corner** does not coincide with the corner of a cell, it will automatically be snapped to the lower left of the nearest cell corner applying the same rules as the Snap Raster environment setting. This snapping action within the **RasterToNumPy** function is not to be confused with the Snap Raster environment setting; the function only uses the same interaction.

**NumPy percentile [***[link](https://numpy.org/doc/stable/reference/generated/numpy.percentile.html)***]**

Computes the q-th percentile of the data along the specified axis.

Returns the q-th percentile(s) of the array elements.

**Reclassify [***[link](https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/reclassify.htm)***]**

This tool reclassifies (or changes) the values in a raster. Its usage is summarized below.

If a range of values is to be reclassed, the ranges should not overlap except at the boundary of two input ranges. Where overlapping occurs, the higher end of the lower input range is inclusive, and the lower end of the higher input range is exclusive.

If the input raster has an attribute table, it will be used to create the initial reclassification table. If the input raster does not have an attribute table, you can run the Build Raster Attribute Table tool from the Data Management toolbox to build one before inputting the raster into the Reclassify tool. Otherwise, when you input the raster, a reclassification table will be created for it by first applying geoprocessing environment settings, such as Extent and Cell size, and scanning the raster.

When the input raster is a layer from **Contents**, the default reclassification table will import the unique values or classified break values as specified by the layer symbology. The current geoprocessing environment settings will be ignored when importing those values. Otherwise, the reclassification must be manually entered or generated using the unique or classification options.

## Buffer vector

**Outline:**

it buffers vector features by specified distance.

**Source:** this

**Description:**

This tool is composed of one arcpy tool and has the following workflow:

1. arcpy.analysis.Buffer(in\_features, out\_feature\_class, buffer\_distance\_or\_field)

**Buffer [***[link](https://pro.arcgis.com/en/pro-app/latest/tool-reference/analysis/buffer.htm)***]**

The buffer routine traverses each of the input feature's vertices and creates buffer offsets. Output buffer features are created from those offsets. The buffer distance parameter can be entered as a fixed value or as a field containing numeric values.

## Intersect

**Outline:**

it calculates the geometric intersection of any number of feature classes and feature layers.

**Source:** tis

**Description:**

This tool is composed of one arcpy tool and has the following workflow:

1. arcpy.Intersect\_analysis(in\_features, out\_feature\_class)

**Intersect [***[link](https://pro.arcgis.com/en/pro-app/latest/tool-reference/analysis/intersect.htm)***]**

This tool computes a geometric intersection of the input features. Features or portions of features that overlap in all layers or feature classes will be written to the output feature class. The Input Features parameter value must be simple features: point, multipoint, line, or polygon. They cannot be complex features such as annotation features, dimension features, or network features.

If the inputs have different geometry types (that is, line on poly, point on line, and so on), the Output Feature Class geometry type will default to be the same as the Input Features geometry type with the lowest dimension geometry. For example, if one or more of the inputs is of type point, the default output will be point; if one or more of the inputs is line, the default output will be line; and if all inputs are polygon, the default output will be polygon.

The Output Type parameter value can be that of the Input Features parameter with the lowest dimension geometry or lower. For example, if all the inputs are polygons, the output can be polygon, line, or point. If one of the inputs is of type line and none are points, the output can be line or point. If any of the inputs are point, the Output Type value can only be point.

## Calculate slope

**Outline:**

it creates a slope raster based on an input elevation raster (DEM).

**Source:** this

**Description:**

This tool is composed of one arcpy tool and has the following workflow:

1. Slope(ElevationRaster)

**Slope [***[link](https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/slope.htm)***]**

This tool identifies the slope (gradient or steepness) from each cell of a raster. For each cell, the Slope tool calculates the maximum rate of change in value from that cell to its neighbours. Basically, the maximum change in elevation over the distance between the cell and its eight neighbours identifies the steepest downhill descent from the cell.

Conceptually, the tool fits a plane to the z-values of a 3 x 3 cell neighbourhood around the processing or centre cell. The slope value of this plane is calculated using the average maximum technique. The direction the plane faces is the aspect for the processing cell. The lower the slope value, the flatter the terrain; the higher the slope value, the steeper the terrain.

If there is a cell location in the neighbourhood with a NoData z-value, the z-value of the centre cell will be assigned to the location. At the edge of the raster, at least three cells (outside the raster's extent) will contain NoData as their z-values. These cells will be assigned the centre cell's z-value. The result is a flattening of the 3 x 3 plane fitted to these edge cells, which usually leads to a reduction in the slope.

## Calculate aspect

**Outline**:itcreates an aspect raster based on an input elevation raster (DEM).

**Source**:this

**Description:**

This tool is composed of one arcpy tool and has the following workflow:

1. Aspect(ElevationRaster)

**Aspect [***[link](https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/aspect.htm)***]**

Aspect identifies the downslope direction of the maximum rate of change in value from each cell to its neighbours. Aspect can be thought of as the slope direction. The values of the output raster are the compass direction of the aspect.

Diagram

Description automatically generated with low confidence

Aspect identifies the downslope direction of the maximum rate of change in value from each cell to its neighbours. It can be thought of as the slope direction. The values of each cell in the output raster indicate the compass direction that the surface faces at that location. It is measured clockwise in degrees from 0 (due north) to 360 (again due north), coming full circle. Flat areas having no downslope direction are given a value of -1. Conceptually, the Aspect tool fits a plane to the z-values of a 3 x 3 cell neighbourhood around the processing or centre cell. The direction the plane faces is the aspect for the processing cell.

## Steep areas

**Outline**: it

calculates steep areas by creating mask of areas with slope equal or higher than a threshold.

**Source**:this

**Description:**

This tool is composed of following arcpy tools and has the following workflow:

1. outSlope = Slope(ElevationRaster)
2. SetNull(outSlope < threshold, 1)
3. arcpy.conversion.RasterToPolygon()
4. arcpy.management.Dissolve()

**Slope**

See [slope](#slope_arcpy) calculate slope.

**Set null [***[link](https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/set-null.htm)***]**

Set Null sets identified cell locations to NoData based on a specified criteria. It returns NoData if a conditional evaluation is true, and returns the value specified by another raster if it is false. The usage for this tool is summarised below.

If the evaluation of the where clause is true, the cell location on the output raster will be assigned NoData. If the evaluation is false, the output raster will be defined by the input false raster or constant value.

If no where clause is specified, the output raster will have NoData wherever the conditional raster is not 0.

The input conditional raster does not affect whether the output data type is integer or floating point. If the input false raster (or constant value) contains floating-point values, the output raster will be floating point. If it contains all integer values, the output will be an integer raster.

**Raster to polygon [***[link](https://pro.arcgis.com/en/pro-app/latest/tool-reference/conversion/raster-to-polygon.htm)***]**

This tool converts a raster dataset to polygon features.

**Dissolve [***[link](https://pro.arcgis.com/en/pro-app/latest/tool-reference/data-management/dissolve.htm)***]**

This tool aggregates features based on specified attributes.

A picture containing diagram

Description automatically generated

## Calculate fill

**Outline:** it calculates filled digital elevation model raster.

**Source:** this tool is part of hydrological\_analysis.py.

**Description:**

This tool is composed of one arcpy tool and has the following workflow:

1. Fill(inSurfaceRaster, zLimit)

**Fill [***[link](https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/fill.htm)***]**

Fills sinks in a surface raster to remove small imperfections in the data. Its usage is summarized below.

A sink is a cell with an undefined drainage direction; no cells surrounding it are lower. The pour point is the boundary cell with the lowest elevation for the contributing area of a sink. If the sink were filled with water, this is the point where water would pour out.

The z-limit specifies the maximum difference allowed between the depth of a sink and the pour point and determines which sinks will be filled and which will remain untouched. The z-limit is not the maximum depth to which a sink will be filled.

All sinks that are less than the z-limit, and lower than their lowest adjacent neighbours, will be filled to the height of their pour points.

## Calculate flow direction

**Outline:** it calculates the flow direction raster based on the input filled raster.

**Source:** this

**Description:**

This tool is composed of one arcpy tool and has the following workflow:

1. FlowDirection(inFillRaster, "NORMAL")

**Flow direction [***[link](https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/flow-direction.htm)***]**

This tool creates a raster of flow direction from each cell to its downslope neighbour, or neighbours, using the D8, Multiple Flow Direction (MFD), or D-Infinity (DINF) method. Its usage is summarized below.

The D8 flow method models flow direction from each cell to its steepest downslope neighbour.

The output of the Flow Direction tool run with the D8 flow direction type is an integer raster whose values range from 1 to 255. The values for each direction from the centre are the following:

Table

Description automatically generated

For example, if the direction of steepest drop was to the left of the current processing cell, its flow direction would be coded as 16.

* If a cell is lower than its eight neighbours, that cell is given the value of its lowest neighbour, and flow is defined toward this cell. If multiple neighbours have the lowest value, the cell is still given this value, but flow is defined with one of the two methods explained below. This is used to filter out one-cell sinks, which are considered noise.
* If a cell has the same change in z-value in multiple directions and that cell is part of a sink, the flow direction is referred to as undefined. In such cases, the value for that cell in the output flow direction raster will be the sum of those directions. For example, if the change in z-value is the same both to the right (flow direction = 1) and down (flow direction = 4), the flow direction for that cell is 1 + 4 = 5. Cells with undefined flow direction can be flagged as sinks using the Sink tool.
* If a cell has the same change in z-value in multiple directions and is not part of a sink, the flow direction is assigned with a lookup table defining the most likely direction.
* The output D8 drop raster is calculated as the difference in z-value divided by the path length between the cell centres, expressed in percentages. For adjacent cells, this is analogous to the percent slope between cells. Across a flat area, the distance becomes the distance to the nearest cell of lower elevation. The result is a map of percent rise in the path of steepest descent from each cell.

## Calculate flow accumulation

**Outline:**

it calculates the flow accumulation raster based on the input filled raster.

**Source:** tis

**Description:**

This tool is composed of one arcpy tool and has the following workflow:

1. FlowAccumulation(inFlowDirection)

**Flow accumulation [***[link](https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/flow-accumulation.htm)***]**

This tool creates a raster of accumulated flow into each cell. Its usage is summarized below.

* The result of Flow Accumulation is a raster of accumulated flow to each cell, as determined by accumulating the weight for all cells that flow into each downslope cell.
* The Flow Accumulation tool supports three flow modelling algorithms while computing accumulated flow. These are D8, Multiple Flow Direction (MFD) and D-Infinity (DINF) flow methods.
* Cells of undefined flow direction will only receive flow; they will not contribute to any downstream flow.

For an input D8 flow direction raster, a cell is considered to have an undefined flow direction if its value in the flow direction raster is anything other than 1, 2, 4, 8, 16, 32, 64, or 128.

* The accumulated flow is based on the number of total or a fraction of cells flowing into each cell in the output raster. The current processing cell is not considered in this accumulation.
* Output cells with a high flow accumulation are areas of concentrated flow and can be used to identify stream channels.
* Output cells with a flow accumulation of zero are local topographic highs and can be used to identify ridges.

## Calculate flow network

**Outline:**

it calculates the flow network raster based on the input accumulation raster with a defined threshold.

**Source:** this

**Description:**

This tool is composed of one arcpy tool and has the following workflow:

1. SetNull(inFlowAccumulation < flow\_acc\_threshold, 1)

**Set null**

See [set null](#SetNull) in steep areas

## Complete hydrological routing

**Outline:** it calculates the complete hydrological routing routine, which generates multiple outputs including both raster and vector.

**Source:** this tool is part of hydrological\_analysis.py.

**Description:**

This tool is composed of following arcpy tools and has the following workflow:

1. outFill = Fill(inSurfaceRaster)
2. outFlowDirection = FlowDirection(outFill, "NORMAL")
3. outFlowAccumulation = FlowAccumulation(outFlowDirection)
4. outFlowAccumulationNetwork = SetNull(outFlowAccumulation < flow\_acc\_threshold, 1)
5. outBasins = Basin(outFlowDirection)
6. outFillDiff = SetNull((outFill - inSurfaceRaster) == 0, 1)
7. arcpy.RasterToPolygon\_conversion(outFillDiff, outPolygons, "NO\_SIMPLIFY")
8. arcpy.RasterToPolygon\_conversion(outFlowAccumulationNetwork, outPolygons, "NO\_SIMPLIFY")
9. arcpy.RasterToPolygon\_conversion(outBasins, outPolygons, "NO\_SIMPLIFY")

**Fill**

See [fill](#fill_arcpy) in calculate fill.

**Flow direction**

See [flow direction](#flow_direction_arcpy) in calculate flow direction*.*

**Flow accumulation**

See [flow accumulation](#flow_accumulation_arcpy) in calculate flow accumulation*.*

**Set null**

See [set null](#SetNull) in steep areas.

**Basin [***[link](https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-analyst/basin.htm)***]**

This tool creates a raster delineating all drainage basins. Its usage is summarized below.

* The drainage basins are delineated within the analysis window by identifying ridge lines between basins. The input flow direction raster is analysed to find all sets of connected cells that belong to the same drainage basin. The drainage basins are created by locating the pour points at the edges of the analysis window (where water would pour out of the raster), as well as sinks, then identifying the contributing area above each pour point. This results in a raster of drainage basins.
* To create the input D8 flow direction raster, the Flow Direction tool must be run using the default flow direction type D8.

The best results will be obtained if when the input D8 Flow Direction raster was created, the Force all edge cells to flow outward option (FORCE in Python) was enabled.

* All cells in the raster will belong to a basin, even if that basin is only one cell.

**Raster to polygon**

See [raster to polygon](#Raster_to_polygon) in steep areas.

## Set null below

**Outline:** it creates constant valued raster based on SetNull operator below threshold.

**Source:** this tool is part of conditional\_eval.py.

**Description:**

This tool is composed of one arcpy tool and has the following workflow:

1. SetNull(inRaster < th, inRaster)

**Set null**

See [set null](#SetNull) in steep areas.

## Set null between

**Outline:**

it creates constant valued raster based on two SetNull operations.

**Source:** this

**Description:**

This tool is composed of following arcpy tool and has the following workflow:

1. set\_null\_between(inRaster, low\_th, high\_th)

**Set null**

See [set null](#SetNull) in steep areas.

## Set null above

**Outline:**

it creates constant valued raster based on SetNull operator above threshold.

**Source:** t

**Description:**

This tool is composed of one arcpy tool and has the following workflow:

1. SetNull(inRaster > th, inRaster)

**Set null**

See [set null](#SetNull) in steep areas.

## Inundation extents

**Outline:**

it creates a set of inundation extents based on elevation thresholds and exports them as polygons and/or rasters.

**Source:** t

**Description**

This tool is composed of following arcpy tools and has the following workflow:

1. set\_null\_above(ElevationRaster,th)
2. set\_null\_between(ElevationRaster, low\_th, high\_th)
3. arcpy.conversion.RasterToPolygon(outRaster, out\_polygon\_features)

**Set null above**

See [set null above](#SetNull_above) in set null above.

**Set null between**

See [set null between](#SetNull_between) in set null between.

**Raster to polygon**

See [raster to polygon](#Raster_to_polygon) in steep areas.