Near (Analysis)

ArcGIS 10.5

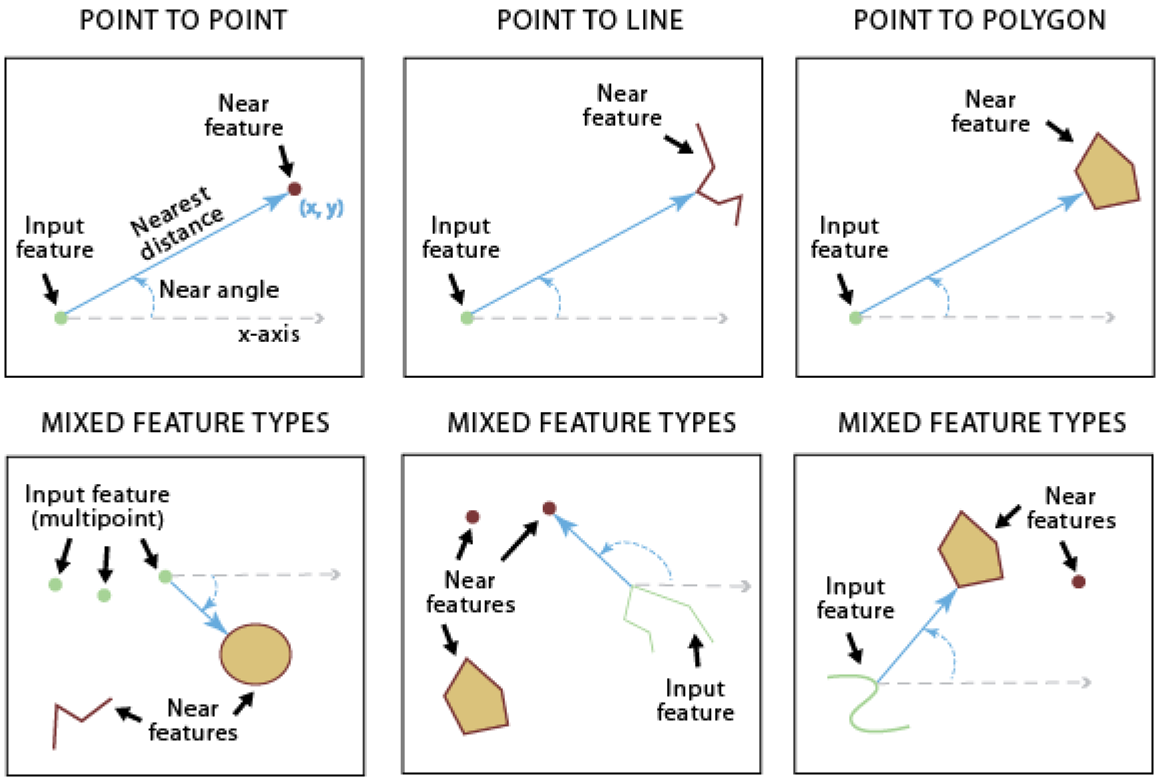
License Level: Basic Standard Advanced Locate topic

Summary

Calculates distance and additional proximity information between the input features and the closest feature in another layer or feature class.

Learn more about how proximity is calculated in geoprocessing tools

Illustration



Usage

- The following fields will be added to the input. If the fields already exist, the field values are updated.

Field Name	Description
`NEAR_FID`	The Object ID of the closest near feature. If no near feature is found, the value will be -1.
`NEAR_DIST`	The distance between the input and near feature. The value is in the linear unit of the input features coordinate system, or Meters when the **Method** parameter is set to GEODESIC and the input is in a geographic coordinate system. If no near feature is found, the value will be -1.
`NEAR_FC`	The catalog path to the feature class that contains the near feature. This field is only added to the output table if multiple **Near Features** are specified. If no near feature is found, the value will be empty string or null.

The following fields will be added to the input features if the **Location** parameter is checked (location parameter set to LOCATION in Python). The field values are updated if the fields already exist. The unit of the values of the fields depend what method is selected on the **Method** parameter. If set to **Planar**, the value is in the linear unit of the input feature's coordinate system. If set to **Geodesic**, the value is in the geographic coordinate system associated with the input feature's coordinate system.

Field Name	Description
NEAR_X	X-coordinate of the location on the near feature which is closest to the input feature. If no near feature is found, the value will be -1.
NEAR_Y	Y-coordinate of the location on the near feature which is closest to the input feature. If no near feature is found, the value will be -1.

The following field will be added to the input features if the **Angle** parameter is checked (angle parameter set to ANGLE in Python). The field values are updated if the fields already exist.

Field	Description
NEAR_ANGLE	Angle of the line at the FROM_X and FROM_Y location which connects the input features to the near feature. If no near feature is found or the near feature intersects the input feature, the value will be 0.

- The values for NEAR_FID and NEAR_DIST will be -1 if no feature is found within the search radius.
- Both input features and near features can be point, multipoint, line, or polygon.
- The **Near Features** can include one or more feature classes of different shape types (point, multipoint, line, or polygon).
- The same feature class or layer may be used as both input and near features. In this situation, the input feature being evaluated is excluded from the near features candidates to avoid all features being closest to themselves.
- The input features can be a layer on which you have performed a selection. The selected features will be used and updated during the execution of the tool. The remaining features will have the values of the newly created fields (such as NEAR_FID and NEAR_DIST) set to -1.
- When more than one near feature has the same shortest distance from an input feature, one of them is randomly chosen as the nearest feature.
- When using the **Planar** option for the **Method** parameter, the input features should be in a projection that is appropriate for distance measurement, such as an equidistant projection.
  Learn more about geographic and projected coordinate systems
- To visualize the FROM_X, FROM_Y, NEAR_X, and NEAR_Y locations, the output table can be used as input to the Make XY Event Layer or XY To Line tools.

Syntax

Near_analysis (in_features, near_features, {search_radius}, {location}, {angle}, {method})

Parameter	Explanation	Data Type
in_features	The input features that can be point, polyline, polygon, or multipoint type.	Feature Layer

near_features [near_features,...]	One or more feature layers or feature classes containing near feature candidates. The near features can be of point, polyline, polygon, or multipoint. If multiple layers or feature classes are specified, a field named `NEAR_FC` is added to the input table and will store the paths of the source feature class containing the nearest feature found. The same feature class or layer may be used as both input and near features.	Feature Layer
search_radius (Optional)	The radius used to search for near features. If no value is specified, all near features are considered. If a distance but no unit or unknown is specified, the units of the coordinate system of the input features are used. If the **Geodesic** option is used, a linear unit such as Kilometers or Miles should be used.	Linear unit
location (Optional)	Specifies whether x- and y-coordinates of the closest location of the near feature will be written to the `NEAR_X` and `NEAR_Y` fields. ● NO_LOCATION — Location information will not be written to the output table. This is the default. ● LOCATION — Location information will be written to the output table.	Boolean
angle (Optional)	Specifies whether the near angle will be calculated and written to a `NEAR_ANGLE` field in the output table. A near angle measures direction of the line connecting an input feature to its nearest feature at their closest locations. When the `PLANAR` method is used in the `method` parameter, the angle is within the range of -180° to 180°, with 0° to the east, 90° to the north, 180° (or -180°) to the west, and -90° to the south. When the `GEODESIC` method is used, the angle is within the range of -180° to 180°, with 0° to the north, 90° to the east, 180° (or -180°) to the south, and -90° to the west. ● NO_ANGLE —The near angle values will not be written. This is the default. ● ANGLE —The near angle values will be written to the `NEAR_ANGLE` field.	Boolean
method (Optional)	Determines whether to use a shortest path on a spheroid (geodesic) or a flat earth (planar) method. It is strongly suggested to use the **Geodesic** method with data stored in a coordinate system that is not appropriate for distance measurements (for example, Web Mercator or any geographic coordinate system) and any analysis that spans a large geographic area. ● PLANAR —Uses planar distances between the features. This is the default. ● GEODESIC —Uses geodesic distances between features. This method takes into account the curvature of the spheroid and correctly deals with data near the dateline and poles.	String

Code sample

Near example 1 (Python window)

The following Python interactive window script demonstrates how to use the Near function in immediate mode.

```
import arcpy
arcpy.env.workspace = "C:/data/city.gdb"

## find the nearest road from each house
arcpy.Near_analysis('houses', 'roads')
```

Near example 2 (stand-alone Python script)

The following Python script demonstrates how to use the Near function in a stand-alone script.

```
# Name: Near.py
# Description: Finds nearest features from input feature class to n

import arcpy

# Set workspace environment
arcpy.env.workspace = "C:/data/city.gdb"

try:
    # set local variables
    in_features = "houses"
    near_features = "parks"

    # find features only within search radius
    search_radius = "5000 Meters"

    # find location nearest features
    location = "LOCATION"

    # avoid getting angle of neares features
    angle = "NO_ANGLE"

    # execute the function
    arcpy.Near_analysis(in_features, near_features, search_radius,

    # get geoprocessing messages
    print(arcpy.GetMessages())

except arcpy.ExecuteError:
    print(arcpy.GetMessages(2))

except Exception as err:
    print(err.args[0])
```

Near example 3 (stand-alone Python script)

The following Python script demonstrate how to convert the near angle to azimuth.

```
# Name: near_angle_to_azimuth.py

import arcpy

# Near tool does not calculate angle in azimuths
# This script, using the output of the tool, converts the angles
# to azimuth angles.

in_table = r"C:/data/city.gdb/near_table"

angles = arcpy.da.SearchCursor(in_table, ["NEAR_ANGLE"])

azimuth_angles = []

with angles as rows:
    for row in rows:
        angle = row[0]
        if angle <= 180 and angle > 90:
            azimuth_angles.append(360.0 - (angle - 90))
        else:
            azimuth_angles.append(abs(angle - 90))

# Use these azimuth angles as necessary.
```

Near example 4 (stand-alone Python script)

Demonstrates postprocessing with the Near tool's derived output. The script finds, for each near feature, which input features it is closest to.

```
# Name: features_closest_to_input.py

"""
    This script finds, for each input feature, a list of near featu
    If near features 6, 7, 10 are closest to input feature 3 then t
    resulting dictionary will have a list [6, 7, 10] as value for k
"""

import os
import arcpy

in_fc = r"C:\data\cities.gdb\cities_many"

# create a dictionary to hold the list of near ids for each input i
nearest_dict = dict()

with arcpy.da.SearchCursor(in_fc, ["OID@", "NEAR_FID"]) as rows:
    for row in rows:
        nearest_id = row[0]  # get OID value for that row
        input_id = row[1]    # get NEAR_FID value

        if input_id in nearest_dict:
            # if a dict key already exists, append near id to value
            nearest_dict[input_id].append(nearest_id)
        else:
            # if the key does not exist then create a new list with
            # and add it to the dictionary
            nearest_dict[input_id] = [nearest_id]

print(nearest_dict)

# output will look like:
# {1: [13, 28], 2: [2, 9, 14, 20, 22], 3: [11, 12, 24, 25]}
```

Environments

Extent, Current Workspace

Licensing information

ArcGIS Desktop Basic: No

ArcGIS Desktop Standard: No

ArcGIS Desktop Advanced: Yes

Related topics

An overview of the Proximity toolset