Correcting topological errors using SSNbler and ArcGIS Pro

David Nagel and Erin E Peterson

May 2, 2024

Contents

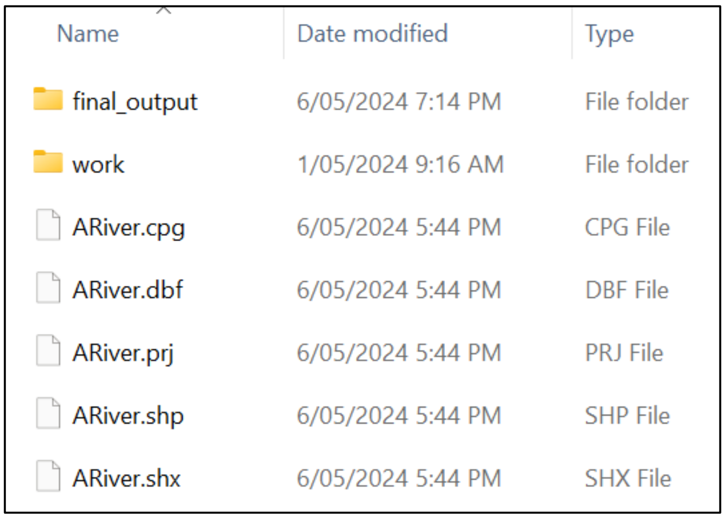
1. Introduction

To generate a valid landscape network (LSN) with the SSNbler R package, users must ensure their digitised streams are free of topological errors and topological restrictions not permitted in SSN2. Vector streams data are sometimes prone to topological errors, even from validated public domain sources such as USGS. The aim of this tutorial is to illustrate the common topological errors sometimes found in digital stream networks and provide guidance on correcting those. It has been written for SSNbler v0.1.0 and assumes the user has some familiarity with ArcGIS Pro and a good working knowledge of R Statistical Software version ≥ 4.0.0 software.

2. Example data and software requirements

This tutorial comes with an example dataset and R script. The dataset can be found in the zip archive, **topology_pro.zip**. The topology dataset should contain "ARiver.shp" (and associated files), which is a shapefile of streams data containing two sub-networks. It will also contain the subfolder, "\final_outputs," which contains a reference version of the final error-free LSN created in this tutorial. The R script, CheckTopologyPro.R, contains the R code described in this tutorial.

For this tutorial, it is assumed that the topology dataset zip file (topology_pro.zip) has been copied to the **C:\temp** folder and unzipped from there. A new folder **C:\temp\topology_pro\** will be generated, and a subfolder called **C:\temp\topology_pro\work** is also created for writing intermediate outputs. It's important that the zip file is unzipped in the C:\temp folder and that "temp" is spelled with a lower case "t." If not, the user will be required to change some path locations in the R script. The folder structure on the user's computer should look like this:



Name	Date modified	Type
final_output	6/05/2024 7:14 PM	File folder
work	1/05/2024 9:16 AM	File folder
ARiver.cpg	6/05/2024 5:44 PM	CPG File
ARiver.dbf	6/05/2024 5:44 PM	DBF File
ARiver.prj	6/05/2024 5:44 PM	PRJ File
ARiver.shp	6/05/2024 5:44 PM	SHP File
ARiver.shx	6/05/2024 5:44 PM	SHX File

3. The Landscape Network

It is important to have a basic understanding of the structure of a Landscape Network (LSN) and the relationship between the edges and nodes within it. Streams data are represented as a collection of directed edges, where the directionality is determined by the digitized direction of the line features. A nodes dataset (nodes.gpkg) is created when the LSN is generated in R using the lines_to_lsn function in SSNbler. Each edge is associated with two nodes, which correspond to the upstream and downstream endpoints of the edge. When more than one edge flows into or out of the node, they *share* a node. Thus, there should always be a single node at the intersection of edges (Fig. 1). If there is more or less than one node at an intersection, it is a topological error. If these errors are not corrected, the connectivity between line features and the observed and prediction sites associated with them will not be accurately represented in the SSN object or the spatial statistical models subsequently fit to the data. Correcting these errors is often the most time-consuming step in the pre-processing and modelling workflow.

3.1 Valid node categories

There are four topologically valid node categories in a LSN:

- **Source**: Water flow originates at these nodes and flows downstream. An edge that begins at a source node is called a headwater segment. Source nodes do not receive flow from any upstream edges.
- **Outlet**: Flow terminates at these nodes. This node is the most downstream point in the edge network. We refer to an edge that terminates at an outlet as an outlet segment. Note that a streams dataset may contain multiple subnetworks with associated outlets.
- **Confluence**: Multiple edges converge, or flow into, a single node.

- **Pseudonode**: One edge flows in and one edge flows out of a single node. While these nodes are not topologically problematic, an excessive number of pseudonodes can slow down geoprocessing operations for large datasets.

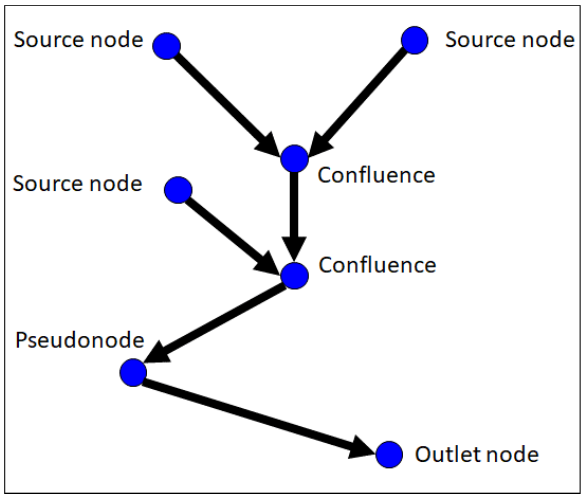


Fig. 1. An illustration of a landscape network (LSN). Nodes are denoted by blue circles, with the node category labelled. Edges are denoted by black arrows, with the arrow indicating flow direction (i.e. digitized direction).

3.2 Types of topology errors

When topological errors exist in an LSN, the flow connectivity is misrepresented, which can negatively affect statistical model outputs. Therefore, these errors must be corrected prior to fitting statistical models. Three common topological errors are recognised (Fig. 2):

1. **Unsnapped node:** occurs when two edges should be connected by a common node but are instead separated by a small distance (Fig. 2a).
2. **Dangling node:** occurs when the end node of an edge is close to another edge that it should be connected to but does not touch (Fig. 2b).
3. **Intersection without nodes:** two edges in a stream network should never intersect without a node present. This error occurs when one edge feature crosses another without a node at the intersection (Fig. 2c) or the end node of one edge feature touches another edge at any point other than the end nodes (Fig. 2d). Note that the 'outlet node' at the end of the offending line segment may also be considered a dangling node (Fig. 2c, yellow dot).

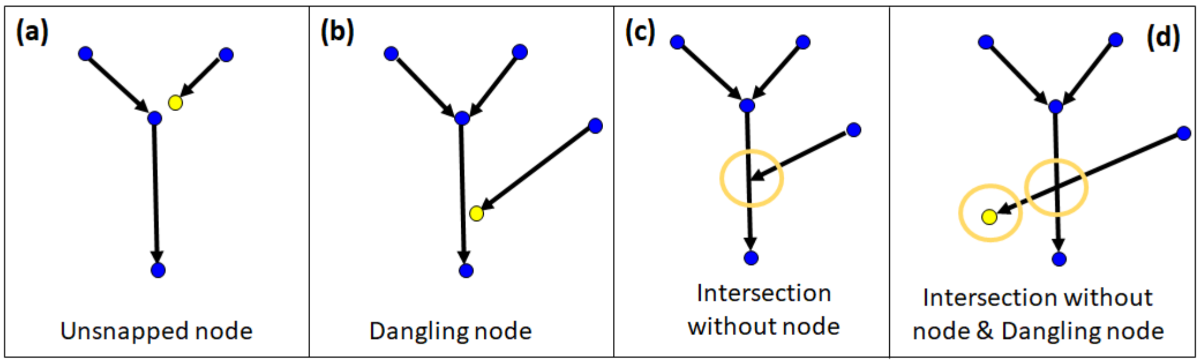

(a)
Unsnapped node
(b)
Dangling node
(c)
Intersection without node
(d)
Intersection without node & Dangling node

Fig. 2. Illustrations of topological errors in a stream network. The nodes represented as blue dots, with the node of interest shown as yellow dots and circles. The edges are represented using black arrows pointing in the direction of flow. Three types of topological errors are shown: a) Unsnapped node, b) Dangling node, c) Intersection without a node, and d) Intersection without node and Dangling node.

3.3 Additional topological restrictions

In addition to true topological errors, some additional topological restrictions are imposed on the nodes (Fig. 3). These include:

- **Converging node:** occurs when two edges flow into a node and no edges flow out (Fig. 3a).
- **Complex confluence:** occurs when more than two edges flow into a node. No more than two edges can flow into a confluence. If this condition is violated, the node is called a *complex confluence* (Fig. 3b).
- **Downstream divergence:** occurs when more than one edge flows out of a node (Fig. 3c). This topological condition occurs naturally in braided streams (Fig. 3d) but is not allowed in an LSN.

Although these topological conditions appear naturally in stream networks, they complicate the derivation of flow-based dependence structures in LSNs. Therefore, these cases must be adjusted or corrected.

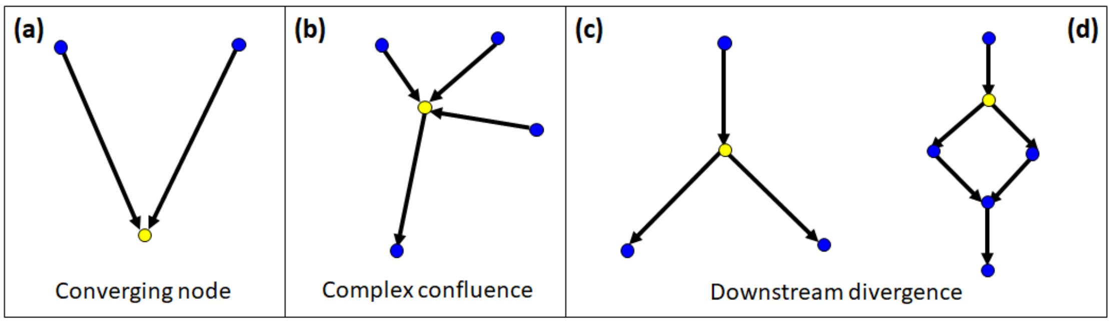


Fig. 3. Illustrations of topological features in stream networks that are restricted in landscape stream networks. Nodes are drawn as circles, and the edges are drawn as arrows pointing in the direction of flow. In each panel, the problem node is highlighted in yellow; the associated label identifies the topological restriction that is violated by the node.

The purpose of this tutorial is to demonstrate a geoprocessing workflow that will help users eliminate topological errors and restricted conditions from their stream network data using ArcGIS Pro. In general, error-correction is an iterative process that may involve repetition of manual and semi-automatic steps in various combinations. Only manual steps are showcased in this tutorial.

4. Identifying topological errors and restrictions in R

Topological errors (Fig. 2) and restrictions (Fig. 3) can be identified using the lines_to_lsn function in SSNbler. This function does three main things:

1. Builds the LSN;
2. Checks network topology and fixes minor topological errors (optional); and
3. Writes output files to a local directory.

We will begin by walking through these steps.

Open CheckTopologyPro.R in RStudio or any other IDE for R. Load the relevant R packages and set your working directory, which may be different to the one below. Run the R code step by step following the guidance below.

```
## Load SSNbler and other useful packages
library(SSNbler)
library(sf)
library(dplyr)
library(purrr)

## Set working directory
setwd("C:/temp/topology_pro")
```

Next create an sf object containing the streams data and build the initial LSN using lines_to_lsn. In this case, we are importing a shapefile, but there are no constraints on the data format, except that it must have LINESTRING geometry.

Checking topology using lines_to_lsn() is optional and there are a few arguments that must be set to do so:

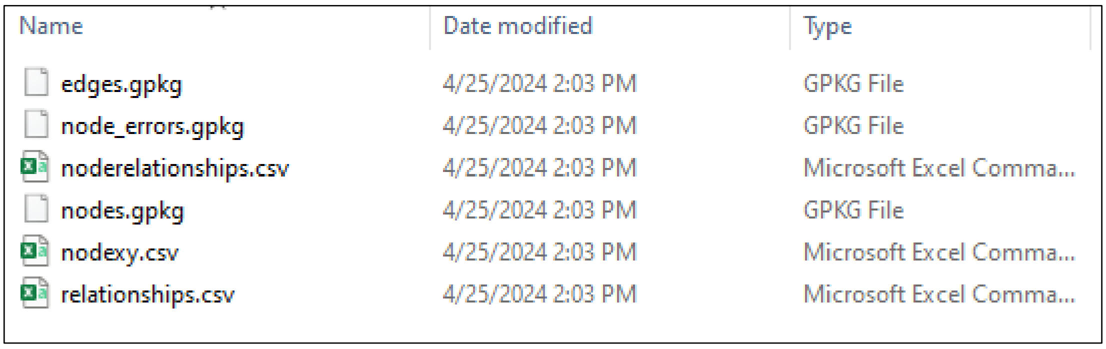
- **check_topology:** Logical indicating whether the topology should be checked;
- **snap_tolerance:** Distance in map units. Two nodes separated by a distance less than or equal to snap_tolerance are assumed to be connected.
- **topo_tolerance:** Distance in map units. Two nodes separated by a distance less than or equal to topo_tolerance are flagged as potential topological errors in the network.

```
## Import the streams dataset as an sf Object
river_net<- st_read("ARiver.shp")

## Build the initial LSN and check the topology
lsn_path1<- "c:/temp/topology_pro/work/lsn1"
edges<- lines_to_lsn(
  streams = river_net,
  lsn_path = lsn_path1,
  snap_tolerance = 1,
  check_topology = TRUE,
  topo_tolerance = 20,
  overwrite = TRUE,
  verbose = TRUE,
  remove_ZM = TRUE)
```

The arguments snap_tolerance and topo_tolerance are both provided in map units. It is generally a good idea to set the snap_tolerance to a relatively small value compared to the topo_tolerance argument. Nodes separated by a Euclidean distance ≤ snap_tolerance are assumed to be connected.

If this distance ≤ (snap_tolerance/10000), the nodes are automatically snapped when check_topology = TRUE. When (snap_tolerance/10000) ≤ distance ≤ snap_tolerance, the nodes are not snapped and are instead flagged as potential errors for the user to check. Similarly, when (snap_tolerance) < distance ≤ topo_tolerance, nodes are flagged as potential errors. To learn more about the other function arguments, simply type help(lines_to_lsn).

The function lines_to_lsn always saves five output files to lsn_path when it runs successfully. These include edges and nodes in geopackage format (.gpkg) and three .csv files containing tables that describe the topological relationships in the LSN (nodexy.csv, noderelationships.csv, and relationships.csv). If potential topological errors are found in the network, there will be an additional file named node_errors.gpkg. The LSN is comprised of the six files listed below. These were generated in the folder: C:\temp\topology_pro\work\lsn1.



Name	Date modified	Type
edges.gpkg	4/25/2024 2:03 PM	GPKG File
node_errors.gpkg	4/25/2024 2:03 PM	GPKG File
noderelationships.csv	4/25/2024 2:03 PM	Microsoft Excel Comma...
nodes.gpkg	4/25/2024 2:03 PM	GPKG File
nodexy.csv	4/25/2024 2:03 PM	Microsoft Excel Comma...
relationships.csv	4/25/2024 2:03 PM	Microsoft Excel Comma...

The contents of the new folder can also be checked from R:

```
> ## Check output files. If node_errors.gpkg exists, then there are
> ## potential errors to check
> list.files(lsn_path1)
[1] "edges.gpkg"            "node_errors.gpkg"      "noderelationships.csv"
[4] "nodes.gpkg"            "nodexy.csv"            "relationships.csv"
```

In this case, node_errors.gpkg exists, indicating that topological errors may exist. Run the following code to look at and summarize the types of topological errors and restrictions identified:

```
> ## Import node errors and format columns
> node_errors <- st_read(paste0(lsn_path1, "/node_errors.gpkg"),
+                        quiet = TRUE) %>%
+   modify_if(is.character, as.factor)
>
> ## Summarise
> summary(node_errors)
    pointid              nodecat                         error               geom
 Min.    : 275.0   Confluence:6   Complex Confluence        :1   POINT         :12
 1st Qu.: 552.0   Outlet    :6   Converging Node           :2   epsg:5070     : 0
 Median : 865.0                  Dangling Node             :4   +proj=aea ...: 0
 Mean   : 881.1                  Downstream Divergence     :2
 3rd Qu.:1328.0                  Intersection Without Node:1
 Max.   :1338.0                  Unsnapped Node            :2
 NA's   :1
```

The object node_errors contains three columns in addition to the geom (or geometry) column that will be useful for checking topology errors.

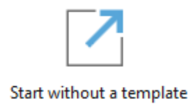
- **pointid:** point identifier for each node.
- **nodecat:** node category, which can take on values of Source, Confluence, Pseudonode, and Outlet.
- **error:** node error type, which includes topological errors (Unsnapped Node, Dangling Node, and Intersection Without Node) and restrictions (Complex Confluence, Converging Node, and Diverging Node).

Note that the pointid and nodecat columns are also found in nodes.gpkg. It is important to recognize that node_errors indicate *potential* topological errors. The user must verify whether these are true errors and then edit the topology accordingly. At this point, we move to ArcGIS Pro where the topological editing will occur. RStudio or your IDE can be minimized or closed.

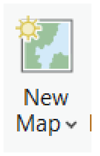
5. Verifying and correcting topological errors in ArcGIS Pro

We assume that the reader has a working knowledge of ArcGIS Pro and is familiar with ArcGIS Pro editing tools. For those that need additional help, the ArcGIS Pro documentation is a good place to start.

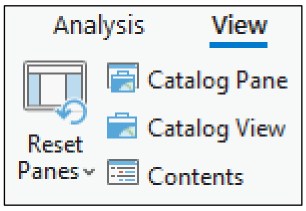
Open ArcGIS Pro. At the initial window, choose Start without a template.



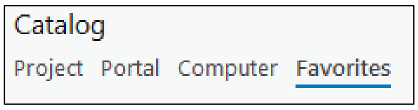
a. Click the New Map feature:



b. Choose View > Catalog Pane



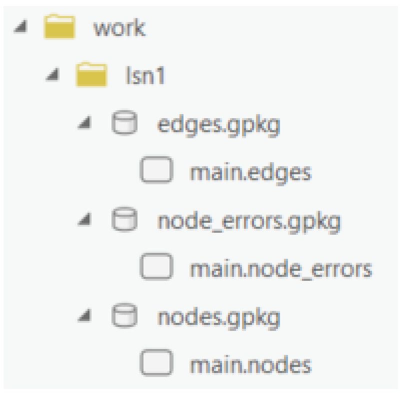
c. In the Catalog Pane, select Favorites, and right click in the pane. Choose Add Folder.



d. Navigate to the C:\temp\topology_pro folder and hit OK.

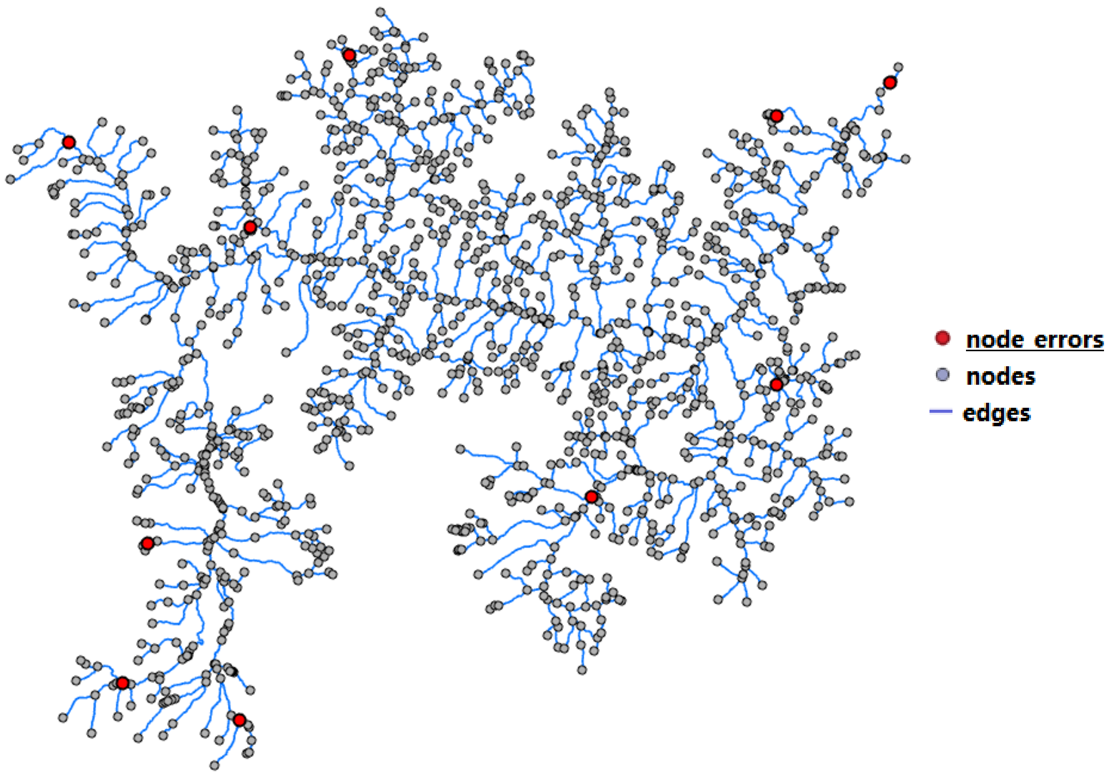
e. In the Catalog Pane, navigate to C:\temp\topology_pro\work\lsn1 folder.

f. Open the edges.gpkg, nodes.gpkg, and node_errors.gpkg containers. These are GeoPackages generated from the R code and are part of the LSN. GeoPackages are an open-source GIS data model that can be read by ArcGIS.



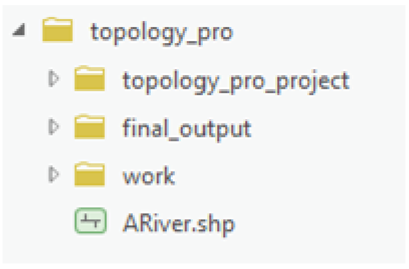
g. Drag the main.edges, main.node_errors, and main.nodes feature classes into the map viewer.

h. Modify the symbology as you see fit.



i. Save the ArcGIS project. Project > Save Project As. Navigate to the folder: C:\temp\topology_pro. Save as Name: topology_pro_project.

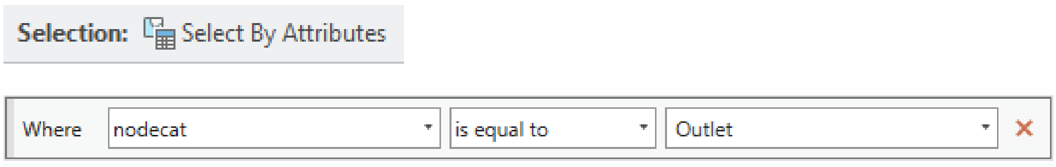
Your \topology_pro folder should now look like this in Catalog.



5.2 Check the nodecat values

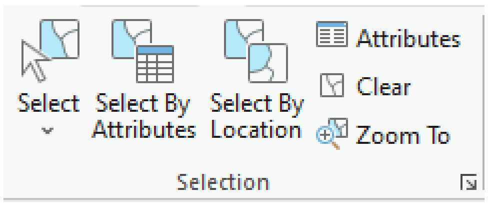
The first step should always be to check the nodes (not node_errors) on the network. In particular, the number and positions of Outlet nodes should be investigated to identify obvious errors. For example, having a large number of outlets may indicate the presence of topological errors, especially when they are found in the interior of a stream network. The extra, erroneous Outlets are usually the result of a failure to snap the end nodes of edges segments (e.g. Unsnapped Nodes or Dangling Nodes).

1. Uncheck main.node_errors so it isn't visible.
2. Open the nodes attribute table by right-clicking main.nodes in the Contents panel and clicking Attribute Table.
3. Select the rows of the attribute table where the "nodecat" = "Outlet". There should be 8 Outlet nodes selected.

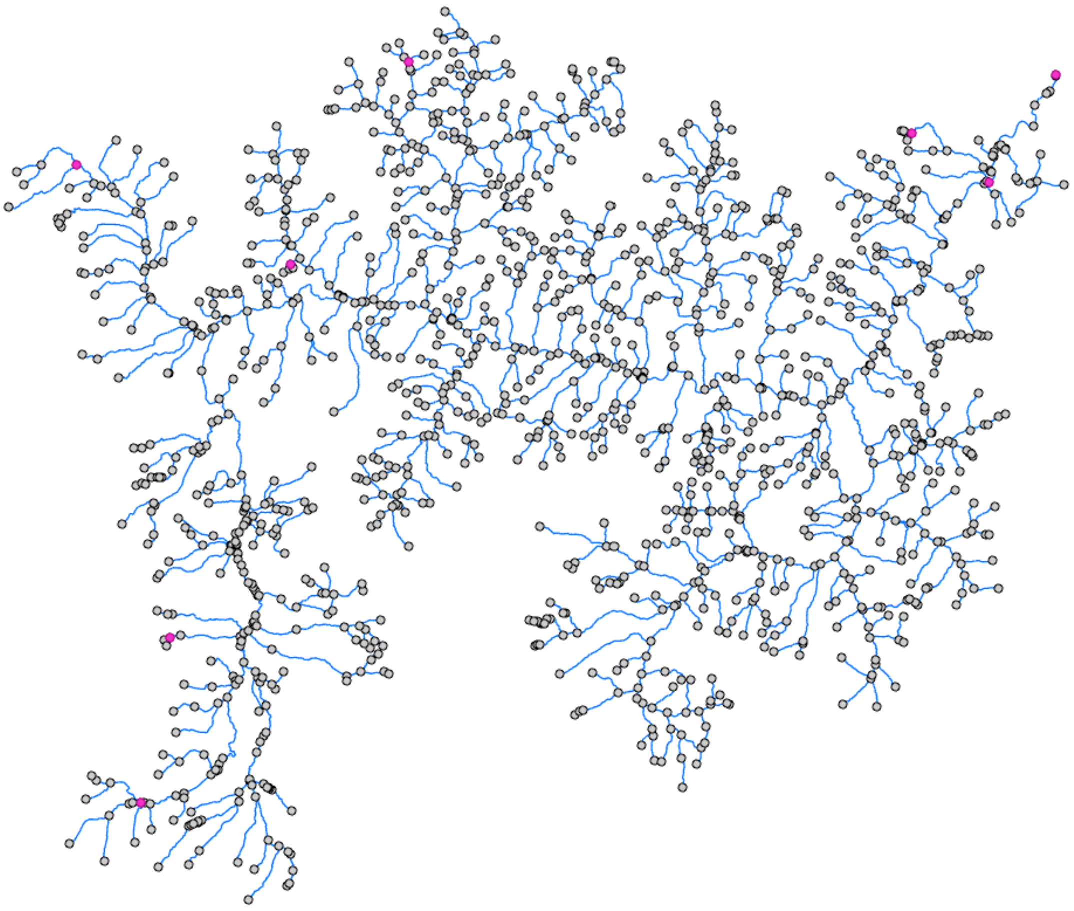


Where nodecat is equal to Outlet

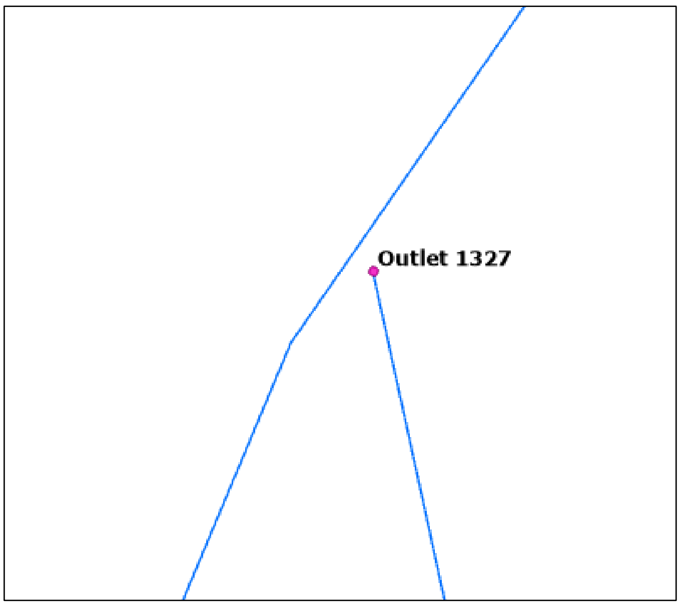
4. Close the attribute table. The selected Outlet nodes are highlighted here in pink below, but the selection colour may vary depending on user default settings. To change the highlight color for selections, choose the arrow in the lower right of the selection group.



The selection should look like this:

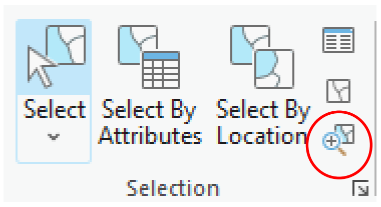


There are only 2 true outlets in this dataset, but 8 Outlet nodes have been identified (magenta). Notice the Outlets highlighted in the interior of this stream network. These are unlikely to be true outlets. Zooming in on one of the selected Outlets (pointid = 1327) reveals a situation like the one below, where a node has been identified as an outlet because it has not been snapped to the neighbouring stream segment.

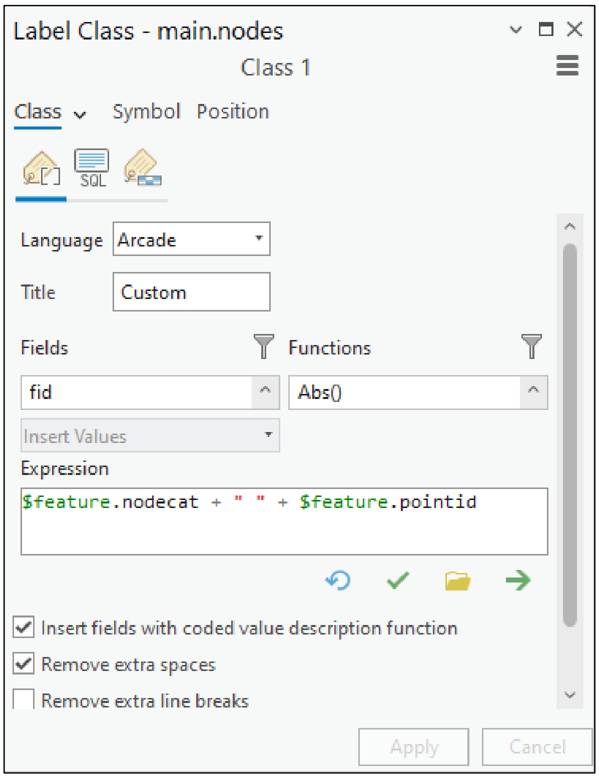


5. Investigate the Outlet nodes. For large datasets with many Outlets, manual inspection of every node will not be feasible. In this case, simply check a few at random, but pay special attention to any Outlets located in the interior of the network.

6. The true outlets are features with pointid 72 and 700. Try to find them by selecting each in the attribute table and zooming in. Use the Zoom to Selection feature to zoom to specific features.



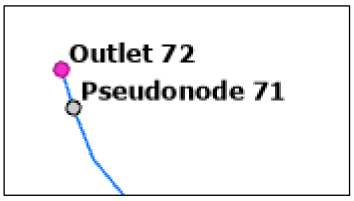
To label features with more than one attribute, right click the feature name in Contents, choose Labing Properties and populate the Label Class – Expression dialog box like this:

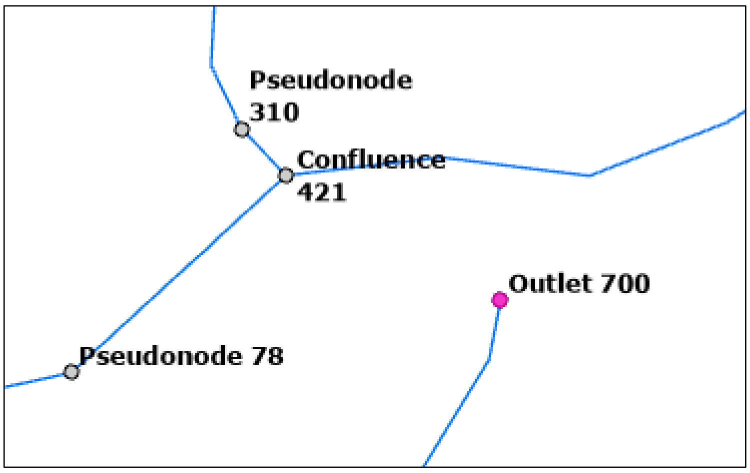


Language: Arcade
Title: Custom
Fields: fid
Functions: Abs()
Insert Values
Expression:

```
$feature.nodecat + " " + $feature.pointid
```

- [✓] Insert fields with coded value description function
- [✓] Remove extra spaces
- [ ] Remove extra line breaks

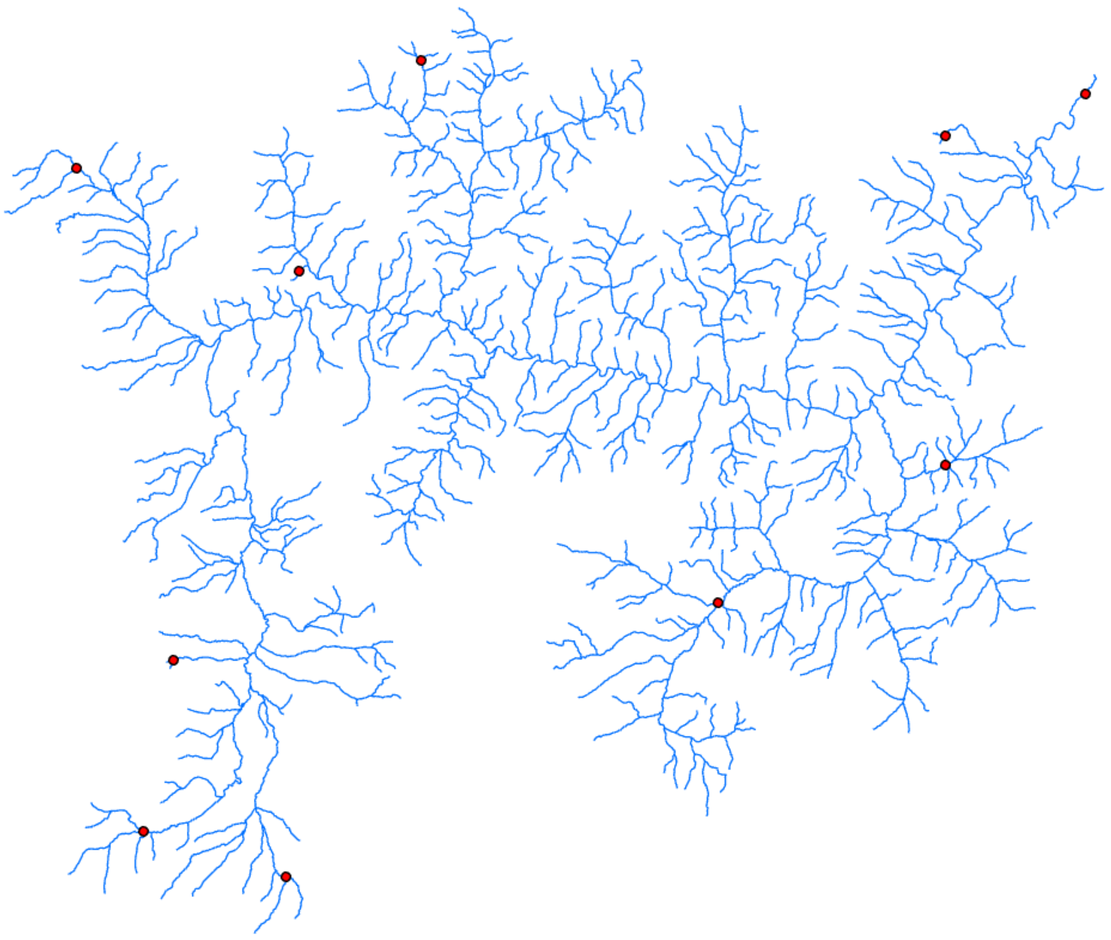




7. Turn off nodes in the Contents Pane because they will not be needed until later in this tutorial. Keep in mind that the number and positions of Outlets should be checked every time an LSN is generated by the lines_to_lsn function.

5.3 Edit various topology errors

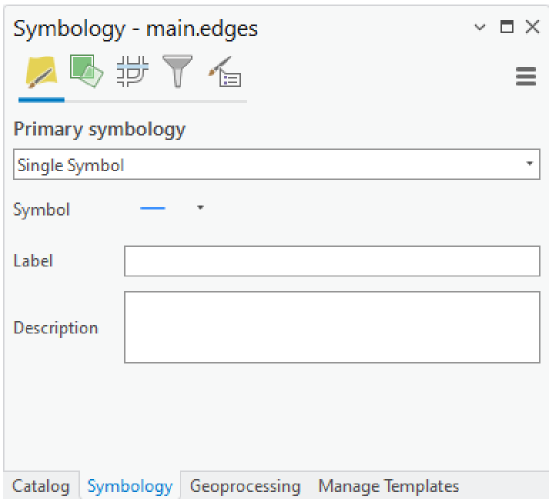
1. Turn on node_errors in the Contents Pane (if it is turned off). The Map view should look something like this, where points represent node_errors and lines represent edges.



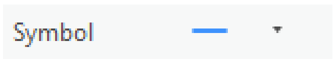
2. Change the symbology of edges to lines capped with arrows indicating the digitized direction and direction of flow in the LSN.

    a. Right-click on main.edges in the Contents panel and choose Symbology. This will open the Symbology window where the symbology for edges can be changed.

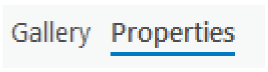
    b. Click on Symbology tab at the bottom if necessary.



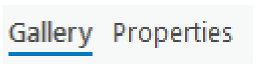
    c. Double click the blue line here, to the right of Symbol.



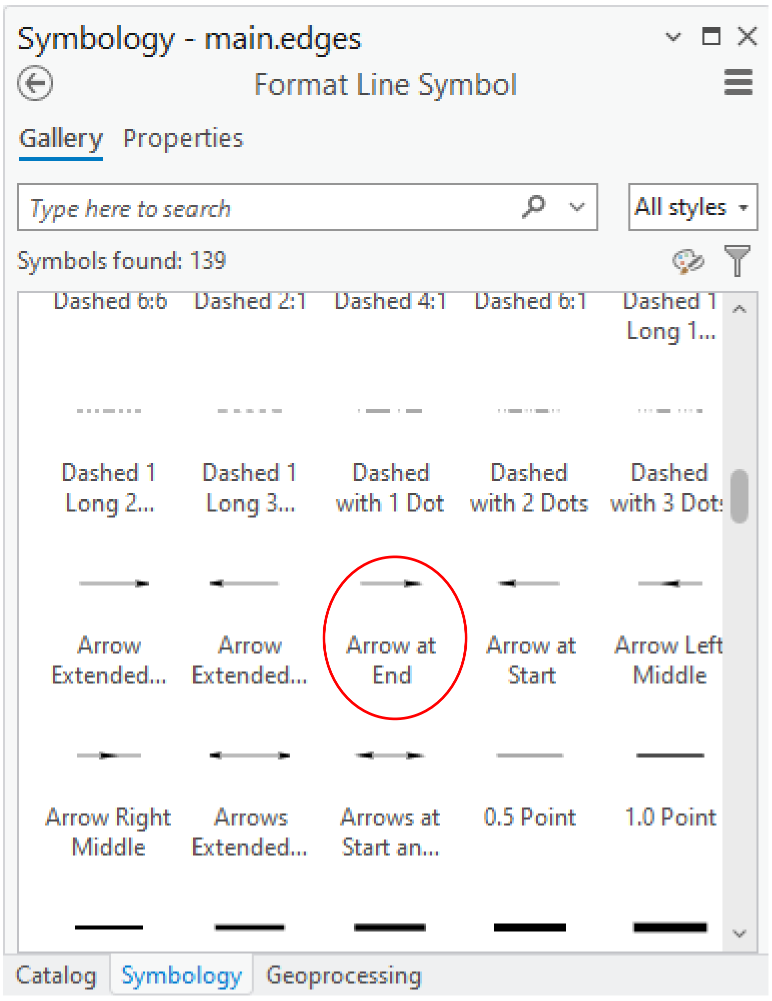
Change the window view from Properties



To Gallery view



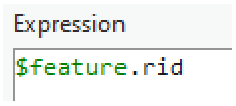
Scroll down to find Arrow at End



Click the Arrow at End options to redraw the streamline symbology.

3. Prepare to edit the Complex Confluence.

   a. First label the edges with the rid field. Right click main.edges and click Label to enable labels. Right click again and select Labeling Properties. Under Expression, type in, $feature.rid, or click rid in the Fields box, and hit Apply.

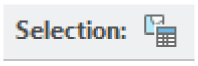


Expression

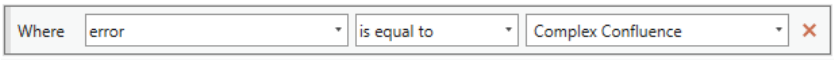
```
$feature.rid
```

   b. Right click on node_errors in the Contents Pane and click Attribute Table.

   c. The node_errors include one Complex Confluence. Choose this function:

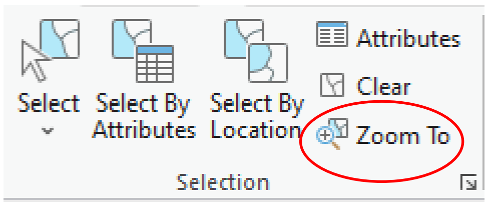


   d. Select this record:

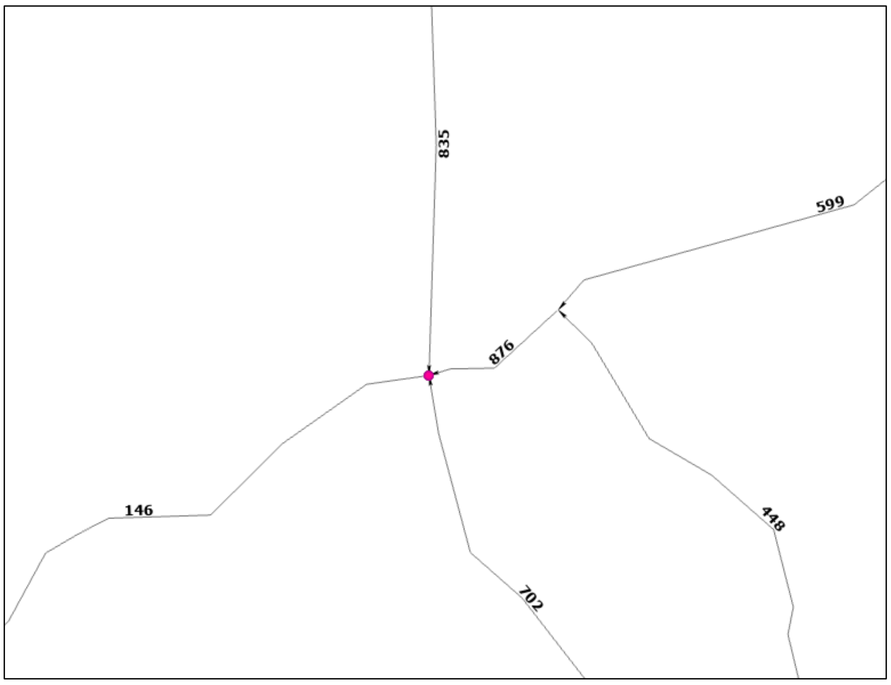


Where error is equal to Complex Confluence

e. Close the attribute table and zoom into the node_error highlighted in magenta (or another highlight color). You may have to temporarily turn off the edges in the Layers panel to see it. You can zoon to selected features using the Zoom to function in the Selection group.

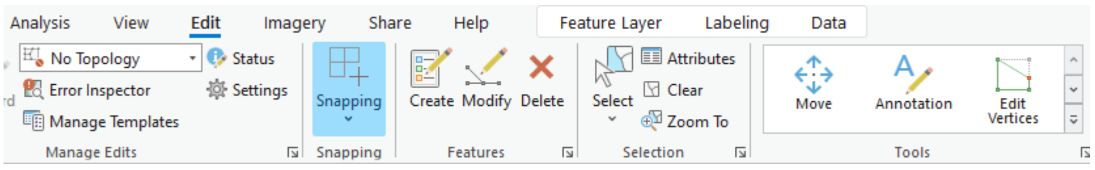


In the image below, the magenta point represents the complex confluence, and the numbers are the rid values for each edge. Notice there are three upstream edge features flowing into the node (rid = 702, 835, and 876) and one downstream edge flowing out (rid = 146). Complex confluences are not allowed and so one of the upstream edges must be edited. If one of the upstream edges is a small headwater stream, it may be appropriate to simply delete it. If not or there is another reason to retain all three edges, the end node of one upstream edge must be moved slightly up or downstream of the of the complex confluence. We describe how to do that next.

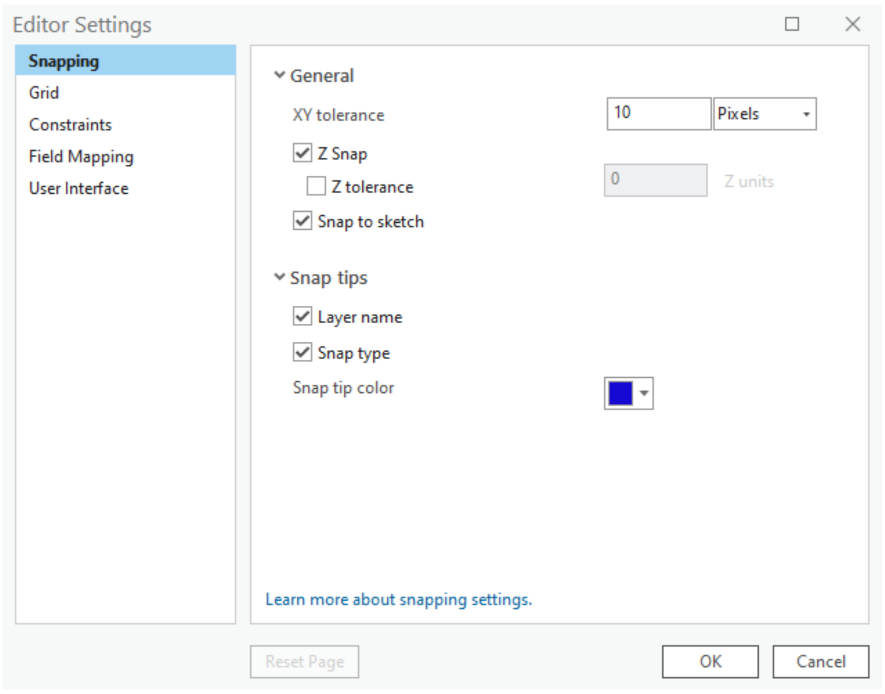


4. Edit Complex Confluence.

a. First set the snapping parameters. Select Edit to reveal the editing ribbon.



b. Click Snapping at the drop-down arrow and then click Snapping Settings. Be sure that XY tolerance is set at 10 pixels.



Editor Settings

- Snapping
- Grid
- Constraints
- Field Mapping
- User Interface

General

XY tolerance: 10 Pixels

[✓] Z Snap

[ ] Z tolerance: 0 Z units

[✓] Snap to sketch

Snap tips

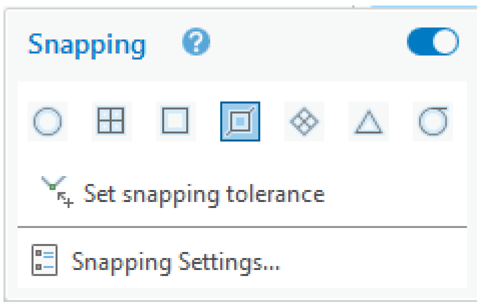
[✓] Layer name

[✓] Snap type

Snap tip color: 🟦

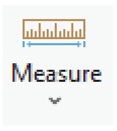
Learn more about snapping settings.

c. Hit OK to close out. Click Snapping again at the drop-down arrow to reveal the feature snapping options. Select the center feature for snapping to the nearest edge. Turn off any other selected features. Also, be sure that the slider is positioned to the right to ensure snapping is turned on.



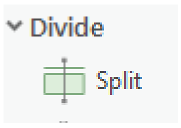
f. Next, we want to split line 146. After doing so we will snap the end point of line 835 to that split location. This procedure will eliminate the single junction with three tributaries and create two new junctions, each with only two tributaries.

First, use the measure tool to locate a position about 50 meters downstream from the Complex Confluence on reach 146. Activate the Map ribbon, then choose the measure tool to estimate the 50 meter distance.

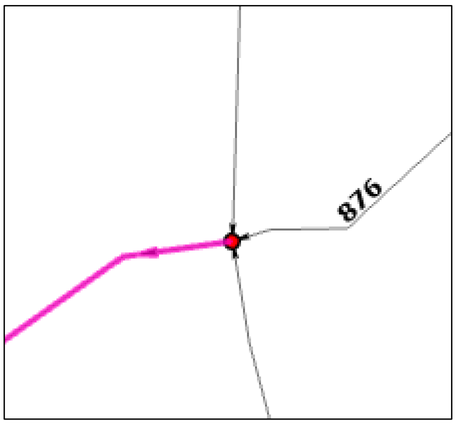


Click Edit again to reactivate the Edit ribbon.

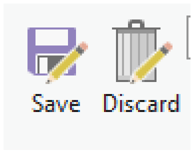
To split line 146, choose Modify, then Split on the Modify Feature pane.

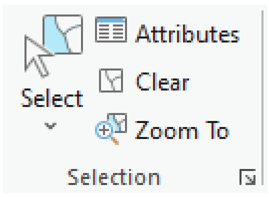


g. Select line 146 by clicking on it. Then hover over the line with the cursor until it snaps to a location about 50 meters downstream from the junction. When the line is split, a new arrowhead will appear on the streamline indicating that a new line segment has been created.

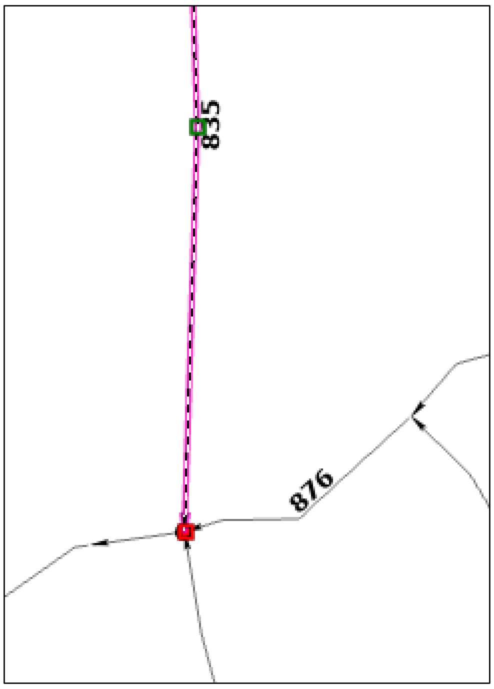


Click the Save button on the editing ribbon. If you would like to discard the change, click Discard. Agree to save all edits and clear the selection using the Clear button.

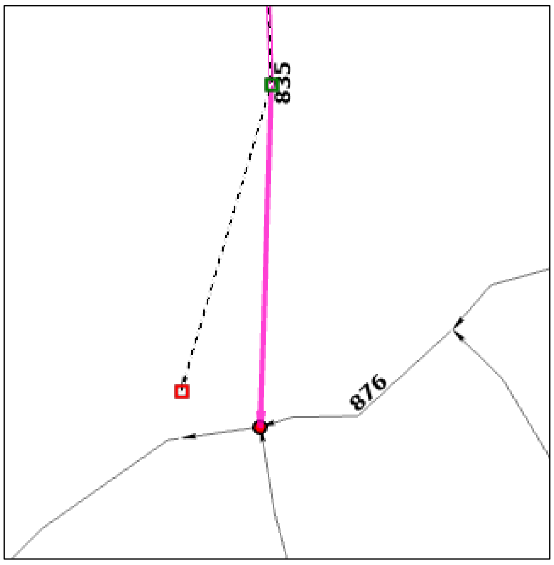




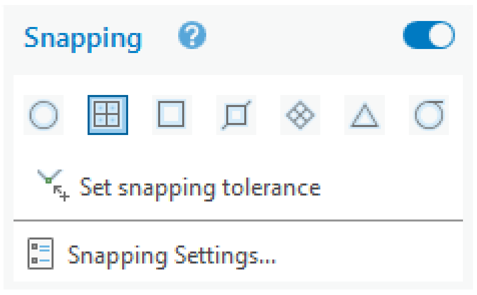
h. Now we will move the end point of line 835 to the split location on line 146. While in the Edit ribbon, select line 835. Right click the line and choose Edit Vertices at the bottom of the pull down.



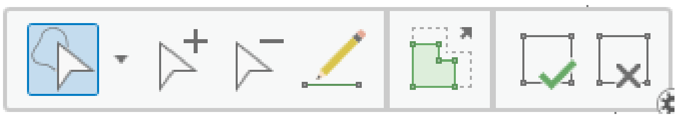
Hover over the endpoint until a diamond appears. Then click and drag the end point away from the junction. Unselect.



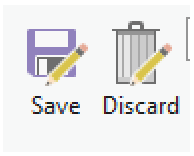
Now change the snapping option to snap at the Endpoint. Be sure to turn off edge snapping.



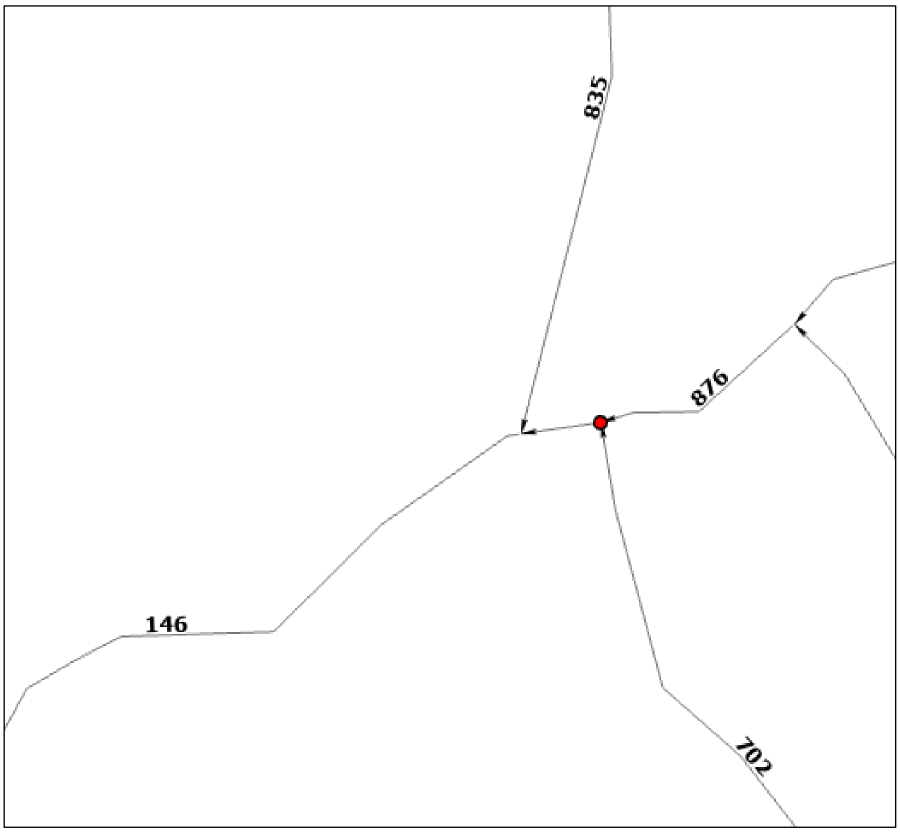
Select line 835. Right click > Edit Vertices. Hover over the end point again to select it and move the end down to the split location until it snaps at the new endpoint you created at the split location. Click the green check box to accept the change.



i. Unselect all features and save the edits with the Save button.

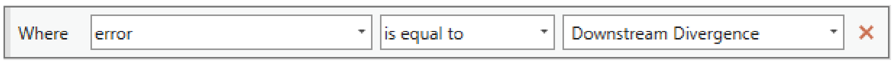


j. The final line work should look like this.



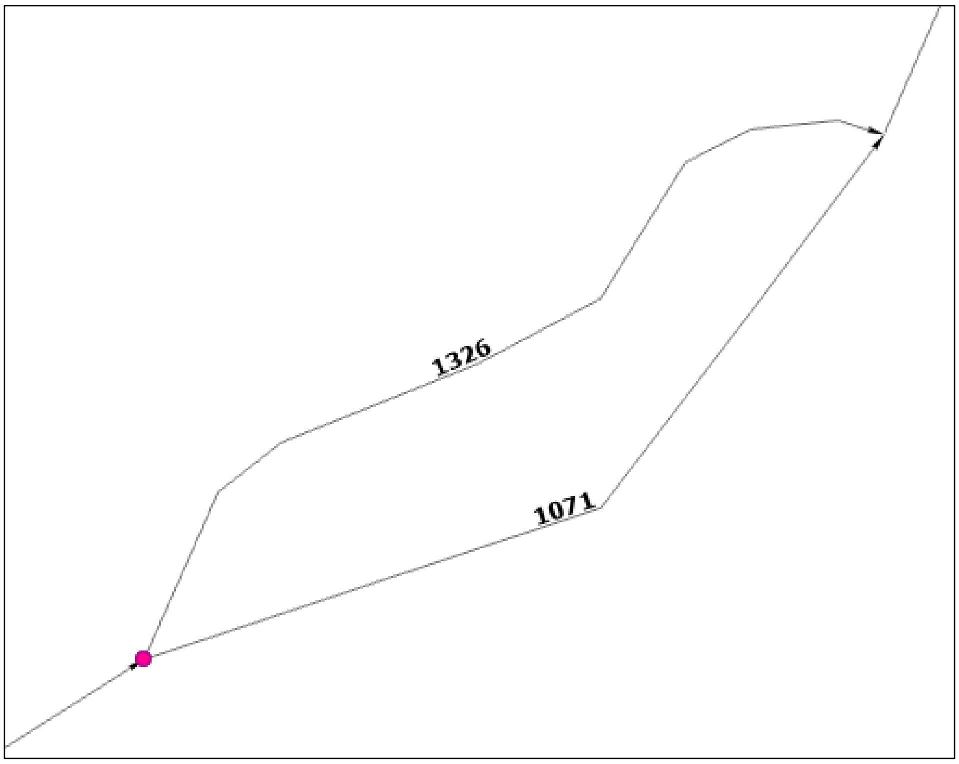
5. **Edit Downstream Divergences.**

   a. Open the node_errors attribute table and use Select by Attributes to select Downstream Divergence.



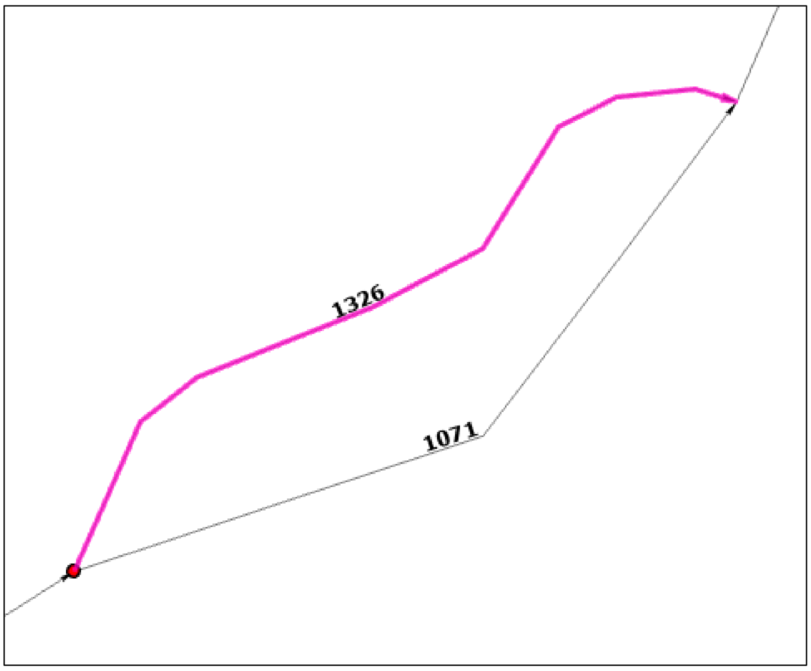
Where error is equal to Downstream Divergence

   b. There are two downstream divergences that are caused by different characteristics of the line work. First go to the error to the northeast, pointid = 627, shown below.

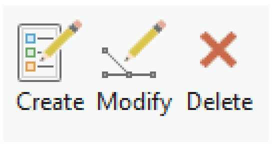


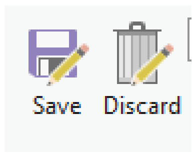
This feature represents a braided channel. If you query the two channels, you'll find that channel 1326 doesn't have a valid drainage area (TotDASqKM) or other features, compared to 1071. We'll delete reach 1326 so the channel is no longer braided in the digital representation.

   c. Activate the editing ribbon by selecting Edit. Then Select reach 1326.

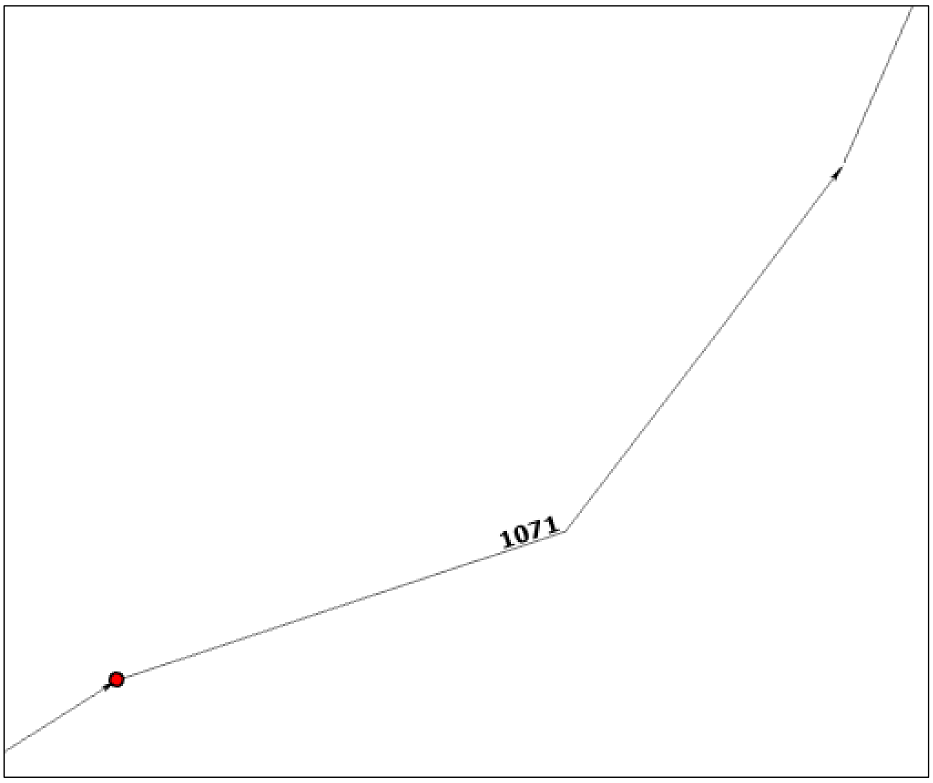


d. Hit the Delete button on the edit ribbon, then Save edits.

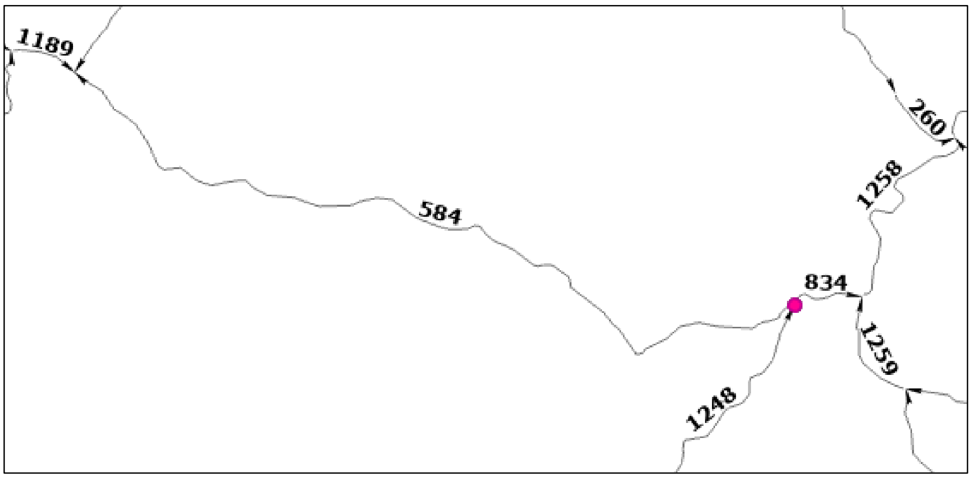




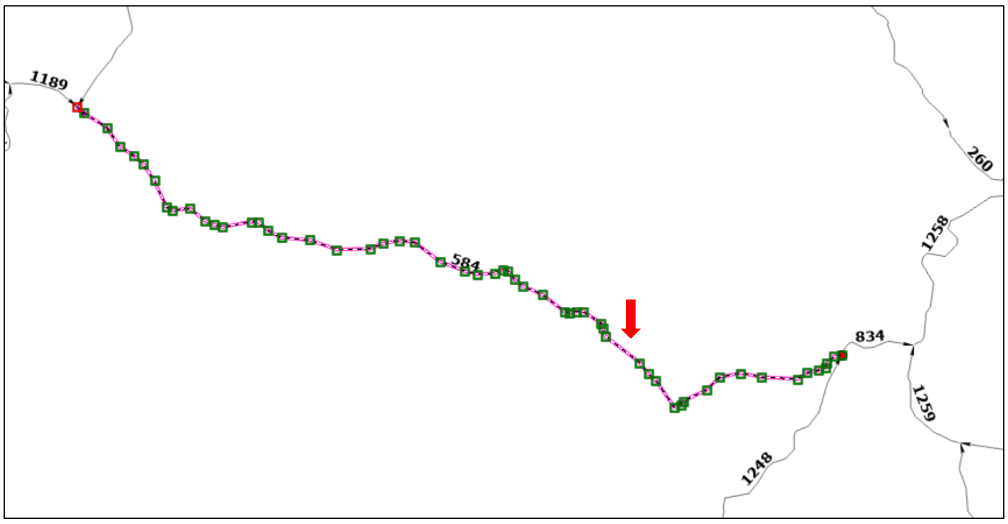
e. This Downstream Divergence has been corrected and should look like the graphic below.



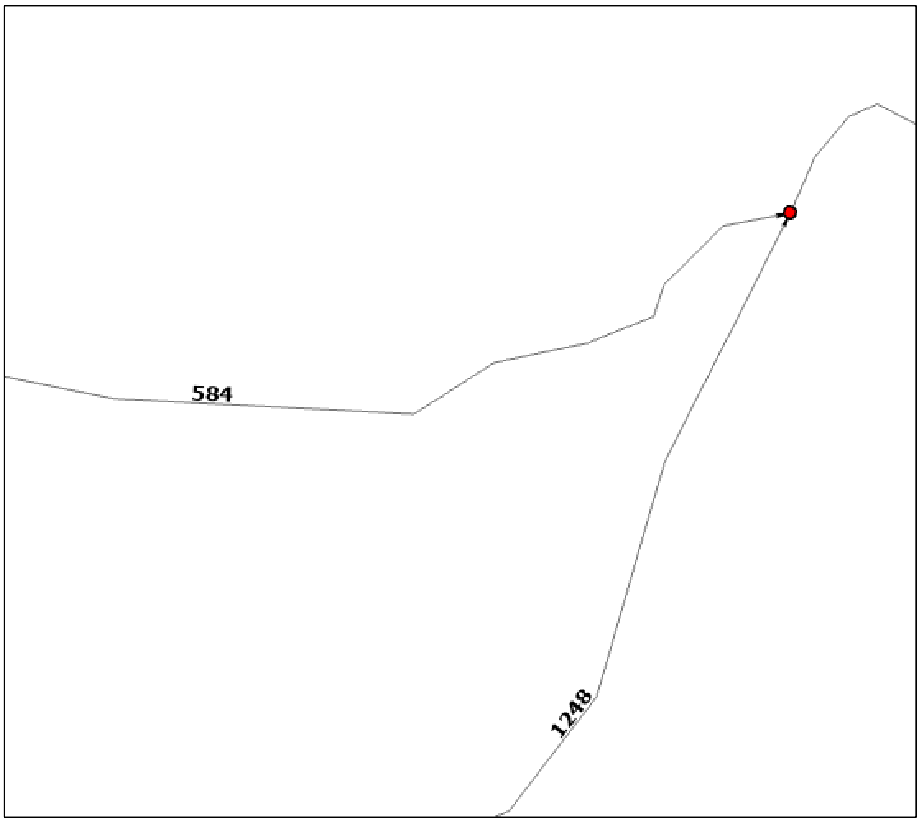
f. Next, we'll edit the Downstream Divergence where pointid = 865. Open the attribute table for main.node_errors and select pointid = 865. Zoom to that location and look at the flow direction on reach 584. You'll see that the arrowhead is pointing in the wrong direction because the digital segment was digitized in the wrong direction. This problem is causing a Downstream Divergence, where reach 1248 seems to split and flow into 584 and 834. We'll correct this issue by flipping the direction of 584.



g. Enable the Edit ribbon and select reach 584. Right click the reach and choose Edit Vertices.

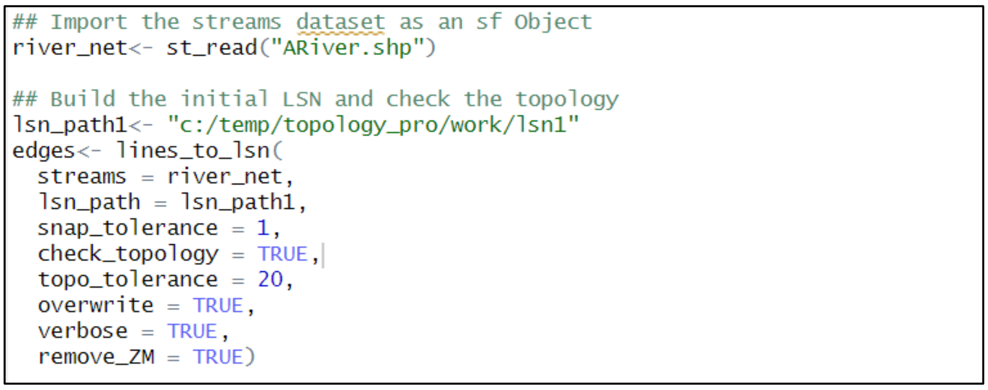


h. After the vertices are displayed, again right click on the selected reach, but do so at a part of the line without vertices (green boxes) so that the cursor isn't snapped to a vertex, like at the red arrow. Click Reverse Direction. Then click the green check box at the bottom of the screen and Save your edits. Clear the selection and you'll see that reach 584 is now pointing in the downstream direction. This Downstream Divergence is now corrected.



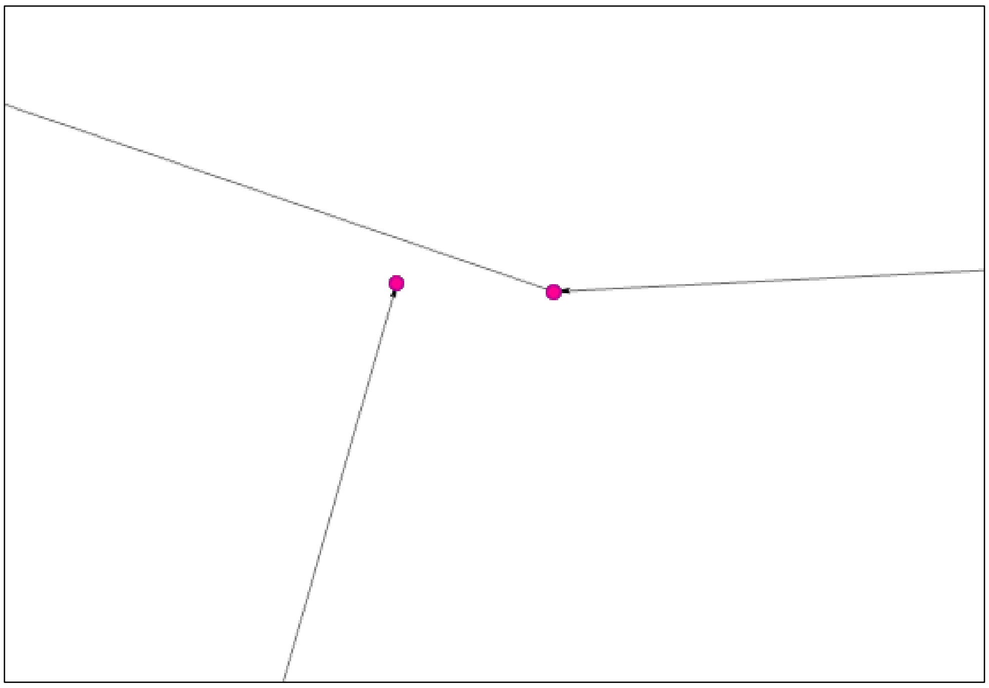
6. **Correct Unsnapped Nodes.**

a. Unsnapped nodes are two nodes that are within the snap tolerance of the lines_to_lsn function shown here.

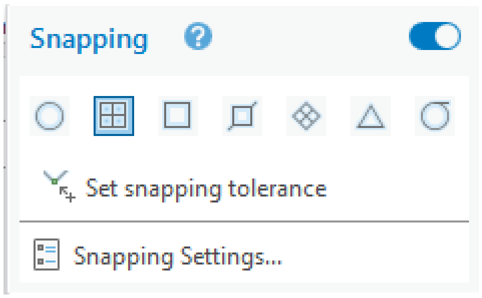


```
## Import the streams dataset as an sf Object
river_net<- st_read("ARiver.shp")

## Build the initial LSN and check the topology
lsn_path1<- "c:/temp/topology_pro/work/lsn1"
edges<- lines_to_lsn(
  streams = river_net,
  lsn_path = lsn_path1,
  snap_tolerance = 1,
  check_topology = TRUE,
  topo_tolerance = 20,
  overwrite = TRUE,
  verbose = TRUE,
  remove_ZM = TRUE)
```

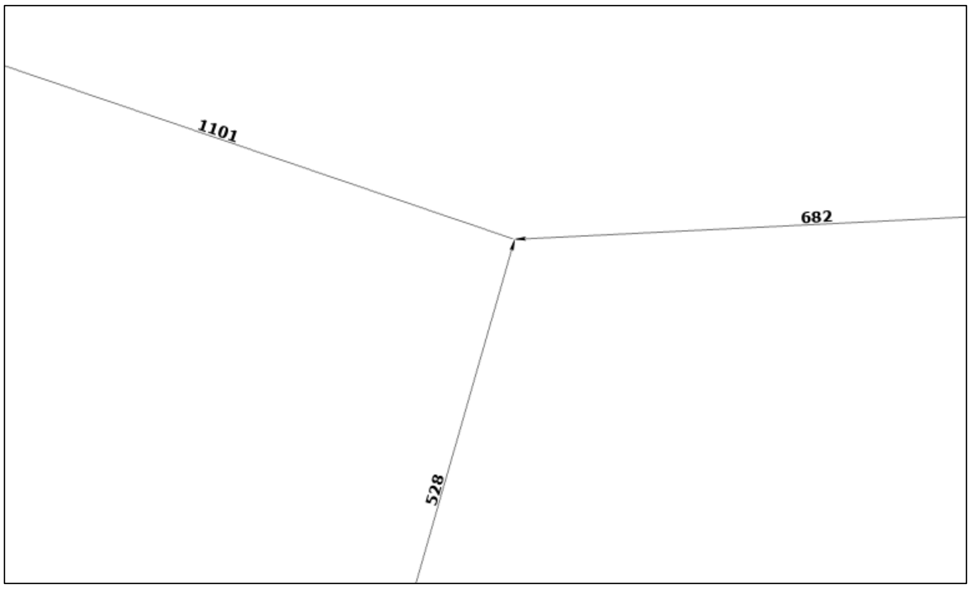
b. Open the attribute table for main.node.errors and select error = Unsnapped Node. Two nodes will be selected that are very close to each other (less than 1 meter). Zoom in until you can see both nodes.



c. Turn off the main.node_errors layer in the Contents pane. Click Edit to make the edit ribbon visible. Set snapping for endpoints.

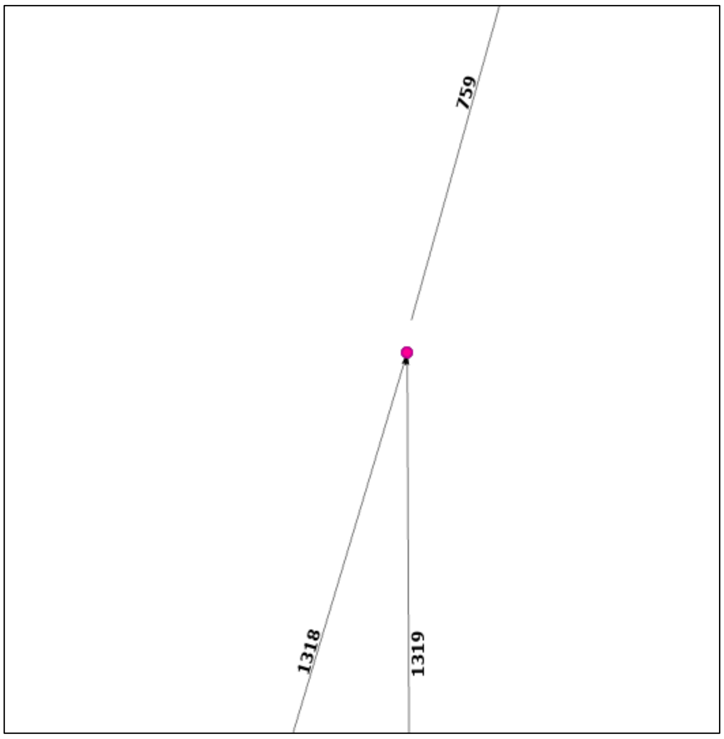


d. Select reach 528. Right click > Edit Vertices. Hover over the endpoint, then click and drag to the other node location to the northeast until they snap. Unselect and Save edits. Both Unsnapped Node errors are now corrected.

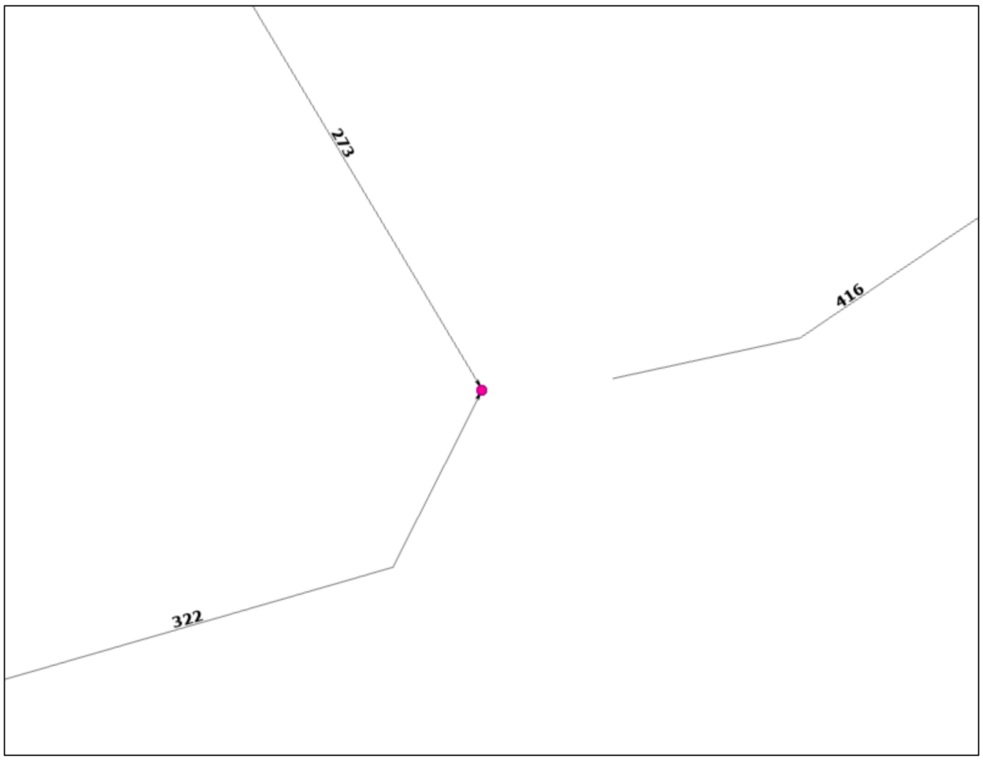


7. Correct Converging Nodes.

a. There are two node errors classified as Unsnapped Nodes. pointid = 477 and 1329. Both can be edited using the same technique. Converging Nodes flag a location where two tributaries come together, but do not continue the flow path to a downstream reach. Turn on main.node_errors.



b. Open the attribute table. Select pointid = 1329. Zoom to the node error. Turn off main.node_errors in the Content pane. Activate the Edit ribbon and make sure snapping is still set to endpoints.
c. Select reach 759. Right click and select Edit Vertices. Hover over the endpoint and click and drag to the confluence of 1318 and 1319, until the endpoints snap. Unselect and Save the edit.
d. Follow the same procedure for the next Converging Node. Zoom to the node error where pointid = 477.



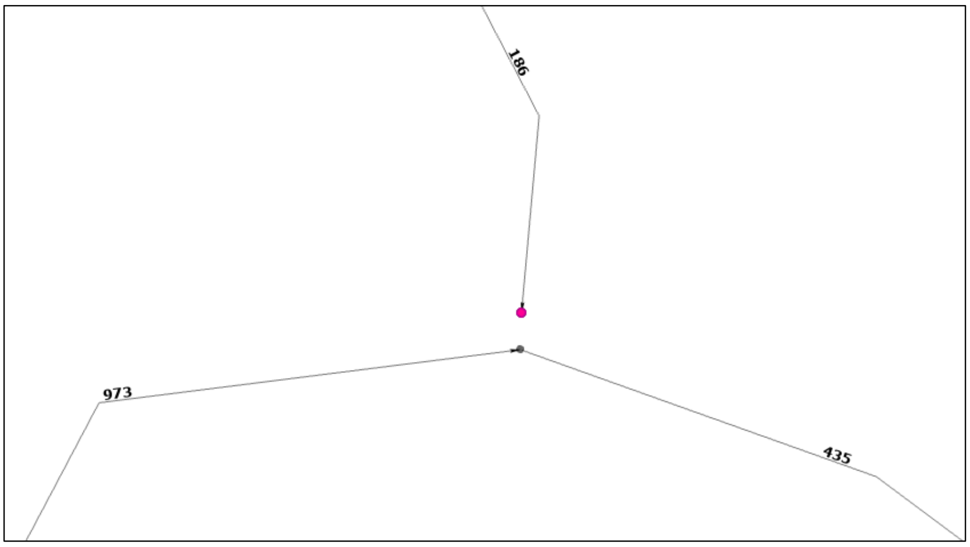
e. Turn off node errors, select reach 416, right click > Edit Vertices, then snap the endpoint to the intersection of 273 and 322. Save edits.

8. **Correct Dangling Nodes.**

a. Dangling nodes are not necessarily errors but should be checked to make that determination. Dangling nodes occur where a node and an edge are in close proximity and less than the distance specified by the topo_tolerance value in the lines_to_lsn function. In this case, 20 meters as shown below. There are four Dangling Nodes to edit in this dataset.

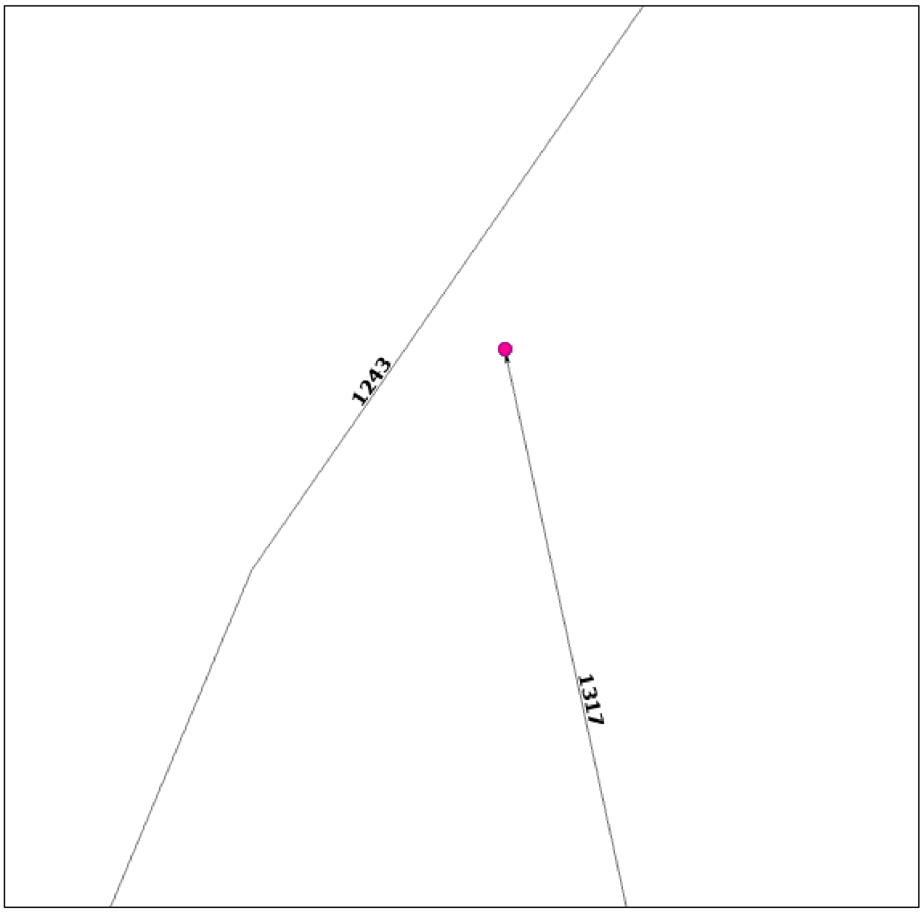
```
## Import the streams dataset as an sf Object
river_net<- st_read("ARiver.shp")

## Build the initial LSN and check the topology
lsn_path1<- "c:/temp/topology_pro/work/lsn1"
edges<- lines_to_lsn(
  streams = river_net,
  lsn_path = lsn_path1,
  snap_tolerance = 1,
  check_topology = TRUE,
  topo_tolerance = 20,
  overwrite = TRUE,
  verbose = TRUE,
  remove_ZM = TRUE)
```

b. Dangling Node 1. Turn on the node errors, nodes, and edges. Snapping should be set to Endpoints. Open the main.node_errors attribute table and select the point with pointid = 344. Zoom to that location. The dangling node and pseudo node are approximately 3 meters apart.

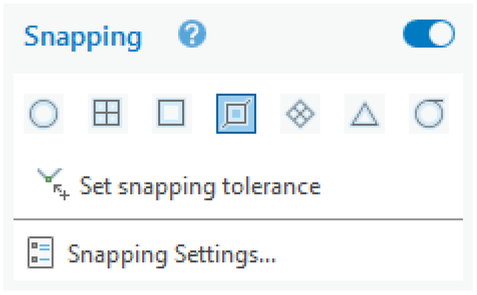


In this case we'll snap the end node on 186 to the pseudo node joining 973 and 435. Turn off the node errors and nodes. Activate the Edit ribbon. Select reach 186. Right click and choose Edit Vertices. As previously, click on the end node and pull it down to the pseudo node until the two snap. Unselect and Save edits.

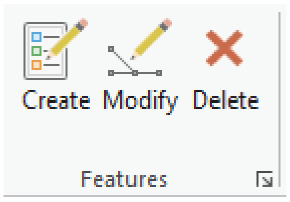
c. Dangling Node 2. In the node errors feature class, zoom to pointid = 1327. This dangling node is approximately 4 meters from the nearest edge. In this case we'll split reach 1243 and then snap reach 1317 at the split location.



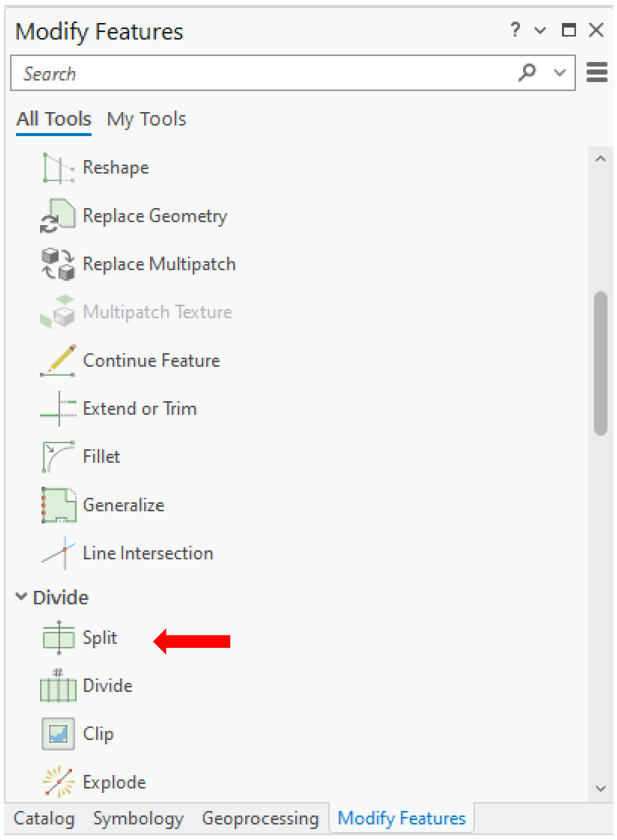
Turn off the node errors and nodes feature classes. Activate the Edit ribbon and set snapping to edge snapping.



Select reach 1243. Select Modify from the edit ribbon.



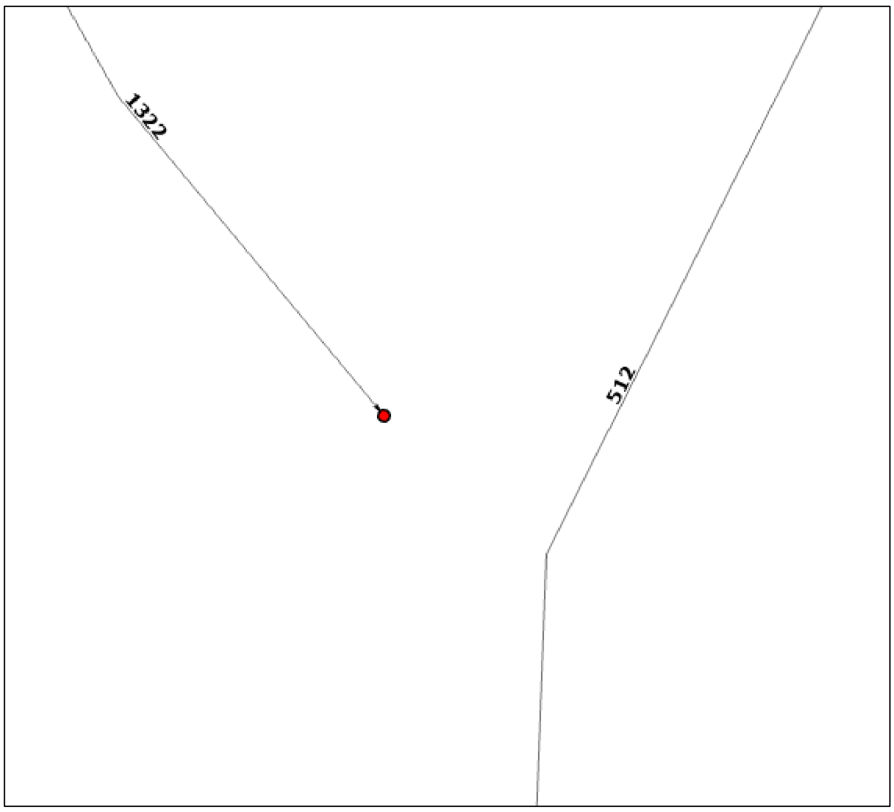
The Modify Features dialog should appear. Scroll down and select the split option.



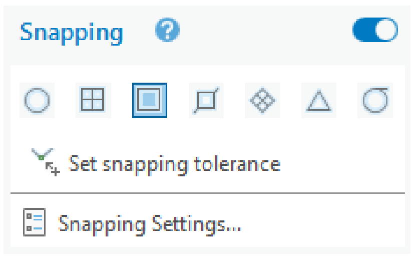
Split the selected reach at the location closest to the dangling node. Select the green check box.

Change snapping to Endpoint and turn off Edge snapping. Select reach 1317. Right click and choose Edit Vertices. Select the end point of 1317 and snap it to the split location. Hit the green check mark and unselect. Save edits.

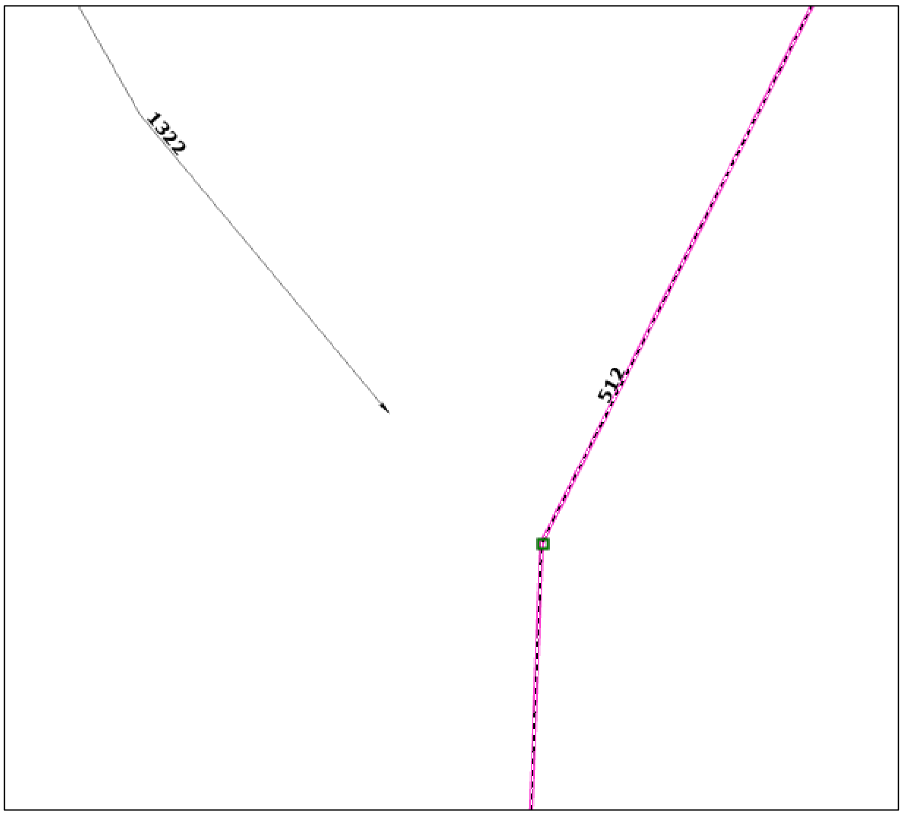
d. Dangling Node 3. Open the attribute table for main.node_errors, select pointid = 1334 and zoom to the location. In this case we'll Split reach 512 and then snap the Endpoint of 1322 to the split location.



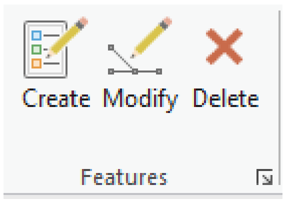
Set snapping to Vertex.

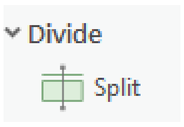


Turn off node errors in the Contents pane. Activate the Edit ribbon. Select reach 512, then right click and choose Edit Vertices. We'll be splitting 512 at the green box (vertex) shown below.

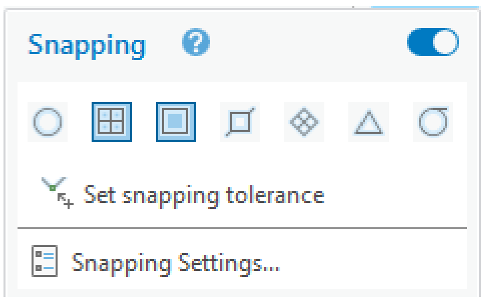


Use the procedures we've used previously to split reaches. Edit ribbon > Modify > Split.



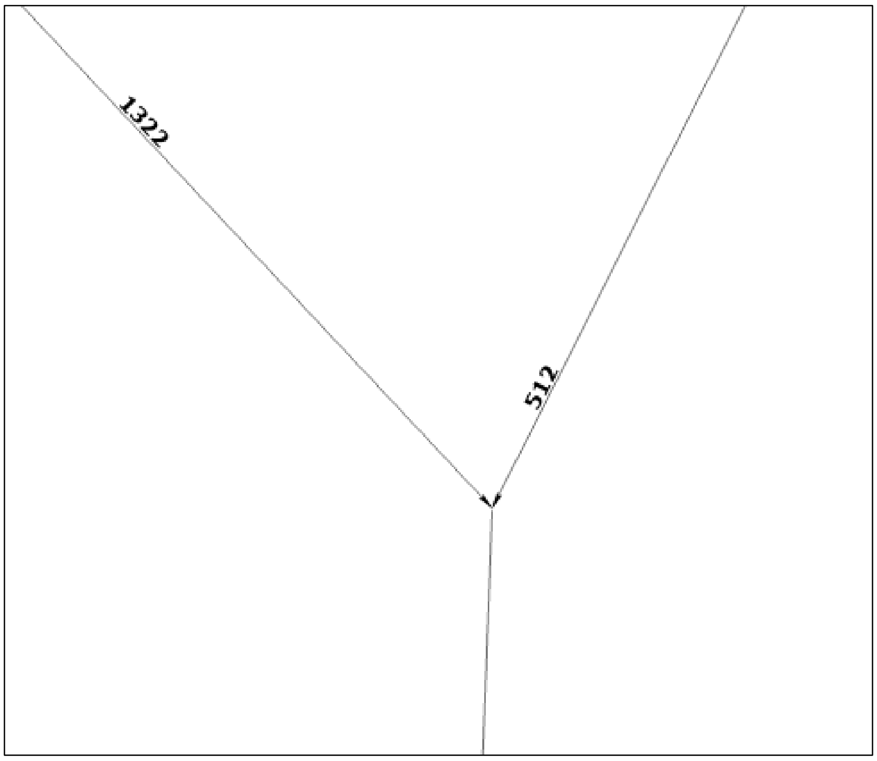


OAfter splitting at the vertex, hit the green check to accept the edit. Next, be sure the snapping options should look like this.

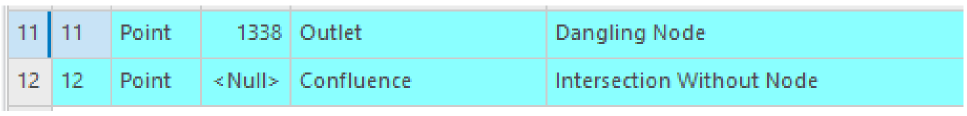


Select reach 1322, right click and select Edit Vertices. Click and hold the endpoint and snap it to the new split location. Clear the selection and Save edits.

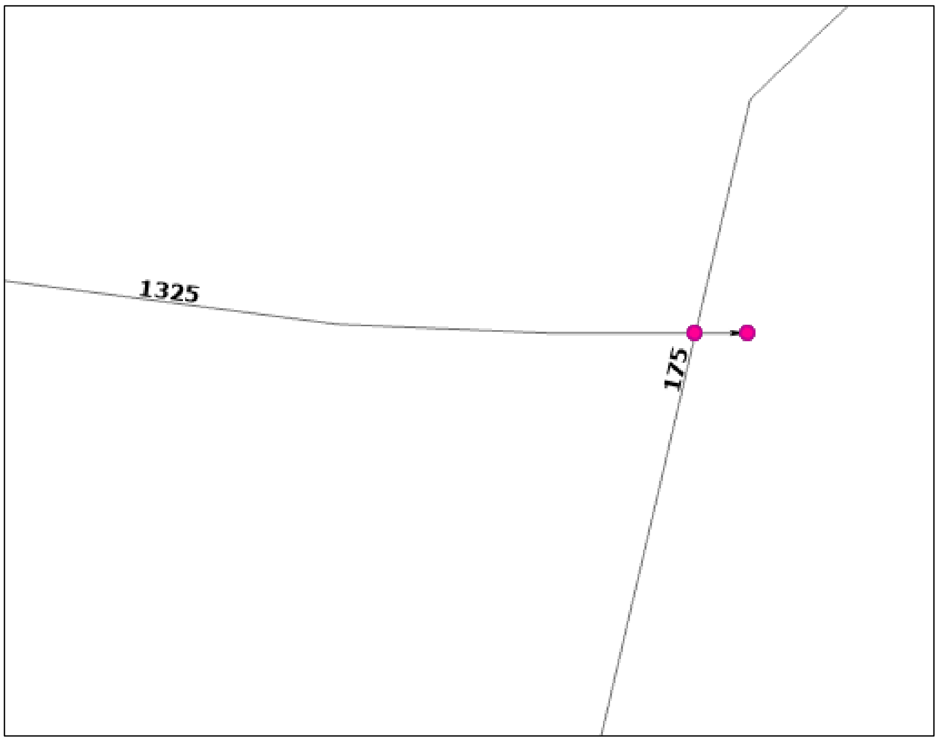
The result should look like this.



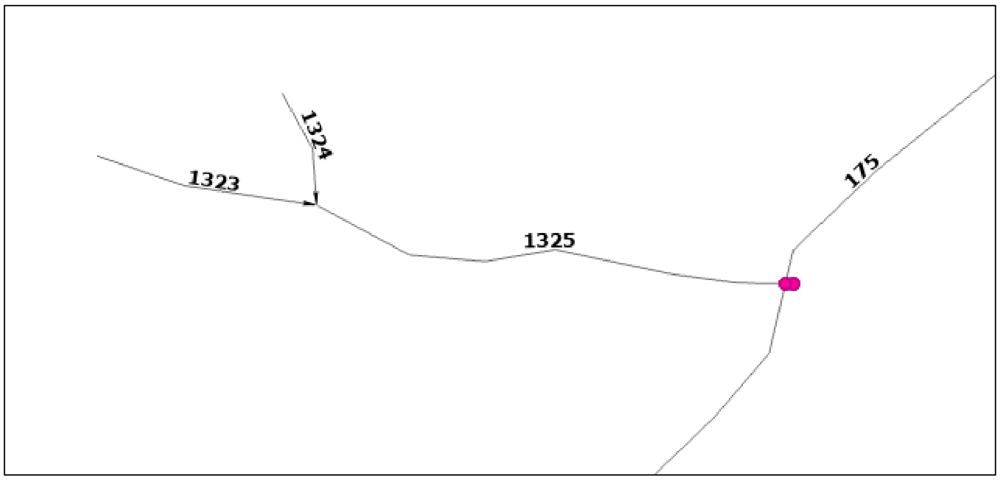
e. Dangling Node 4. The final scenario illustrates a Dangling Node and an Intersection Without Node error that occurs when one reach crosses another without a valid node intersection. Open the main.node_errors attribute table and manually select pointid = 1338 and pointid = <Null>.



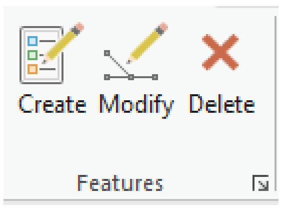
11	11	Point	1338	Outlet	Dangling Node
12	12	Point	<Null>	Confluence	Intersection Without Node



Note that two node errors are created. One where 1325 intersects 175, and another at the dangling node. Because we've split reaches and snapped endpoints previously, to correct this error we'll simple delete the small network causing the error.

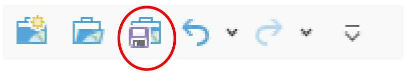


Click Edit to activate the editing ribbon. Manually select reaches 1323, 1324, and 1325. Click the Delete button in the features group.



Save edits.

Save the ArcGIS Pro project using the save button on the frame



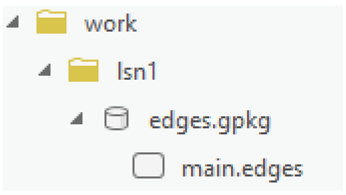
Remove all layers from the map window.

The editing exercise is complete, but we will now create a new LSN to determine if all the errors were edited correctly.

9. **Generate LSN2 and Check Node Errors**

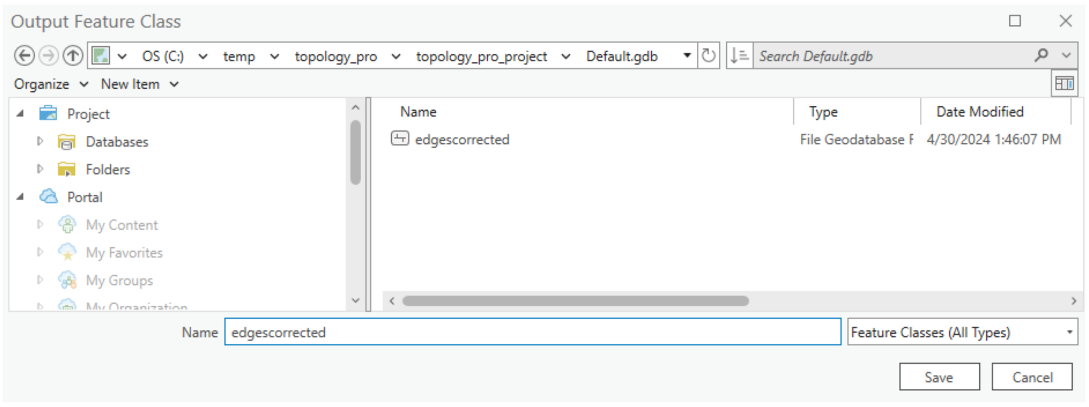
   a. Discussion: The R script generated an LSN by importing a shapefile called ARiver.shp and creating edges, nodes, node_errors, and .csv files that describe the topology. The edges, nodes, and node_errors were stored in a data model called a GeoPackage. GeoPackage is an open source, platform independent GIS model. ArcGIS Pro can read and write aspects of the GeoPackage format, but the feature topology may not be perfect after editing GeoPackages in Pro. To clean up potential topological problems it's a good idea to convert the edited GeoPackage edges to Geodatabase format, and then to shapefile for reintegrating into a new LSN.

   b. In Catalog, locate the feature class you just edited called main.edges and drag it into the Pro map viewer.



work
lsn1
edges.gpkg
main.edges

Right click main.edges in the Contents pane and select Data > Export Features. In the Export Features dialog box, choose a new Output Feature Class. Navigate to the default Geodatabase we created for the editing Project called Default.gdb at:
C:\temp\topology_pro\topology_pro_project\Default.gdb
Call the output name: edgescorrected.



Output Feature Class

OS (C:) > temp > topology_pro > topology_pro_project > Default.gdb

Search Default.gdb

Organize   New Item

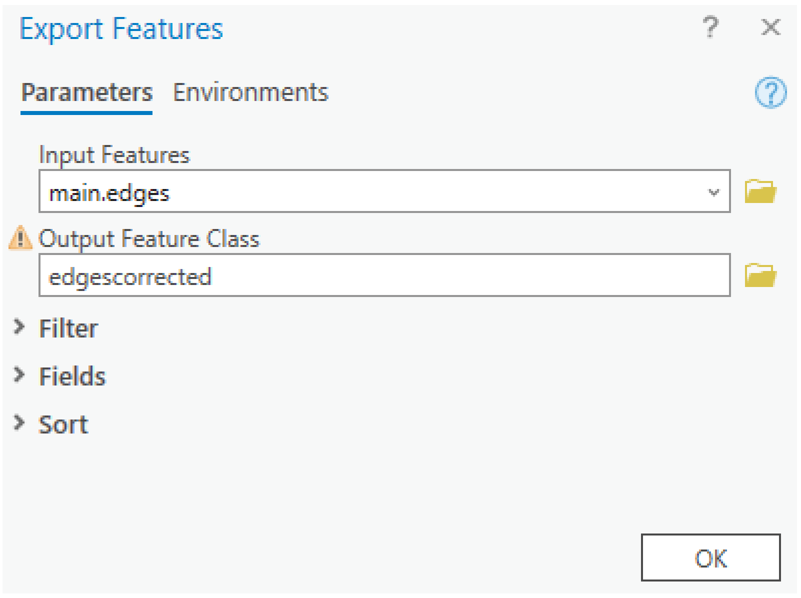
Project
Databases
Folders
Portal
My Content
My Favorites
My Groups
My Organization

Name	Type	Date Modified
edgescorrected	File Geodatabase F	4/30/2024 1:46:07 PM

Name: edgescorrected   Feature Classes (All Types)

Save   Cancel

Click OK in the dialog box.



Export Features

Parameters / Environments

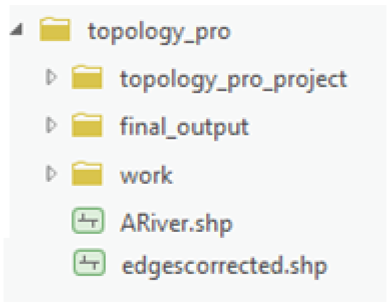
Input Features: main.edges

Output Feature Class: edgescorrected

A new layer will be added to the map viewer Contents pane called edgescorrected. Now we will convert this feature class to a shapefile.

Right click edgescorrected in the Contents pane and again select Data > Export Features. In this case, set the output location to C:\temp\topology_pro. The output shapefile name should be: edgescorrected.shp.

The shapefile should be here in Catalog.



```
topology_pro
    topology_pro_project
    final_output
    work
    ARiver.shp
    edgescorrected.shp
```

Remove all the edges feature classes from the Map viewer.

c. If you closed R, open CheckTopologyPro.R again in RStudio or any other IDE for R. Load the relevant R packages and set your working directory, which may be different to the one below. Run the R code step by step as described below.

```r
## Load SSNbler and other useful packages
library(SSNbler)
library(sf)
library(dplyr)
library(purrr)

## Set working directory
setwd("C:/temp/topology_pro")
```

d. Import the edited shapefile and build lsn2.

```r
## Import the edited streams dataset as an sf Object
river_net<- st_read("edgescorrected.shp")

## Build the initial LSN and check the topology
lsn_path2<- "c:/temp/topology_pro/work/lsn2"
edges<- lines_to_lsn(
  streams = river_net,
  lsn_path = lsn_path2,
  snap_tolerance = 1,
  check_topology = TRUE,
  topo_tolerance = 20,
  overwrite = TRUE,
  verbose = TRUE,
  remove_ZM = TRUE)
```

e. List files that were generated in the LSN.

```r
## Check output files. If node_errors.gpkg exists, then there are
## potential errors to check
list.files(lsn_path2)
```

```
[1] "edges.gpkg"          "noderelationships.csv" "nodes.gpkg" "nodexy.csv"
[5] "relationships.csv"
```

If there is no file named node_errors.gpkg, then there are no topological errors in the edges.gpkg feature class.

If *potential* errors still exist, you will need to go back to ArcPro and check them. If they are not true topological errors, then it is ok to ignore them and move onto the next processing steps in SSNbler.

Congratulations! You've completed this exercise.