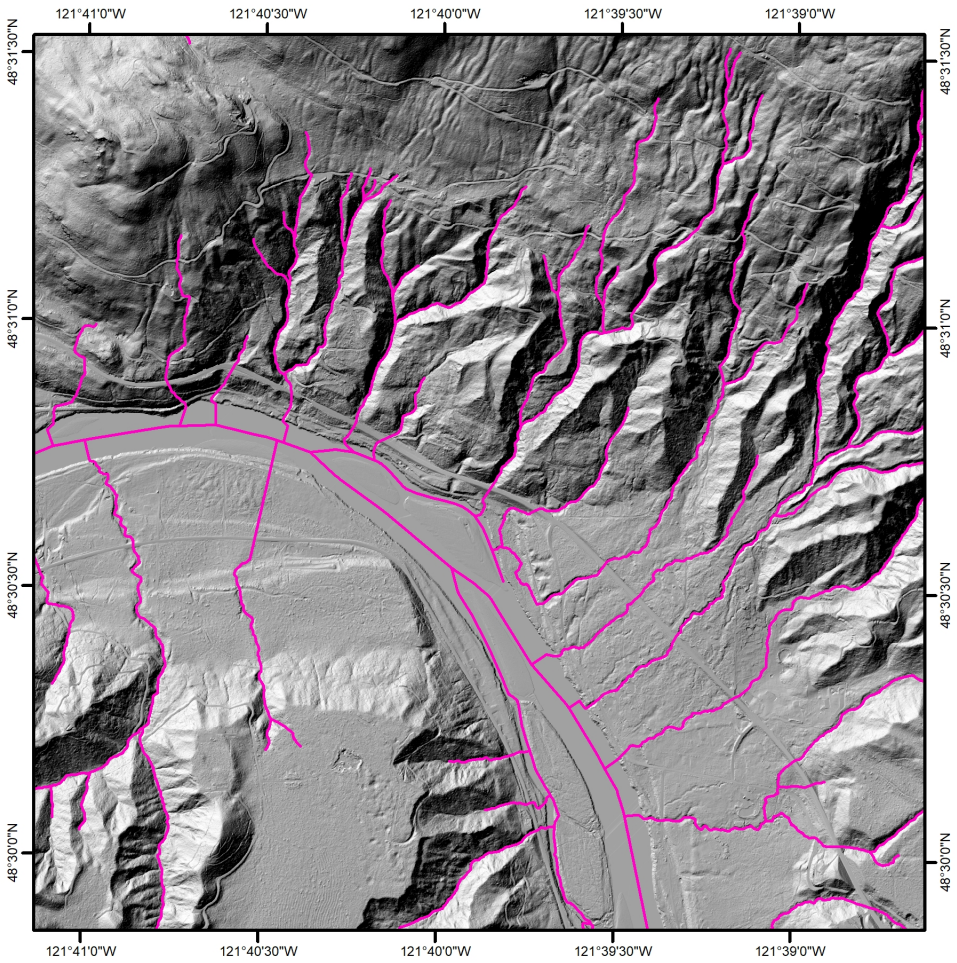
Methods description for the Skagit River lidar-derived hydrography



Tim Hyatt, Gus Seixas, & Kate Ramsden
Skagit River System Cooperative
February 2, 2022

Skagit River System Cooperative

Introduction

Analysis of fish populations and fish habitats in the Skagit River watershed has been hampered for decades by the lack of mapped hydrography that accurately depicts the existence and location of streams in a digital geographic information system (GIS) format. Without accurate hydrography it is impossible to meaningfully evaluate the shading effects of riparian vegetation on streams at a spatial scale relevant for watershed-level restoration planning, and difficult to map the location, extent, and distribution of anadromous and resident fish. Similarly, locating fish-blocking culverts at road crossings is often ambiguous when mapped streams do not correspond with the actual topography, and mapping protective stream buffers is inaccurate at best, and likely much less effective, when maps either include streams that don't exist on the ground, or don't include streams that do exist. Lidar-derived hydrography (LDH) does not completely eliminate this last problem, but it does improve the locational accuracy of most streams.

In 2016 the Swinomish Tribe and Skagit County contributed to a Washington DNR effort to collect lidar data that, when combined with previous lidar, would fully encompass Skagit anadromous zones. The Skagit River System Cooperative (SRSC), an environmental protection and restoration consortium of the Swinomish Indian Tribal Community and the Sauk-Suiattle Indian Tribe, has since constructed an updated hydrography layer that combines new lidar-derived hydrography where it can be generated, with existing hydrography from the National Hydrography Dataset (NHD) in areas where the NHD is as good or better than can be generated from lidar. Thus in the headwaters where lidar does not yet exist for the Skagit, the NHD was retained. Likewise, the NHD was retained in the mainstems and many lowland areas where the NHD has been recently updated using aerial photography, particularly in the agricultural drainages. In the many miles of small and intermediate streams, particularly those in forested areas, the new lidar hydrography provides distinctly superior stream arcs over the NHD and other available hydrography layers.

This document is a description of the data sources and methods used to create the new Skagit LDH and incorporate within it the existing NHD hydrography.

Data sources

High-resolution lidar rasters (Table 1) were assembled for the entire Skagit anadromous zone and converted to common horizontal (State Plane Washington North NAD83, ft) and vertical (NAVD88, ft) spatial reference frames. To calculate consistent watershed areas throughout the basin, areas of the Skagit watershed where lidar was not available were represented instead with a USGS 10m digital elevation model (DEM), and merged with the high-resolution lidar on a 3-foot grid spacing.

Table 1. lidar flight characteristics

Lidar flight	Acquisition Year	Source/Client	First return point density points/m2	Ground Classified points/m2
Western Washington 3DEP	2016 - 2017	USGS, WaDNR	12.29	2.46
Glacier Peak	2014-2015	USGS Volcano Observatory	27.05	2.86
Mt Baker	2015	USGS Volcano Observatory	19.73	2.34
Skagit Coastal*	2019	NOAA	14.92	8.94

* Water-only tiles omitted from density calculations

Generating single-thread hydrography

New hydrography for the Skagit and Samish watersheds (WRIAs 3 & 4) was generated using the combined lidar and 10m DEM mosaic. The process was to 1) run an initial set of hydro routines, which often depicted flow alignment errors, 2) digitize correct flow paths to reroute those errors, and 3) re-run the entire hydro when all the errors had been corrected. To ease the computational burden, USGS HUC-10 watersheds were run individually and then merged when final. An ArcPy script, provided at the end of this document, was run iteratively to generate an efficient work flow and assure consistency of results.

Channel initiation was set to 25.8 acres (125,000 3x3 ft pixels), which approximates hydrography on the USGS 7.5 minute topographic maps.

Extensive editing and modification was necessary to direct flow correctly through natural flow impoundments and anthropogenic channel modifications such as road culverts, dams, and ditches. The ArcPy script operates on two inputs: 1) an existing, unmodified DEM and 2) an arc feature class of flow modification paths (“trenches"). These trenches can represent culverts, dams, ditches, or other hydromodifications where the lidar does not initially detect the correct flow path Figure __). These arc features are typically digitized manually, but could be adapted from existing data on road culverts or other hydromodifications.

Once the inputs have been assembled and the filenames, paths, licenses, and other variables have been set, the script proceeds through several steps to generate hydrography.

- First, if only working on a sub-basin, the script clips out a temporary DEM on which to calculate the hydro. This step greatly speeds processing by limiting the processing in smaller areas, or can be skipped if working on an entire watershed.
- The script then “excavates” the DEM surrounding the "trench" arcs, to a depth equivalent to the lowest elevation within six feet of the trench, and then creates a new, temporary DEM that incorporates the excavations.
- Using the trenched DEM, the script applies the standard sink filling, flow direction, and flow accumulation algorithms. During this step the trenches are re-filled so that flow is (usually)

directed to the correct downstream outlet. New hydrography arcs are generated along the corrected flow paths.

The script can be run iteratively for sub-basins until a correct configuration of trenches is digitized, then the basin run as a whole for a consistent and connected hydrography feature class. It is important that the final run is on a hydrographically complete DEM (which extends to the uppermost ridges) to obtain consistent stream initiation points, which are based on contributing watershed area.

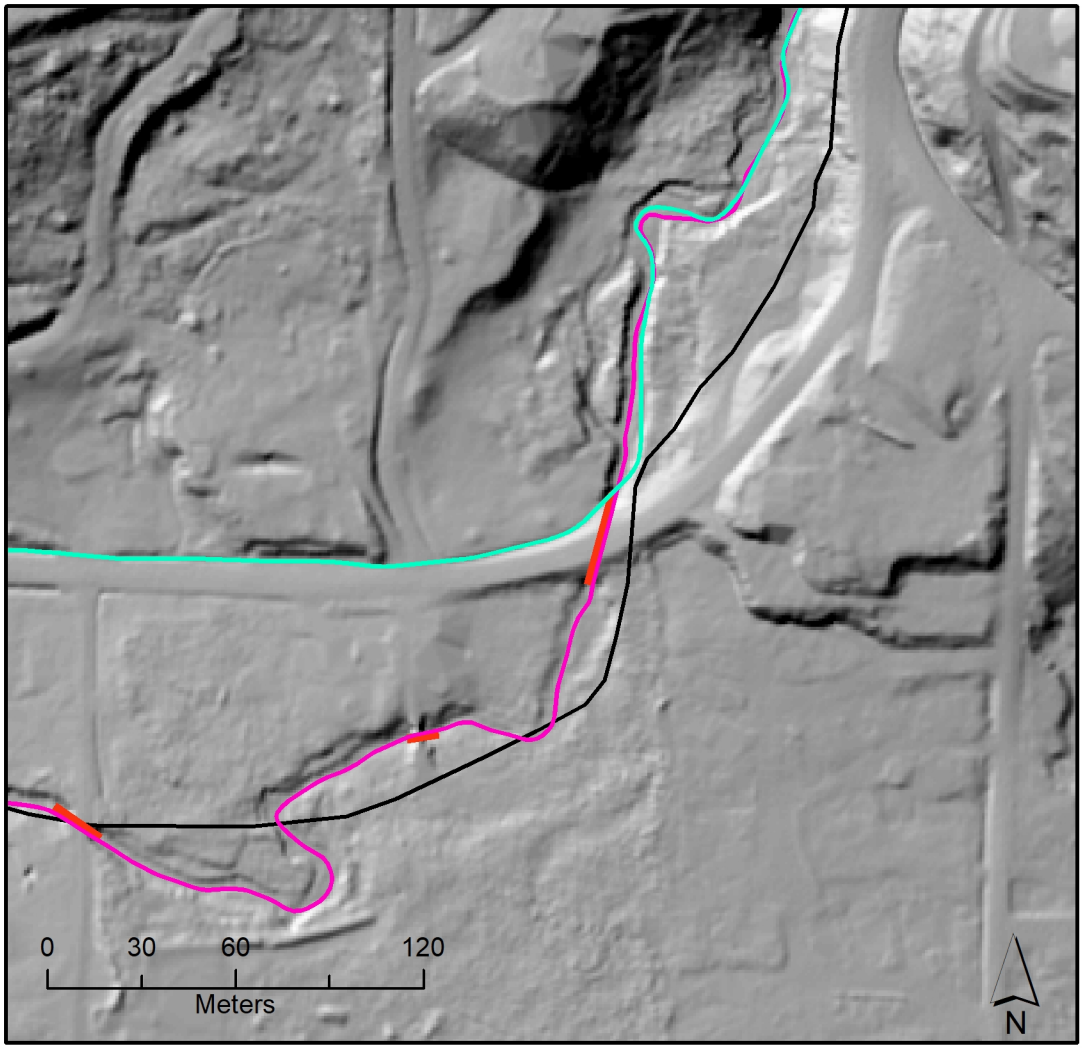


Fig __. First pass lidar-derived streams (cyan) are directed through road culverts and other obstructions by the (orange) trenches to correctly follow topography (magenta). The NHD streams (black) are shown for comparison.

The lidar-derived streams were combined with NHD arcs in the floodplain zones and/or manually edited to create a complete hydro data set. Manual editing was necessary at lakes, ponds, wetlands, floodplains, and other closed depressional areas to correctly align single-thread

hydrography to known routes visible on aerial photographs and in the field. The standard ArcGIS hydrography tools (Fill, Flow Direction, Flow Accumulation) do not function well in areas of low topographic relief and, in the Skagit LDH case, were corrected manually.

Channel bank outlines for mainstems, larger tributaries, and ponds and wetlands were created under a separate effort. This document describes single-thread hydrography only.

Smoothing the single-thread hydrography

The single-thread hydrography dataset then went through a series of algorithms to smooth and add additional physical attributes. In following the cell-by-cell path of steepest descent, the D8 flow routing algorithm produces unrealistically-sinuous flow paths, especially in low gradient reaches. The streamlines were smoothed to 1) produce more realistic reach lengths (and therefore more realistic gradients), and 2) to produce a more realistic looking cartographic effect. The Smooth Lines geoprocessing tool was used in ArcGIS 10.7, which allows the user to specify the maximum distance a vertex will be moved in the smoothing process (referred to as the 'smoothing tolerance'). The algorithm used by this tool keeps reach endpoint vertices fixed. Because stream junctions are mapped as reach endpoints in the mapping scheme, this feature ensures all confluences remain connected during smoothing. After an exploration of several smoothing schemes (uniform smoothing using a variety of fixed smoothing tolerances, varying the smoothing tolerance based on stream order) each reach was smoothed using a tolerance of 50 ft. because it was the minimum tolerance that produced a visually-realistic cartography. In comparison to the unsmoothed lines, the median percent increase in channel gradient using a tolerance of 50 ft was 12% (Fig --) for a test sub-basin.

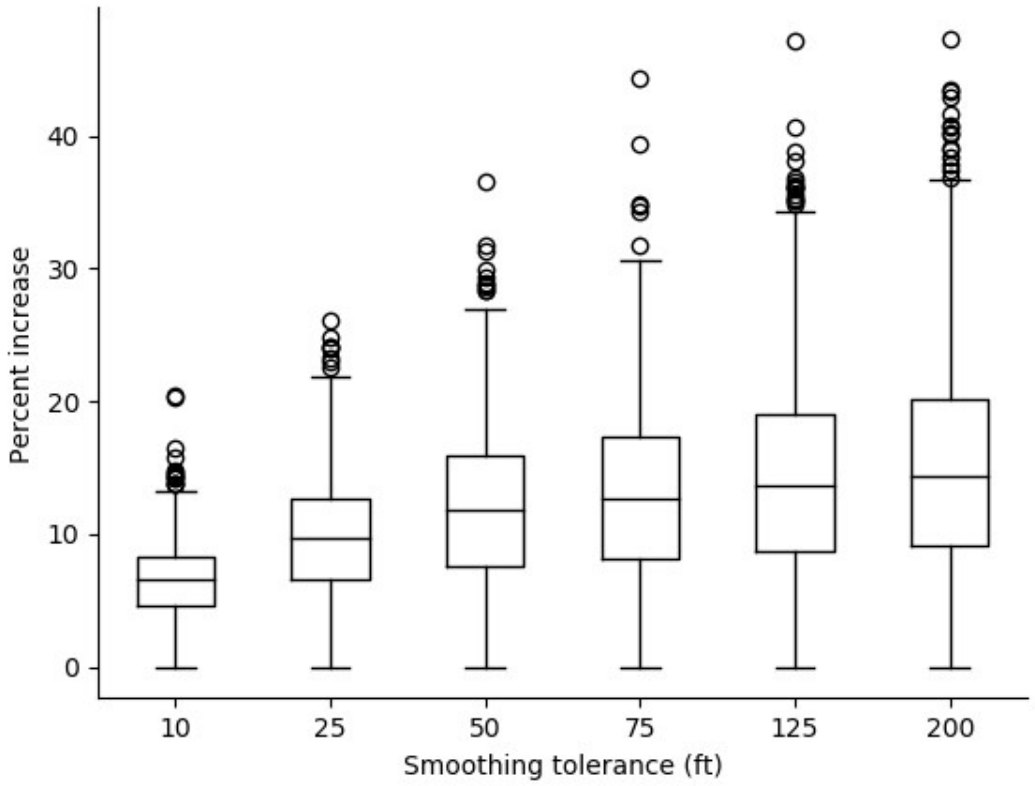


Figure --. Percent increase in channel gradient due to smoothing of the streamlines using a range of smoothing tolerances for a subset of the hydrographic dataset (the Loretta Creek-Skagit River HUC).

Assigning physical stream attributes

A combination of ArcGIS geoprocessing tools and python routines was used to calculate contributing drainage area, gradient, and bankfull channel width for each reach. Channel gradient was computed by dividing the difference between elevations at reach endpoints (pulled from the lidar DEM) by the reach length (smoothed with a tolerance of 50 ft as discussed above).

Other attributes, such as watershed area and accumulated precipitation, were calculated using the National Elevation Dataset DEM (10m XY resolution). The hydrographic network was first 'burned' into the 10m DEM by subtracting 5m from all DEM cells that were overlain by a stream pixel. The standard ArcGIS flow routing tools were used to calculate flow accumulation. An additional flow accumulation weighted by PRISM gridded precipitation data (PRISM Climate Group 2014) was performed to estimate total upstream precipitation. Mean upstream precipitation is the accumulated precipitation divided by the unweighted flow accumulation.

To calculate channel bankfull width, regression equations were used from both Davies et al. (2007) and Hyatt et al. (2004). The Davies et al. equation is

$$BFW = 0.04(A^{.048})\ (P^{0.74}),$$

where BFW is bankfull channel width, A is contributing drainage area, and P is the mean upstream precipitation. The Hyatt equation is

$$BFW = \exp[-5.52 + (0.46P) + (0.23\text{gradcode})],$$

where P in this case is the natural logarithm of the cumulative upstream precipitation and gradcode = 0 where channel gradient < 0.04 ft/ft and gradcode = 1 where gradient > 0.04 ft/ft.

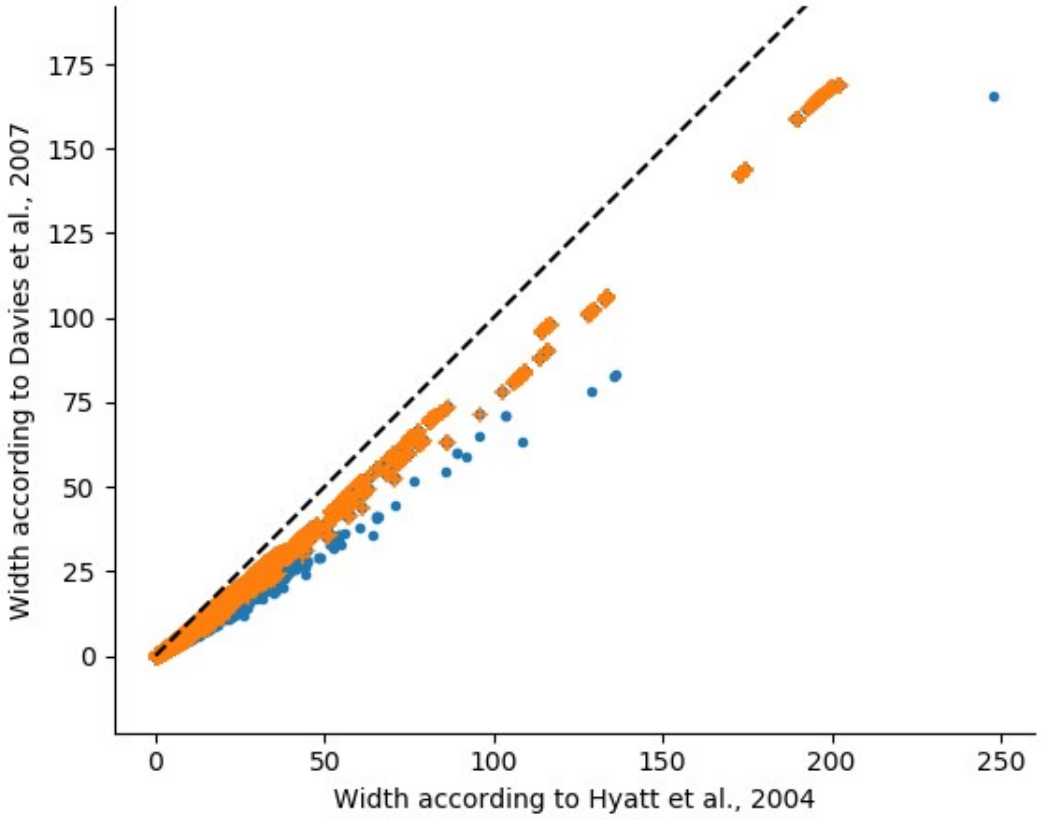


Figure --. Comparison of the two channel width equations used. Blue dots represent reaches where gradcode = 1 in the Hyatt width equation (higher gradient reaches) and orange symbols represent reaches where gradcode = 0 (low gradient reaches). The 1:1 line is shown as a dashed black line.

A comparison of the two width equations demonstrates that the Hyatt equation consistently predicts larger widths than the Davies equation (Fig. --). The difference between the two equations is more pronounced at wider channels and in higher sloped reaches, the latter being a function of the gradcode feature of the Hyatt equation. This discrepancy could be explained by the differing geographic extents of the two studies: The Davies equation was calibrated to sites across Puget Sound and Olympic Coast drainage basins (with only six sites in the Skagit River basin), while the Hyatt equation focused on sites in the Skagit, Nooksack and Stillaguamish basins. Without additional field work to verify the two equations, it was assumed the Hyatt equation is the most appropriate for the Skagit River basin hydrography. Both width estimates remain in the attribute table for future testing.

Questions regarding these methods should be directed to:

Tim Hyatt, Skagit River System Cooperative, thyatt@skagitcoop.org

References cited

Davies, J.R., Lagueux, K.M., Sanderson, B. and Beechie, T.J., 2007. Modeling Stream Channel Characteristics From Drainage-Enforced DEMs in Puget Sound, Washington, USA 1. JAWRA Journal of the American Water Resources Association, 43(2), pp.414-426.

Hyatt, T.L., T.Z. Waldo, and T.J. Beechie. 2004. A watershed-scale assessment of riparian forests, with implications for restoration. *Restoration Ecology* 12(2): 175-183

PRISM Climate Group, Oregon State University, https://prism.oregonstate.edu, data created 4 Feb 2014, accessed 16 June 2021.

Example Python script for generating hydrography that bypasses culverts and other hydromodifications. Paste into IDLE or other Python console for use.

```
# -*- coding: utf-8 -*-
# ---------------------------------------------------------------------------
# Hydro_8.py
# Created on: 20180809      last modified 20220202
# Runs the Hydro model from an input DEM to output arcs
#   burns in the culvert crossings where sinks don't fill properly
#   converts the culvert lines to polygons to create a broader "well" that is filled
#   This version 8 is edited somewhat for release in the methods document

# Sequence for flow is FillSinks, FlowDirction, FlowAccumulation, then create arcs as vectors
# Earlier versions would smooth the arcs, which was time consuming and is now left for later

# Syntax: Buffer_analysis (in_features, out_feature_class, buffer_distance_or_field,
#       {line_side}, {line_end_type}, {dissolve_option}, {dissolve_field}, {method})
# Syntax: ZonalStatistics (in_zone_data, zone_field, in_value_raster, {statistics_type},
          {ignore_nodata})
# Syntax: Con (in_conditional_raster, in_true_raster_or_constant, {in_false_raster_or_constant},
          {where_clause})
# Syntax: Fill (in_surface_raster, {z_limit})
# Syntax: FlowDirection (in_surface_raster, {force_flow}, {out_drop_raster})
# Syntax: FlowAccumulation (in_flow_direction_raster, {in_weight_raster}, {data_type})
# Syntax: CalculateStatistics_management (in_raster_dataset, {x_skip_factor}, {y_skip_factor},
          {ignore_values}, {skip_existing}, {area_of_interest})
# Syntax: StreamToFeature (in_stream_raster, in_flow_direction_raster, out_polyline_features,
          {simplify})
# ---------------------------------------------------------------------------
# Import arcpy module
import arcpy, datetime
from arcpy import env
from arcpy.sa import *

# Set the workspace and environment
arcpy.env.workspace = r"C:\TLHwork\GISprojects\Hydro\Hydro8.gdb"
arcpy.env.overwriteOutput = True
arcpy.env.cellSize = "MINOF"

# Check out any necessary licenses
arcpy.CheckOutExtension("spatial")

startTime = datetime.datetime.now()
print("starting at " + str(startTime))

# Choose your favorite input DEM and set the output DEM
```

```
InputDEM = r"E:\LiDAR\Skagit_2020\skagit_dem"
InDEMmask = "hull_CasClip"
DEMclip = "CasClip1"

print("Extracting DEM by mask")
# If you don't already have a local grid, then first use ExtractByMask
outExtractByMask = ExtractByMask(InputDEM, InDEMmask)
outExtractByMask.save(DEMclip)

# Set your input and output filenames here
site = "CascTest"
version = "_a"
wigglyFC = site + "_125k" + version  # this is the output hydro file

# Local variables to extract elevations to the Culvert FC:
CrossFC = "Trench_merge_20191125"
culvPoly = "t_poly"
buffDist = "6"
Field = "OBJECTID"

# Hydro Variables
trenched = "t_trench"
outFill = "t_Fill"
outFdir = "t_Fdir"
outFacc = "t_Facc"
zLimit = "" # if blank then all sinks will be filled regardless of depth
inWeightRaster = ""
dataType = "INTEGER"
ifTrue = "1"
ifFalse = "" #means "NULL" or "NO_DATA"
whereClause = "VALUE >= 125000"  # 125k 3x3 pixels about matches USGS 7.5 min topo
StreamRaster = "t_hydropixel"

print "Variables are set. Now buffering trenches"

# First, buffer each of the culverts by 6 feet to broaden the "well"
arcpy.Buffer_analysis(CrossFC, culvPoly, buffDist, "FULL", "ROUND", "NONE", "")

print("culverts are buffered, now calculating MIN elevations and extracting temp grid")
# this step finds the lowest elevation within the trench buffer
outRAS = ZonalStatistics(culvPoly, Field, DEMclip,"MINIMUM", "NODATA")

# Con statement keeps the DEMclip values where outRAS is Null, and the outRAS values where
        not null
arcpy.env.extent = DEMclip
outCon = Con(IsNull(outRAS), DEMclip, outRAS)
```

```
# outCon.save(Raster(r"C:\Projects\Hydro\Trenches.gdb\t_trench"))

print("Filling sinks")
# Start here if you're not trenching the culverts; that is, if you're running the hydro for the first
        time
# if using Extract By Mask input = DEMclip or InputDEM; if using Trenches then outCon
#outFil = Fill(DEMclip, zLimit) ####
outFil = Fill(outCon, zLimit) ####
outFil.save(outFill)

print("Calculating Flow Direction")
outFlowDirection = FlowDirection(outFill, "NORMAL", "")
outFlowDirection.save(outFdir)

print("Calculating Flow Accumulation (this takes a while)")
outFlowAccumulation = FlowAccumulation(outFdir, inWeightRaster, dataType)
outFlowAccumulation.save(outFacc)

print("Flow Accumulation finished, now creating stream arcs")
# Calc statistics so the Con statement works over the correct range of data
arcpy.CalculateStatistics_management(outFacc)

# Use a Con statement to create a raster of stream pixels
outCon = Con(outFacc, ifTrue, ifFalse, whereClause)
outCon.save(StreamRaster)

# Use the stream pixels to create arc features
StreamToFeature(StreamRaster, outFdir, wigglyFC, "NO_SIMPLIFY")

print("Done!")

endTime = datetime.datetime.now()
stringTime = endTime.strftime('%Y%m%d %H:%M')
print("finished at " + str(endTime))
```