Automation Of Spatial Data Validation

Using ArcGIS Pro, Pandas And Other Technologies, The National Hydrographic Dataset High Resolution (NHDPlusHR), National Geospatial Technical Operations Center (NGTOC), USGS

Greg Cocks | (Research) Spatial Data Scientist | Contractor Formerly Supporting The USGS (Now Supporting NOAA)

gregcocks.kiwi@gmail.com | #AllDataIsSpatial | www.linkedin.com/in/gregcocks/

Presentation for the ESRI User Conference, 2021

Background

"The NHDPlusHR is a nationwide, integrated hydrography and elevation dataset that includes the NHD [National Hydrography Dataset] stream network, WBD [Watershed Boundary Dataset] hydrologic units, local drainage areas or "catchments", flow direction and flow accumulation rasters, and value-added attributes (VAAs)...

The NHDPlus HR... is built using the NHD at 1:24,000-scale or better, nationally complete WBD, and 1/3 arc-second seamless 3D Elevation Program (3DEP) data..."

nhd.usgs.gov ← *the short URL ~smile~*

Background (cont)

- As they continue to be generated, 220+ HUC4 (hydrologic units, level 4) NHD large spatial data sets across the contiguous US (and then more in Alaska) will need to be validated.
- These same 220+ data sets - as well as Alaska's - will need to be refreshed and then re-validated due to ongoing updates, edits & additions
- These are **large** datasets – with, for instance, **25+ million** vector lines in just **one** feature class (NHDFlowline) across CONUS alone
- The previous manual validation took inordinate amounts of time & resources and by its nature could usually only spot check (<< 1% records in some tests)

Outline

This presentation outlines 'autonomous' task(s), writing 75+ data validation scripts for the National Hydrography Plus High Resolution (NHDPlusHR) datasets – whilst working as a contractor in support of the USGS' National Geospatial Program (NGTOC.)

I sought out the work after seeing a need, researching independently - and then through managing the processes, research and investigation 'discovered' methods that allowed the various scripts to run across large datasets (i) with viable time-frames and (ii) within the resources of the available hard- and software.

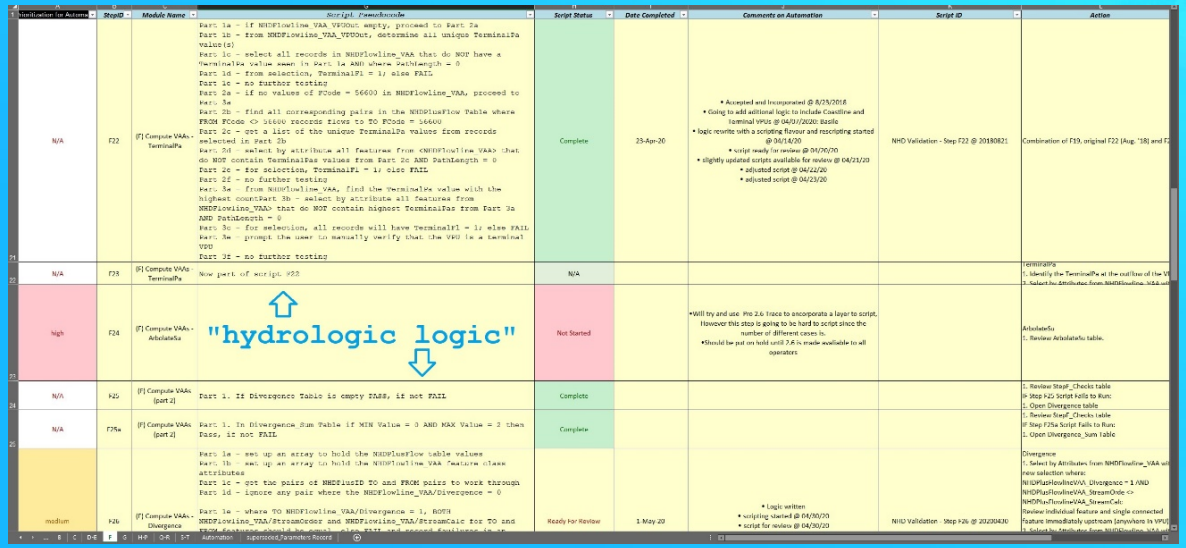
Disclaimer

This presentation is the author's 'good faith' explanation of the professional research and then production support task(s) performed - and may not reflect how the data's steward might choose to present these efforts

Pseudo-Code? Pseudo-Kanban? Both??

This generated resource was the primary connection:

- between those needing the work done, the various subject matter experts (hydrologists, etc) AND
- designing, writing, pro-typing, testing, and iteratively adjusting the various scripts

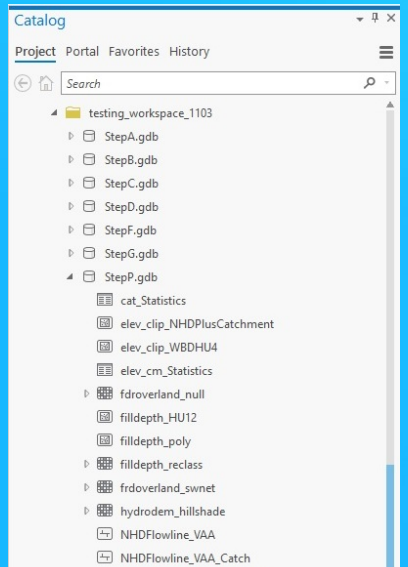


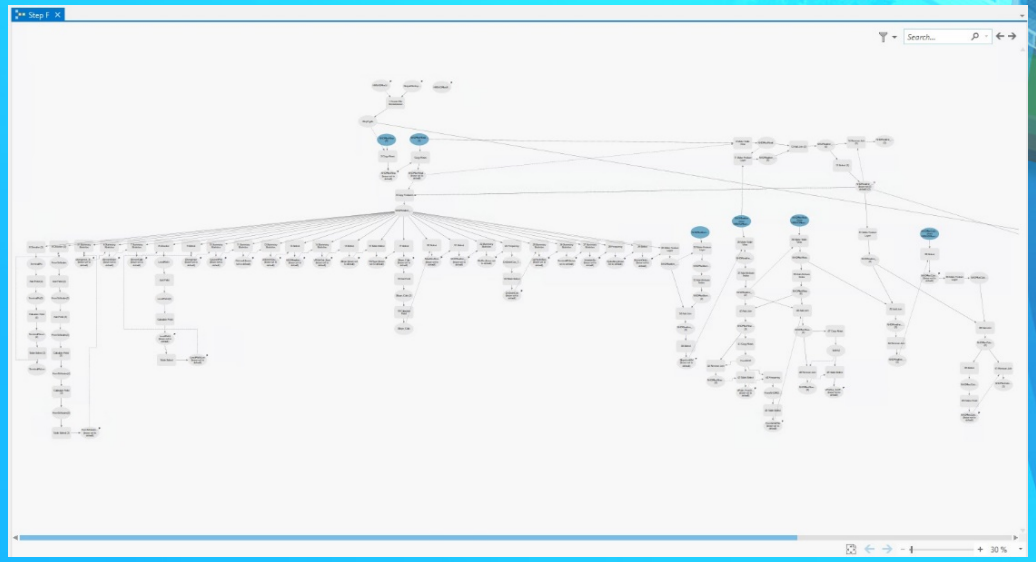
Prioritization for Automa	StepID	Module Name	Script Logic Pseudocode	Script Status	Date Completed	Comments on Automation	Script ID	Action
N/A	F22	[F] Compute VAAs - TerminalPa	Part 1a - if NHDFlowline_VAA_VPUOut empty, proceed to Part 2a Part 1b - from NHDFlowline_VAA_VPUOut, determine all unique TerminalPa value(s) Part 1c - select all records in NHDFlowline_VAA that do NOT have a TerminalPa value seen in Part 1a AND where PathLength = 0 Part 1d - from selection, TerminalFl = 1; else FAIL Part 1e - no further testing Part 2a - if no values of FCode = 56600 in NHDFlowline_VAA, proceed to Part 3a Part 2b - find all corresponding pairs in the NHDPlusFlow Table where FROM FCode <> 56600 records flows to TO FCode = 56600 Part 2c - get a list of the unique TerminalPa values from records selected in Part 2b Part 2d - select by attribute all features from <NHDFlowline_VAA> that do NOT contain TerminalPas values from Part 2c AND PathLength = 0 Part 2e - for selection, TerminalFl = 1; else FAIL Part 2f - no further testing Part 3a - from NHDFlowline_VAA, find the TerminalPa value with the highest countPart 3b - select by attribute all features from NHDFlowline_VAA> that do NOT contain highest TerminalPas from Part 3a AND PathLength = 0 Part 3c - for selection, all records will have TerminalFl = 1; else FAIL Part 3e - prompt the user to manually verify that the VPU is a terminal VPU Part 3f - no further testing	Complete	25-Apr-20	• Accepted and incorporated @ 8/25/2018 • Going to add additional logic to include Coastline and terminal VPUs @ 04/0//2020: Basile • logic rewrite with a scripting flavour and rescripting started @ 04/14/20 • script ready for review @ 04/20/20 • slightly updated scripts available for review @ 04/21/20 • adjusted script @ 04/22/20 • adjusted script @ 04/23/20	NHD Validation - Step F22 @ 20180821	Combination of F19, original F22 (Aug. '18) and F
N/A	F23	[F] Compute VAAs - TerminalPa	Now part of script F22	N/A				TerminalPa 1. Identify the TerminalPa of the outflow of the V
high	F24	[F] Compute VAAs - ArbolateSu	"hydrologic logic"	Not Started		•Will try and use Pre 2.6 trace to incorporate a layer to script. However this step is going to be hard to script since the number of different cases is. •Should be put on hold until 2.6 is made available to all operators		ArbolateSu 1. Review ArbolateSu table.
N/A	F25	[F] Compute VAAs (part 2)	Part 1. If Divergence Table is empty PASS, if not FAIL	Complete				1. Review StepF_Checks table IF Step F25 Script Fails to Run: 1. Open Divergence table
N/A	F25a	[F] Compute VAAs (part 2)	Part 1. In Divergence_Sum Table if MIN Value = 0 AND MAX Value = 2 then Pass, if not FAIL	Complete				1. Review StepF_Checks table IF Step F25a Script Fails to Run: 1. Open Divergence_Sum table
medium	F26	[F] Compute VAAs - Divergence	Part 1a - set up an array to hold the NHDPlusFlow table values Part 1b - set up an array to hold the NHDFlowline_VAA feature class attributes Part 1c - get the pairs of NHDPlusID TO and FROM pairs to work through Part 1d - ignore any pair where the NHDFlowline_VAA/Divergence = 0 Part 1e - where TO NHDFlowline_VAA/Divergence = 1, BOTH NHDFlowline_VAA/StreamOrder and NHDFlowline_VAA/StreamCalc for TO and FROM features should be equal; else FAIL and record divergence is or	Ready For Review	1-May-20	• Logic written • scripting started @ 04/30/20 • script for review @ 04/30/20	NHD Validation - Step F26 @ 20200430	Divergence 1. Select by Attributes from NHDFlowline_VAA wi new selection where: NHDPlusFlowlineVAA_Divergence = 1 AND NHDPlusFlowlineVAA_StreamOrde <> NHDPlusFlowlineVAA_StreamCalc Review individual feature and single connected feature immediately upstream (anywhere in VPU 1. Select by Attributes from NHDFlowline_VAA wi

It is also being used as a reference by those bringing the scripts into the production environment, as intended (see later.)

Production To Interim FGDBs

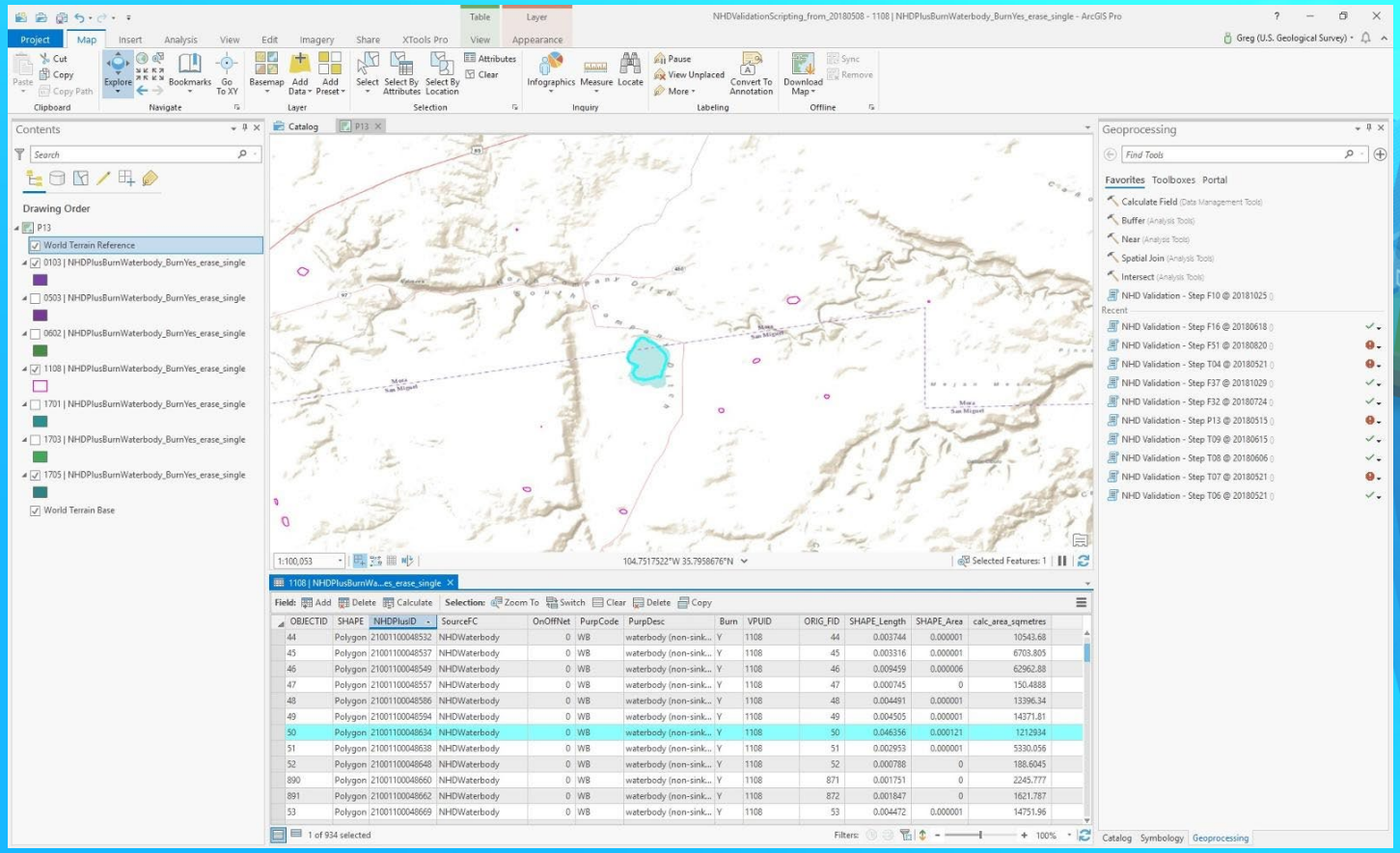
Various interim / scratch file geodatabases (FGDB) were generated (in ModelBuilder) by another party for every step of the validation (matching the NHDPlusHR data production Steps A → U)





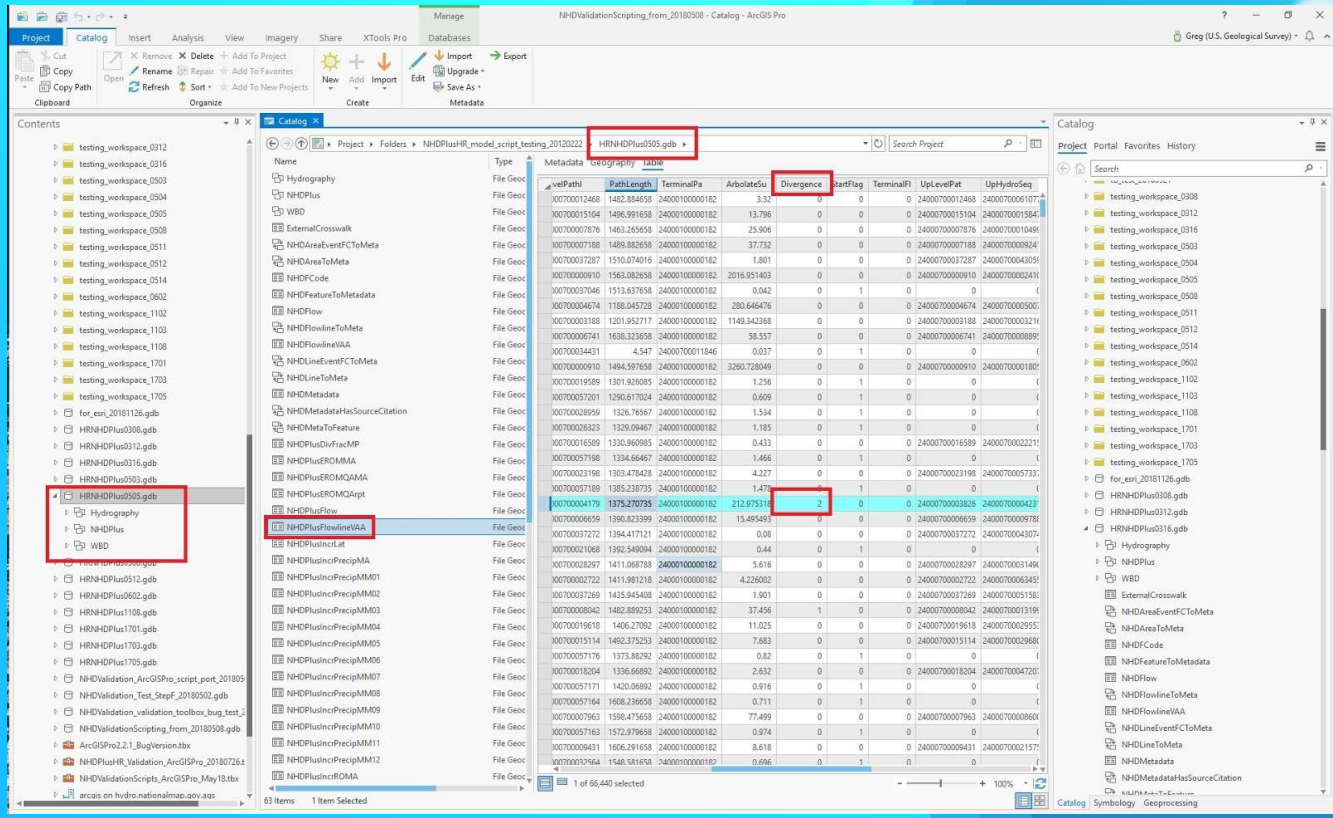
These ModelBuilder FGDB generation processes are to be replaced with Python

Spatial Overview



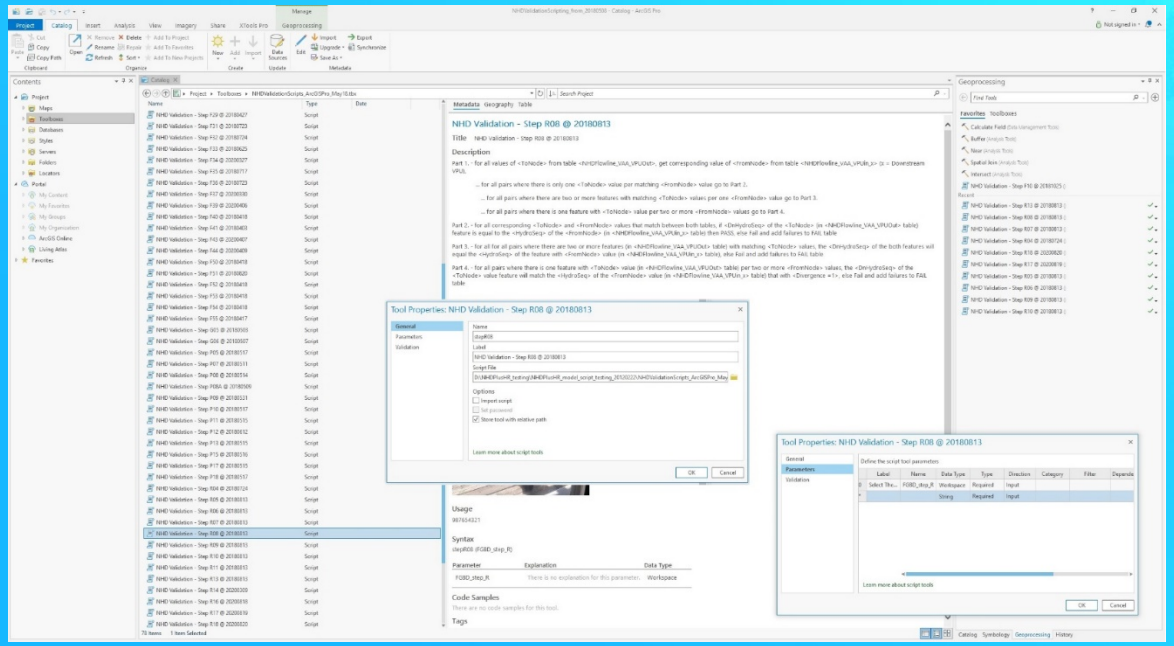
‘Nature Of The Data’

- Initially working with vectors, vector attributes and linked tables (like VAAs)
- Later planned with geometric network and rasters



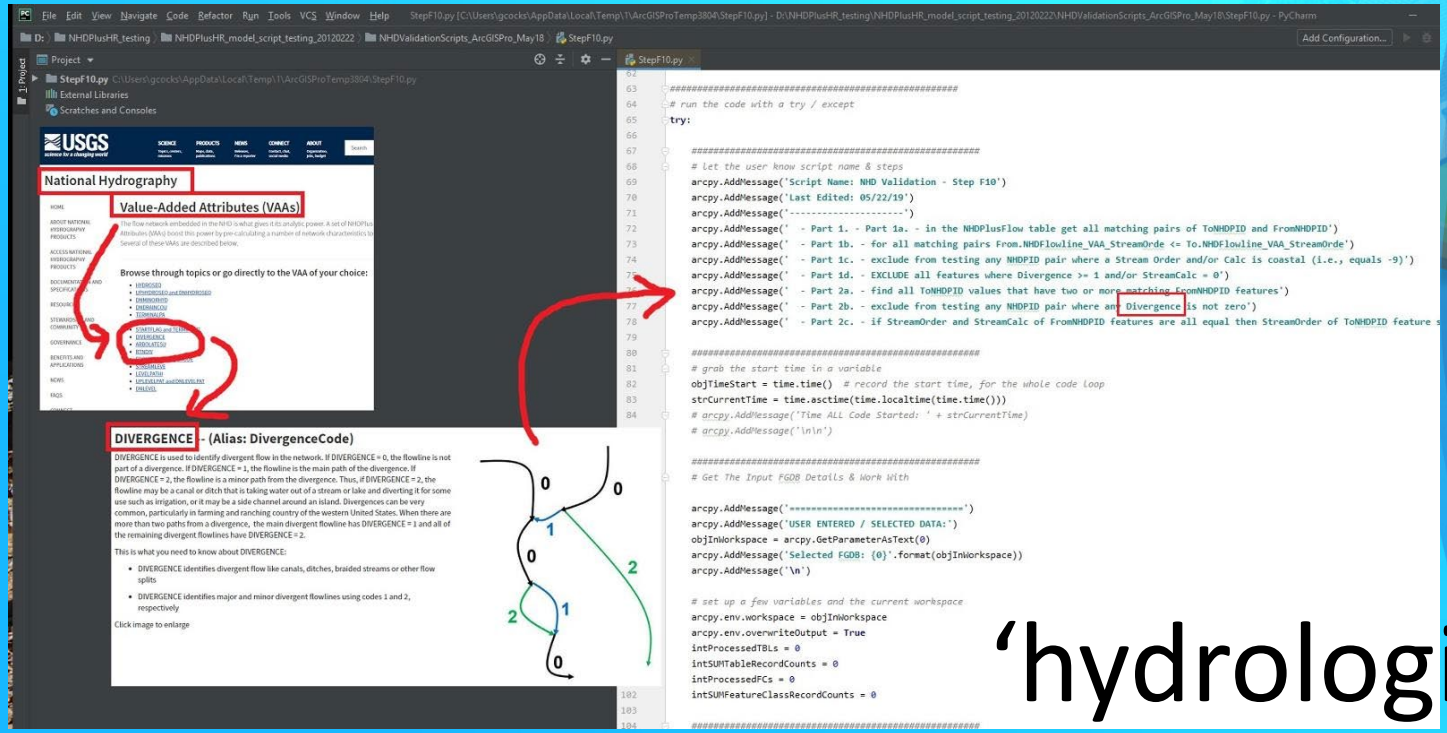
The Scripting Environment

- ArcGIS Pro 2.x with PyCharm IDE accessing Python 3.x
- Try to use standard ‘Pro’ install so don’t have to set-up / propagate custom project environments (*)



(*) GeoPandas might ‘break’ this intention, see later

ALWAYS Kept in Mind The 'Rules' Of The Hydrologic Science Driving The Validation…



‘hydrologic logic…’

Documentation!
Win Lotto? Hit By Bus?

main.py | StepR08.py

```
# -------------------------------------------------------------------------
#
#  Name:          NHD Validation - Step R08
#
#  Purpose:       Part 1. - for all values of <ToNode> from table <NHDFlowline_VAA_VPUOut>, get corresponding value of <FromNode> from table <NHD
#                           ... for all pairs where there is only one <ToNode> value per matching <FromNode> value go to Part 2.
#                           ... for all pairs where there are two or more features with matching <ToNode> values per one <FromNode> value go to P
#                           ... for all pairs where there is one feature with <ToNode> value per two or more <FromNode> values go to Part 4.
#                 Part 2. - for all corresponding <ToNode> and <FromNode> values that match between both tables, if <DnHydroSeq> of the <ToNode>
#                 Part 3. - for all for all pairs where there are two or more features (in <NHDFlowline_VAA_VPUOut> table) with matching <ToNode>
#                 Part 4. - for all pairs where there is one feature with <ToNode> value (in <NHDFlowline_VAA_VPUOut> table) per two or more <Fro
#
#  Note:          - script name is based on (simplified) test plan dated 20180412 (and can change)
#
#  Author:        Greg Cocks
#                 gcocks@usgs.gov
#
#  Created:       08/13/18
#  Last Edited:   08/24/20
#
#  Section:       NGTOC, USGS
#
# -------------------------------------------------------------------------

import ...


#########################################################
# FUNCTIONS
```

main.py × StepP13.py ×

```
178
179  # Part 1a. - subtract elev_cm raster from hydrodem raster
180  objTimePartStart = time.time()
181  arcpy.CheckOutExtension('Spatial')
182  arcpy.AddMessage('---------------------------')
183  arcpy.AddMessage('Part 1a  --->  subtract elev_cm raster from hydrodem raster')
184  objRasterSubtract = arcpy.sa.Raster(objRasterHydroDEM) - arcpy.sa.Raster(objRasterElevCm)
185  if arcpy.Exists(os.path.join(objInWorkspace, strRasterSubtract)):
186      arcpy.Delete_management(os.path.join(objInWorkspace, strRasterSubtract))
187  objRasterSubtract.save(os.path.join(objInWorkspace, strRasterSubtract))
188  objTimePartEnd = time.time()
189  strTimeMessage = '   * it took {} to execute Part 1a code'.format(funhmsstring(objTimePartEnd - objTimePartStart))
190  arcpy.AddMessage(strTimeMessage)
191  arcpy.AddMessage(strBlank)
192
193  # Part 1b. - raster reclassify result to make zero and positive values as NoData
194  objTimePartStart = time.time()
195  arcpy.AddMessage('---------------------------')
196  arcpy.AddMessage('Part 1b  --->  raster reclassify result to make zero and positive values as NoData')
197  objRasterSubtractReclass = arcpy.sa.Reclassify(objRasterSubtract, strRasterField, '-100000000 -0.000100 1;-0.000100 10000000000 NODATA', 'NODATA')
198  if arcpy.Exists(os.path.join(objInWorkspace, strRasterSubtractReclass)):
199      arcpy.Delete_management(os.path.join(objInWorkspace, strRasterSubtractReclass))
200  objRasterSubtractReclass.save(os.path.join(objInWorkspace, strRasterSubtractReclass))
201  arcpy.CheckInExtension('Spatial')
202  arcpy.Delete_management('in_memory')  # just in case
203  objTimePartEnd = time.time()
204  strTimeMessage = '   * it took {} to execute Part 1b code'.format(funhmsstring(objTimePartEnd - objTimePartStart))
205  arcpy.AddMessage(strTimeMessage)
206  arcpy.AddMessage(strBlank)
207
208  # Part 1c. - convert this raster to a polygon
209  arcpy.AddMessage('---------------------------')
210  arcpy.AddMessage('Part 1c  --->  convert this raster to a polygon')
211  # get the spatial reference of the NHDPlusBurnLineEvent FC <strFlowlines> in the VPU FGDB
212  objDesiredSpatialRef = arcpy.Describe(os.path.join(objVPUWorkspace, strFDSOfInterest01, strFlowlines)).spatialReference
213  # convert the raster to poly, in the just determined spatial reference
214  objTimePartStart = time.time()
215  arcpy.RasterToPolygon_conversion(objRasterSubtractReclass, os.path.join(objInWorkspace, strRasterToPolygonRaw), 'NO_SIMPLIFY', strRasterField)
216  objTimePartEnd = time.time()
217  strTimeMessage = '   * it took {} to execute Part 1c code - Raster To Polygon'.format(funhmsstring(objTimePartEnd - objTimePartStart))
218  # re-project the result to desired
219  objTimePartStart = time.time()
220  arcpy.Project_management(os.path.join(objInWorkspace, strRasterToPolygonRaw), os.path.join(objInWorkspace, strRasterToPolygon), objDesiredSpatialRef)
221  del strRasterToPolygonRaw
222  objTimePartEnd = time.time()
223  strTimeMessage = '   * it took {} to execute Part 1c code - Re-Project Raster'.format(funhmsstring(objTimePartEnd - objTimePartStart))
224  arcpy.AddMessage(strTimeMessage)
225  # # 'just in case' repair FC
226  # arcpy.AddMessage('  - repair geometry of just created FC \'just in case\'')
227  # arcpy.RepairGeometry_management(os.path.join(objInWorkspace, strRasterToPolygon))
228  # arcpy.Delete_management('in_memory')  # just in case
229  # objTimePartEnd = time.time()
230  # strTimeMessage = '   * it took {} to execute Part 1c code - repair geometry'.format(funhmsstring(objTimePartEnd - objTimePartStart))
```

Kept The 'Connection' With The '1st Principles Science'

- Commenting with the 'hydrologic logic', etc

StepF10.py

PD — Match Case | Words | Regex | 45 matches

```
strFCOfInterest = 'NHDFlowline_VAA'
strFCField1 = 'NHDFlowline_NHDPlusID'
intFCField1 = 0
strFCField1DataType = 'int64'  # double
strFCField2 = 'NHDPlusFlowlineVAA_StreamOrde'  # <-- data type: short (int8)
intFCField2 = 1
strFCField2DataType = 'int8'  # short
strFCField3 = 'NHDPlusFlowlineVAA_Divergence'  # <-- data type: short
intFCField3 = 2
strFCField3DataType = 'int8'  # short
strFCField4 = 'NHDPlusFlowlineVAA_StreamCalc'  # <-- data type: short
intFCField4 = 3
strFCField4DataType = 'int8'  # short

intNULLDesignator = -123456789  # as NumPy doesn't like NULLs very much, datatype = 'i10'
intCoastalStreamDesignator = -9

# set up the table of interest's selected attributes in a NumPy array
lstTableFields = [strTableofInterestField01, strTableofInterestField02]
npaTblNHDPlusFlowRaw = arcpy.da.TableToNumPyArray(os.path.join(objInWorkspace, strTableofInterest), lstTableFields, null_value=intNULLDesignator)
lstTableDataTypes = [(strTableofInterestField01, intTableofInterestField01DataType), (strTableofInterestField02, intTableofInterestField02DataType)]
npaTblNHDPlusFlow = npaTblNHDPlusFlowRaw.astype(lstTableDataTypes)
del npaTblNHDPlusFlowRaw
arcpy.AddMessage('Step F Table: {} - Record Count: {:,}'.format(strTableofInterest, len(npaTblNHDPlusFlow)))

# set up the feature class of interest's selected attributes in a NumPy array
lstFCFields = [strFCField1, strFCField2, strFCField3, strFCField4]
npaFCFlowlineVAARaw = arcpy.da.FeatureClassToNumPyArray(os.path.join(objInWorkspace, strFCOfInterest), lstFCFields, null_value=intNULLDesignator)
lstFCDataTypes = [(strFCField1, strFCField1DataType), (strFCField2, strFCField2DataType), (strFCField3, strFCField3DataType), (strFCField4, strFCField4DataType)]
npaFCFlowlineVAA = npaFCFlowlineVAARaw.astype(lstFCDataTypes)
npaFCFlowlineVAA = np.asanyarray(npaFCFlowlineVAA)
del npaFCFlowlineVAARaw
arcpy.AddMessage('Step F    FC: {} - Record Count: {:,}'.format(strFCOfInterest, len(npaFCFlowlineVAA)))
arcpy.AddMessage('----------')

# ###################################################

# convert the table numpy array to a pandas data frame so as to better manipulate
pdaTblNHDPlusFlow = pd.DataFrame(npaTblNHDPlusFlow)
arcpy.AddMessage('pandas Table Length: {:,}'.format(len(pdaTblNHDPlusFlow)))

# convert the FC numpy array to a pandas data frame so as to better manipulate
pdaFCFlowlineVAA = pd.DataFrame(npaFCFlowlineVAA)
# sort the resultant pandas data frame to (potentially) help with speed
pdaFCFlowlineVAA = pdaFCFlowlineVAA.sort_values(by=strFCField1)  # sorting by the NHDPlusID
arcpy.AddMessage('pandas FC Length: {:,}'.format(len(pdaFCFlowlineVAA)))
arcpy.AddMessage('----------')

# Part 1a. - in the NHDPlusFlow table get all matching pairs of ToNHDPID and FromNHDPID')
```

Feature Class

FC Attributes --> NumPy

GeoPandas????

NumPy --> pandas

Keep Learning, Exploring – And Trying!

Keep Learning, Improving And Exploring (cont)

- For example, through autonomous research and then learning, took advantage of:
  1. ArcGIS Pro's parallel processing GP tools (as more and more 'came on line')
  2. The Dice GP tool to be able to manage the 'Godzilla' features with available resources (soft- & hardware)

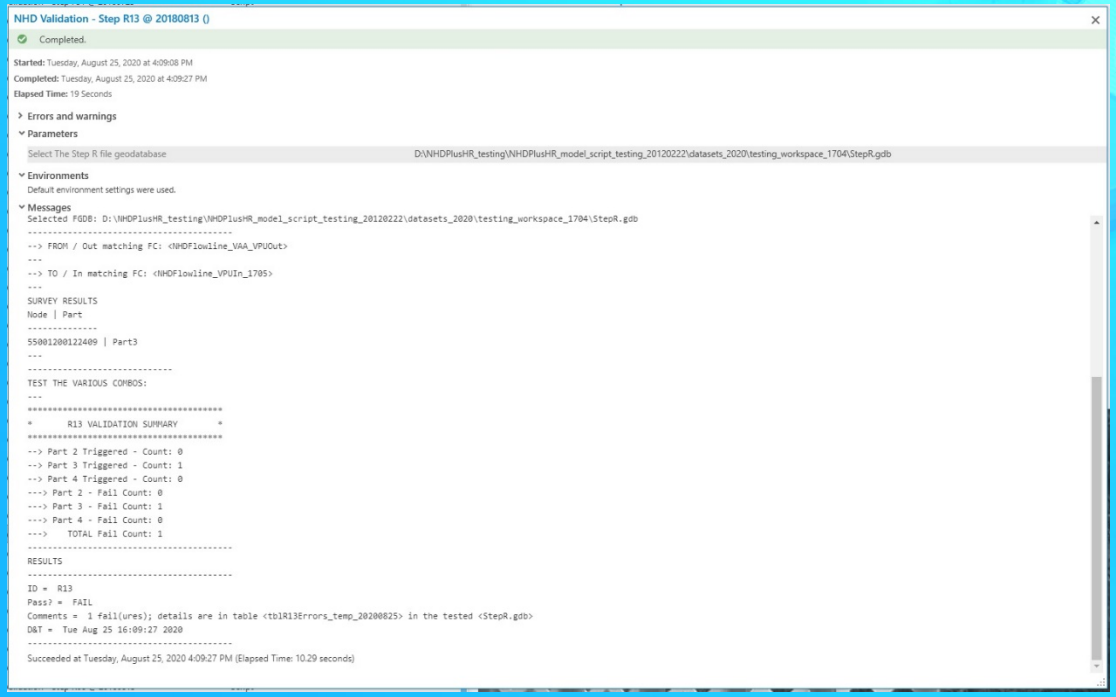
```
# various parameters, thresholds, etc
intBufferDistance = 15  # <-- in metres, broadly the diagonal length of a 10m raster cell
intBufferTolerance = 0.25  # <-- in metres
strBufferDistance = '{} meters'.format(intBufferDistance)
strBufferTolerance = '{} meters'.format(intBufferTolerance)
intDistanceOfInterest = 15  # <-- Step 1g, stream partial lengths greater than this. in metres
intAreaOfInterest = 100  # <-- Step 1l, waterbody portions greater
intDiceLimit = 1000000  # <-- the limit on the Dice Vertices, to he
```

```
# set up a few variables and the current workspace
arcpy.env.workspace = objInWorkspace  # not always a 100% honoured!
arcpy.env.overwriteOutput = True
arcpy.env.parallelProcessingFactor = '98%'  # Use 98% (i.e.  all) of the cores on the machine for applicable GP tools
```

```
# Part 1d. - buffer this polygon by 15 metres (i.e., broadly the diagonal of one 10m raster cell)
objTimePartStart = time.time()
arcpy.AddMessage('---------------------------')
arcpy.AddMessage('Part 1d  --->  buffer this polygon by {} metres'.format(intDistanceOfInterest))
arcpy.Dice_management(os.path.join(objInWorkspace, strRasterToPolygon), os.path.join(objInWorkspace, strRasterToPolygonDiced), intDiceLimit)
# using pairwise buffer to take advantage of parallel processing
arcpy.analysis.PairwiseBuffer(os.path.join(objInWorkspace, strRasterToPolygonDiced), os.path.join(objInWorkspace, strRasterToPolygonBuffer), strBufferDista
objTimePartEnd = time.time()
strTimeMessage = '    * it took {} to execute Part 1d code | create dissolved buffers (pairwise)'.format(funhmsstring(objTimePartEnd - objTimePartStart))
arcpy.AddMessage(strTimeMessage)
```

User Informed As The Script Is Running

- May not be see in automated systems - but can be accessed in saved text files for any problematic HUC4 dataset validation steps



Updated Table – Overall Summary Of Step's Scripts

StepR_Checks

Field: Add | Calculate | Selection: Select By Attributes | Zoom To | Switch | Clear | Delete | Copy

OBJECTID *	CheckID	Result	Comments	TestDateTime
55	R05	Pass		Wed Aug 19 17:54:14 2020
54	R06	Pass		Wed Aug 19 17:45:15 2020
53	R09	Pass		Wed Aug 19 17:30:10 2020
52	R10	Pass		Wed Aug 19 17:25:18 2020
51	R11	Pass		Wed Aug 19 17:07:49 2020
50	R16	Pass		Wed Aug 19 17:04:15 2020
9	R05	Pass		Tue Jun 16 19:10:37 2020
8	R05	Pass		Tue Jun 16 19:07:43 2020
7	R05	FAIL	1 failures; details in table <tblR05Errors_temp_20200616> in tested <StepR.gdb>	Tue Jun 16 19:02:15 2020
6	R05	Pass		Tue Jun 16 18:51:12 2020
5	R05	Pass		Tue Jun 16 18:49:42 2020
4	R05	Pass		Tue Jun 16 16:54:42 2020
3	R05	Pass		Tue Jun 16 16:53:12 2020
2	R05	Pass		Tue Jun 16 13:20:04 2020
11	R06	Pass		Tue Jul 14 18:51:14 2020
77	R13	FAIL	1 fail(ures); details are in table <tblR13Errors_temp_20200825> in the tested <Ste...	Tue Aug 25 16:15:04 2020
76	R13	FAIL	1 fail(ures); details are in table <tblR13Errors_temp_20200825> in the tested <Ste...	Tue Aug 25 16:10:36 2020
75	R13	FAIL	1 fail(ures); details are in table <tblR13Errors_temp_20200825> in the tested <Ste...	Tue Aug 25 16:09:27 2020
74	R13	Pass		Tue Aug 25 15:37:14 2020
49	R16	Pass		Tue Aug 18 18:06:02 2020
48	R16	Pass		Tue Aug 18 17:57:12 2020
47	R11	FAIL	1 fail(ures); details are in table <tblR11Errors_temp_20200818> in the tested <Ste...	Tue Aug 18 16:09:54 2020
46	R11	FAIL	1 fail(ures); details are in table <tblR11Errors_temp_20200818> in the tested <Ste...	Tue Aug 18 16:08:42 2020
45	R11	FAIL	1 fail(ures); details are in table <tblR11Errors_temp_20200818> in the tested <Ste...	Tue Aug 18 16:07:22 2020
1	R05	Pass		Thu Jun 11 17:33:21 2020
22	R09	Pass		Thu Jul 30 20:14:27 2020
21	R09	SCRIPT FAIL	name 'lstMatches' is not defined	Thu Jul 30 20:13:46 2020
20	R09	Pass		Thu Jul 30 20:05:04 2020
19	R09	Pass		Thu Jul 30 20:03:29 2020
18	R09	SCRIPT FAIL	index 0 is out of bounds for axis 0 with size 0	Thu Jul 30 20:02:36 2020
17	R09	Pass		Thu Jul 30 18:38:36 2020
16	R09	Pass		Thu Jul 30 18:13:20 2020
15	R09	SCRIPT FAIL	name 'fred' is not defined	Thu Jul 30 17:43:40 2020

```
# overall error record, generic
except Exception as err:
    strRecordPass = 'SCRIPT FAIL'
    strRowComments = err.args[0]
    strRecordTime = time.asctime(time.localtime(time.time()))
    arcpy.AddMessage('###############')
    arcpy.AddMessage('# CODE FAILED #')
    arcpy.AddMessage('###############')
    arcpy.AddMessage('D\oh - it failed!')
    arcpy.AddMessage('ERROR DETAILS:')
    arcpy.AddMessage(err.args[0])
    arcpy.AddMessage('--------------------------------')
    arcpy.AddMessage(PrintException())
    arcpy.AddMessage('--------------------------------')
    arcpy.AddMessage('###############')
    arcpy.AddMessage('@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@')
    arcpy.AddMessage('@          CAUTION             @')
    arcpy.AddMessage('@  CODE EXITING - UNFINISHED  @')
    arcpy.AddMessage('@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@')

finally:
    # populate the results Check table in the specific results FGBD
    # create insert cursor for table
    rows = arcpy.InsertCursor(strRecordTableName)
    # insert a single new row
    row = rows.newRow()
    # insert data
    row.setValue(strFieldName01, strRecordCheckID)
    row.setValue(strFieldName02, strRecordPass)
    row.setValue(strFieldName03, strRowComments)
    row.setValue(strFieldName04, strRecordTime)
    # save the row data
    rows.insertRow(row)
    # clean up some, though strictly unnecessary as the code is ending  ~smile~
    del row
    del rows
```

Unique Step Error Table Generated ‘As Needed’

- IF the validation script determines a data validation FAIL:
  - Record added to a new / overwritten table in the <Step_.gdb>
  - Details of part that failed (hydrologic logic) any values, and lengthier description

tblR07Errors_temp_20200824

Field: Add Calculate | Selection: Select By Attributes Zoom To Switch Clear Delete Copy

OBJECTID *	R07_Error_Part	R07_UpstreamOut_ToNode	R07_UpstreamOut_FromNode	R07_UpstreamOut_PathLength	R07_Error_Description
1	Part 2 - TO Node / Upstream / Out Details	55001200122409	55001200000009	1414	The Upstream / Out PathLength (1414) does NOT equal the Downstream / In Path Length (1409) + LengthKM (4)
2	Part 2 - TO Node / Upstream / Out Details	55001200122409	55001200041319	1414	The Upstream / Out PathLength (1414) does NOT equal the Downstream / In Path Length (1409) + LengthKM (4)

Click to add new row.

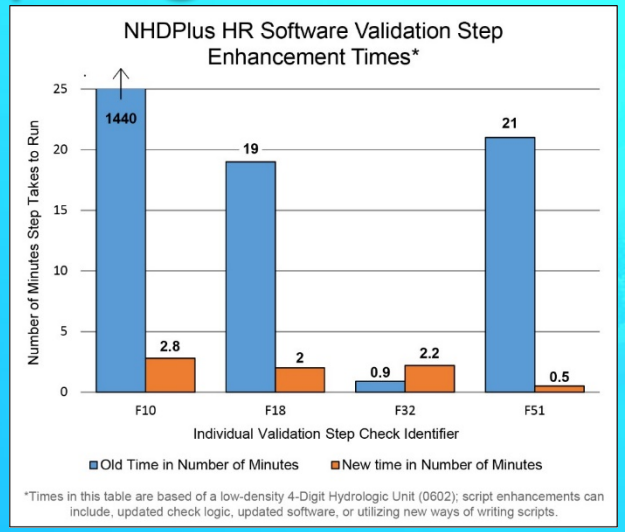
Share Your Scripting Effort's Metrics

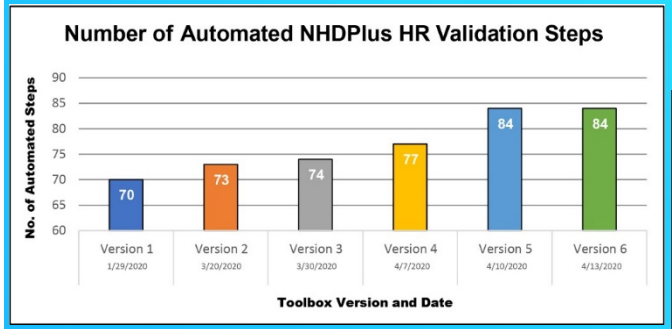
NHDPlusHR B/R Validation Scripting

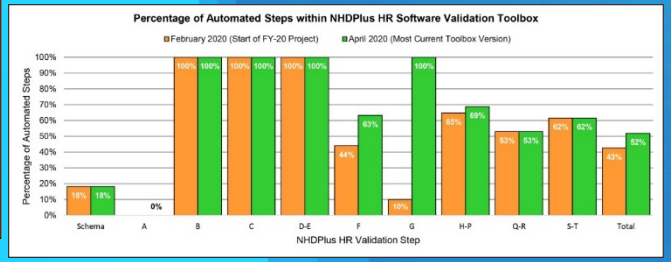
Examples of May 2019 Recent Script Speed Increases With Use Of pandas for manipulation of FC and table attribute combinations ***(independent research and autonomous self-learning)***

Script	VPU	ESRI Methods Only (2018)	NumPy Only (2018)	NumPy +pandas (2019)
F10	VPU 0508 (2018 Beta)	24 hr ++ (if will run at all, memory issues)	2 hr 40 min 22 sec	7 min 45 sec
F10	VPU 0602 (2019 Refresh)	24 hr ++ (if will run at all, memory issues)	1 hr 36 min 57 sec	2 min 46 sec
F10	VPU 1701 (dense) (2018 Beta)	40 hr ++ (if will run at all, memory issues)	34 hr 40 min 26 sec	28 min 46 sec
F18	VPU 0602 (2019 Refresh)	18 min 50 sec	-	1 min 20 sec
F18	VPU 0103 (2019 Refresh)	33 min 41 sec	-	2 min 26 sec
F18	VPU 0508 (2018 Beta)	3 hr 21 min	-	5 min 56 sec
F18	VPU 1701 (dense) (2018 Beta)	24 hr 16 min 11 sec	-	19 min 4 sec
F16	VPU 1701 (dense) (2018 Beta)	-	20 min 57 sec	2 min 16 sec

Script F17 also written with pandas methodologies, similar speeds





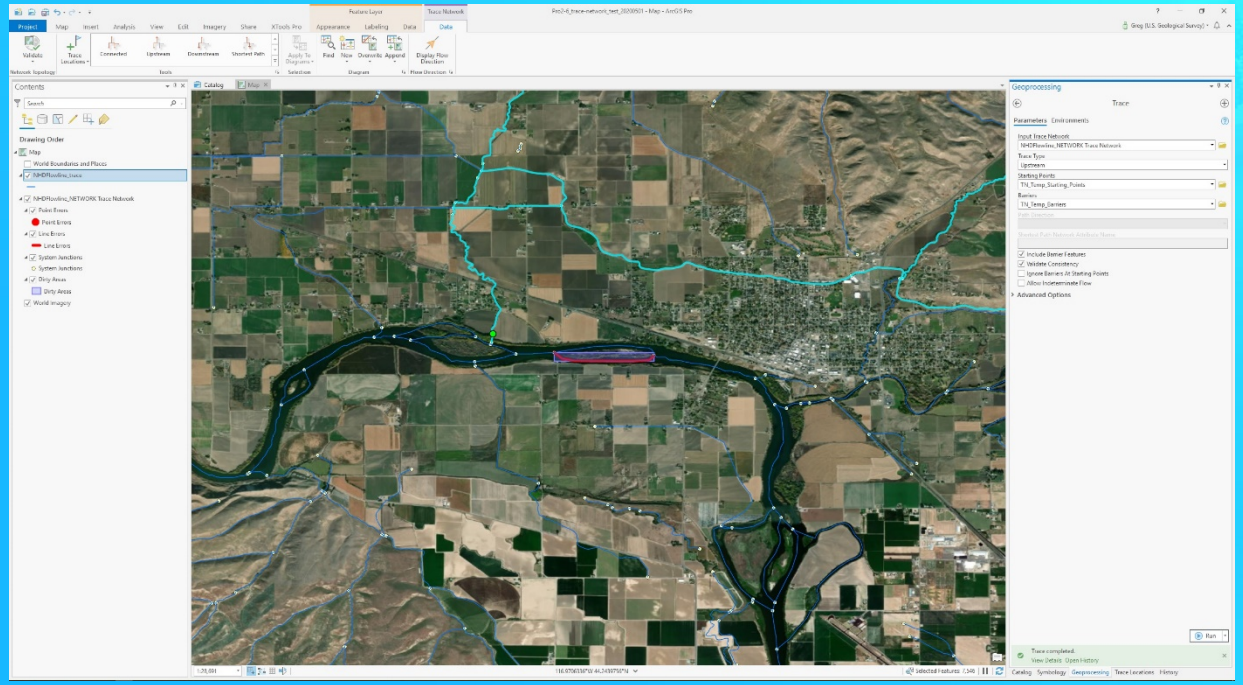


Ongoing Development

- Exploring faster handling of vector data

Ongoing Development (cont)

- Scripted use of Trace Network in ArcGIS Pro 2.x
  - Was part of a series of ESRI holistic testing teams

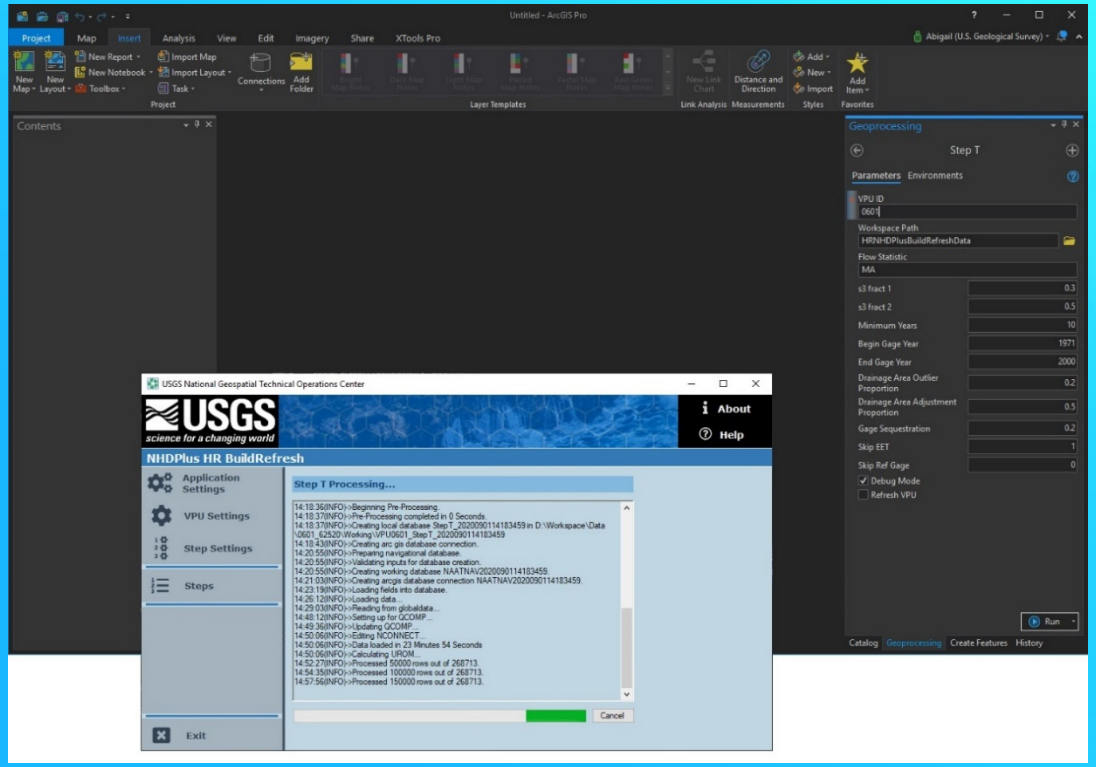


```
try:

    # get the projection of the flowline FC
    objSpatialRef = arcpy.Describe(strFCSourcePath).spatialReference

    # delete the FDS to be used, if it exists
    if arcpy.Exists(os.path.join(objInWorkspace, strFDSTrace)):
        arcpy.AddMessage('(0) DELETE EXISTING TRACE FDS...')
        arcpy.Delete_management(os.path.join(objInWorkspace, strFDSTrace))
        arcpy.AddMessage('--> Existing FDS <{}> deleted'.format(strFDSTrace))
        arcpy.AddMessage('-----------------------------------------------')

    # create a FDS
    arcpy.AddMessage('(1) CREATE TRACE FDS...')
    arcpy.CreateFeatureDataset_management(objInWorkspace, strFDSTrace, objSpatialRef)
    arcpy.AddMessage('--> New FDS <{}> created'.format(strFDSTrace))
    arcpy.AddMessage('-----------------------------------------------')

    # copy the flowline FC to the new FDS
    arcpy.AddMessage('(2) COPY FLOWLINE FC TO TRACE FDS...')
    arcpy.CopyFeatures_management(strFCSourcePath, strFCTracePath)
    arcpy.AddMessage('--> Flowline FC <{}> copied to FDS as <{}>'.format(strFCSource, strFCSourcePath))
    arcpy.AddMessage('-----------------------------------------------')

    # remove duplicate vertices in flowline FC, nominally anything less than ~ 10mm distance
    arcpy.AddMessage('(3) REMOVE DUPLICATE VERTICES IN FLOWLINE FC COPY IN THE TRACE FDS...')
    arcpy.AddMessage('-----------------')
    funremoveduplicatevertices(strFCTracePath)
    arcpy.AddMessage('--> Flowline FC copy <{}> had duplicate vertices - if any - removed across various flowlines'.format(strFCTrace))
    # arcpy.AddMessage('***** CURRENTLY TURNED OFF ****')
    arcpy.AddMessage('-----------------------------------------------')

    # create the network trace
    arcpy.AddMessage('(4) CREATE THE TRACE NETWORK FROM THE FLOWLINE FC COPY IN THE TRACE FDS...')
    strFCTraceDetails = '{} COMPLEX_EDGE'.format(strFCTrace)
    arcpy.tn.CreateTraceNetwork(os.path.join(objInWorkspace, strFDSTrace), strNetworkTrace, None, strFCTraceDetails)
    arcpy.AddMessage('--> Trace Network <{}> created in FDS'.format(strNetworkTrace, strFDSTrace))
    arcpy.AddMessage('-----------------------------------------------')

    # enable the network topology
    arcpy.AddMessage('(5) ENABLE THE NETWORK TOPOLOGY...')
    arcpy.tn.EnableNetworkTopology(os.path.join(objInWorkspace, strFDSTrace, strNetworkTrace), intMaxNumberTopologyErrors, "ENABLE_TOPO")
    arcpy.AddMessage('--> Network topology enabled for trace network <{}>'.format(strNetworkTrace))
    arcpy.AddMessage('-----------------------------------------------')
```

Data ‘tidying’ / adjustments (e.g., already written delete duplicate vertices (< 10 mm))

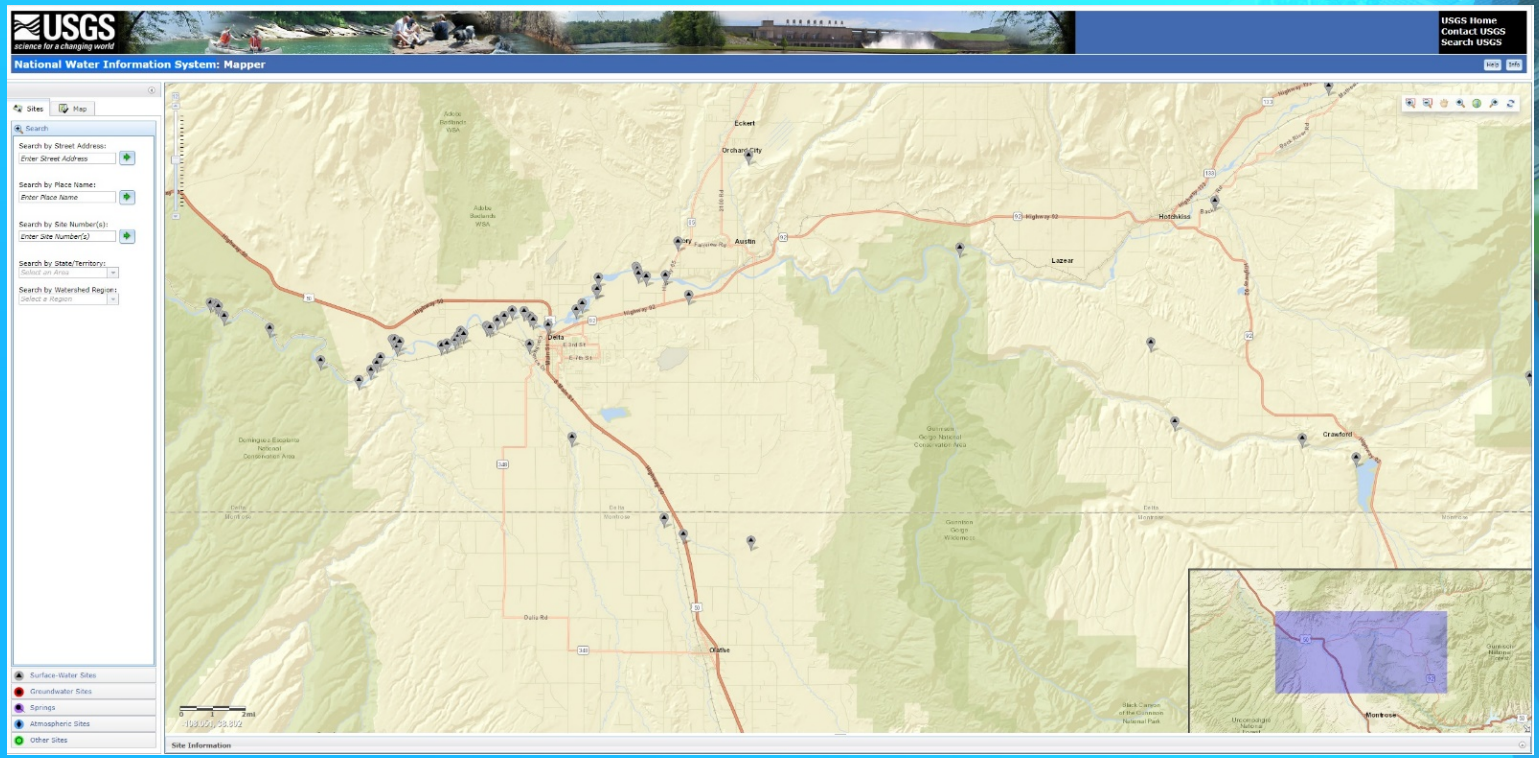
Ongoing Development (cont)

- NHDPlusHR data production is being brought ‘in house’, and the data validation scripts will be integrated into those processes



Ongoing Development (cont)

- (Possible) automated ‘scrapping’ of National Water Information System (NWIS) gauge & other for NHDPlusHR calibration, etc



Summary

- scripts are now being used in production validation of the generated NHDPlusHR datasets
- in contrast to the previous “1%" manual checking of some datasets with large time spent and potentially missed data validation issues
- being used for further production testing & scripting
- validation scripts will be an integrated part of the ‘in house’ NHDPlusHR data generation process, as mentioned

Summary (cont)

- Further - and as above - some of the techniques 'discovered' and then used in these tests (pandas, etc) are also being applied to the generation of the data - as the consultant-developed processes are brought ‘in house’ to the USGS, and modernised / standardised
- I continued to be involved in that process, researching, as well as directly supporting and encouraging the developers for various tasks (until I accepted a new role supporting NOAA)

Closing

The NHD is and will continue to be part of numerous scientific efforts, applied science, resource management, planning - and so much more…

It was a genuine honour to be a small part of its generation and validation.