THE UNIVERSITY OF TEXAS AT AUSTIN
CENTER FOR TRANSPORTATION RESEARCH

Product 0-7050-P1
TxDOT PROJECT NUMBER 0-7050

Automated Methodological Procedure for Identifying Curve-Related Crashes in CRIS

Yang Xu
Zhe Han
Michael Murphy
Zhanmin Zhang

May 2022
Published September 2022

https://library.ctr.utexas.edu/ctr-publications/0-7050-P1.pdf

THE UNIVERSITY OF TEXAS AT AUSTIN
CENTER FOR TRANSPORTATION RESEARCH

Automated Methodological Procedure for Identifying Curve-Related Crashes in CRIS (0-7050-P1)

Yang Xu
Zhe Han
Michael Murphy
Zhanmin Zhang

CTR Technical Report:	0-7050-P1
Report Date:	Submitted: May 2022
Project:	0-7050
Project Title:	Improving the Identification of Curve-Related Crashes in the Crash Records Information System (CRIS)
Sponsoring Agency:	Texas Department of Transportation
Performing Agency:	Center for Transportation Research at The University of Texas at Austin

Project performed in cooperation with the Texas Department of Transportation and the Federal Highway Administration.

Automated Methodological Procedure for Identifying Curve-Related Crashes in CRIS (0-7050-P1)

1. Introduction

The product developed by Project 0-7050 is an automated methodological procedure for identifying curve-related crashes in CRIS. This document provides detailed information on the implementation of the automated methodological procedure.

In order to automatically implement the methodological procedure for identifying curve-related crashes, the CTR research team first performed data cleaning and generated customized CRIS datasets using Python programming language via Jupyter Notebook. After the data preparation, the customized CRIS datasets were automatically imported into ArcGIS Pro for further visualization and analysis using Python libraries including ArcGIS API for Python and ArcPy. To evaluate the accuracy of the automated methodological procedure, a comparative study was performed using CRIS 2019 data. Based on the results, the CTR research team observed that the outputs of the automated procedure were identical to the results of the manual procedure performed in Task 5.

Figure 1 presents the framework of the automated methodological procedure, which is comprised of two phases. The first phase mainly focuses on data cleaning and preparation; the second phase is data processing and analysis.

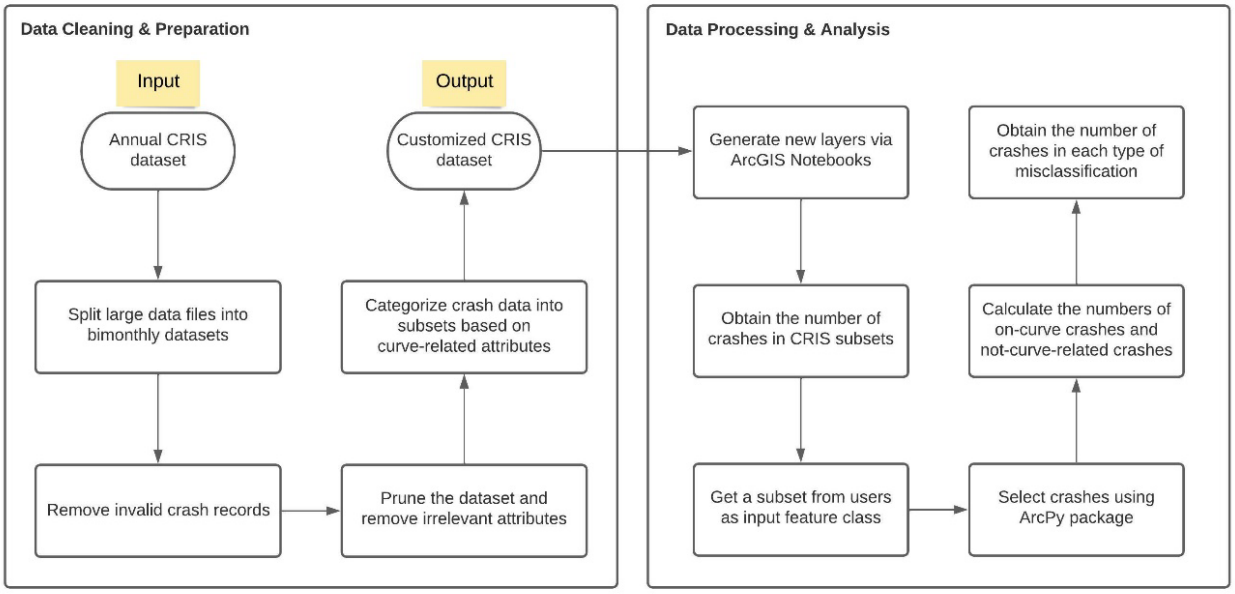

Figure 1. Implementation of the Automated Methodological Procedure for Identifying Curve-Related Crashes in CRIS

2. CRIS Data Cleaning and Preparation

To boost the overall efficiency of data cleaning and preparation, the first phase is completed via Jupyter Notebook, an open-source web application used to manage Python code. For the first phase, the input is an annual CRIS dataset, and the output is a customized CRIS dataset that is comprised of 24 bimonthly subsets, as shown in Figure 2.

Figure 2. Large Annual Crash Data File Split into 24 Bimonthly Subsets

Split large data files into a manageable size (i.e., bimonthly datasets)

The data files extracted from the CRIS Share platform are massive because these files contain all crashes that occurred in a specific year across the state. It is challenging and very time-consuming to directly process this magnitude of data. To improve data processing efficiency, the original annual crash data is partitioned into several smaller, bimonthly datasets, as shown in Figure 3, using the Python code illustrated in Figure 4. The six bimonthly datasets are then saved for further analysis.

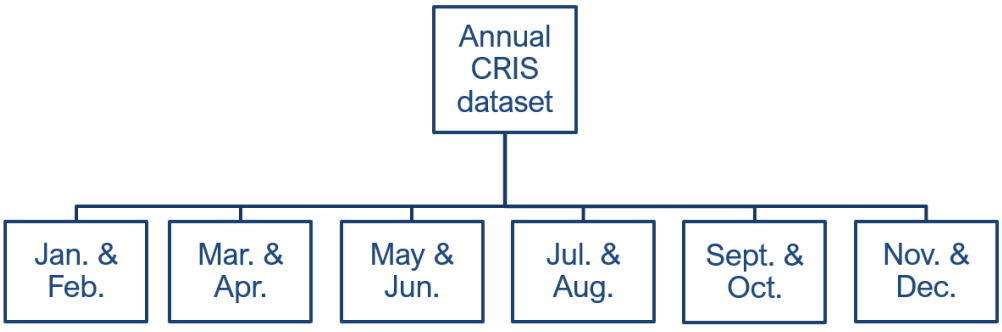

Figure 3. Large Annual Crash Data File Split into Six Bimonthly Datasets

```python
# Generate 6 bimonthly datasets
# 2020 CRIS, if the year is not 2020, please update it with the right year
crash_0101_0228or29 = cleanDF[(cleanDF['Date'] >= '01-01-2020') & (cleanDF['Date'] < '03-01-2020')]
crash_0301_0430 = cleanDF[(cleanDF['Date'] >= '03-01-2020') & (cleanDF['Date'] <= '04-30-2020')]
crash_0501_0630 = cleanDF[(cleanDF['Date'] >= '05-01-2020') & (cleanDF['Date'] <= '06-30-2020')]
crash_0701_0831 = cleanDF[(cleanDF['Date'] >= '07-01-2020') & (cleanDF['Date'] <= '08-31-2020')]
crash_0901_1031 = cleanDF[(cleanDF['Date'] >= '09-01-2020') & (cleanDF['Date'] <= '10-31-2020')]
crash_1101_1231 = cleanDF[(cleanDF['Date'] >= '11-01-2020') & (cleanDF['Date'] <= '12-31-2020')]

#Remove the field used to generat six bimonthly datasets
crash_0101_0228or29 = crash_0101_0228or29.drop(['Date'], axis=1)
crash_0301_0430 = crash_0301_0430.drop(['Date'], axis=1)
crash_0501_0630 = crash_0501_0630.drop(['Date'], axis=1)
crash_0701_0831 = crash_0701_0831.drop(['Date'], axis=1)
crash_0901_1031 = crash_0901_1031.drop(['Date'], axis=1)
crash_1101_1231 = crash_1101_1231.drop(['Date'], axis=1)

#Save six CRIS bimonthly datasets
# 2020 CRIS, if the year is not 2020, please update it with the right year in the file name
crash_0101_0228or29.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/crash_0101-0228_cleanDF.csv", index=False)
crash_0301_0430.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/crash_0301-0430_cleanDF.csv", index=False)
crash_0501_0630.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/crash_0501-0630_cleanDF.csv", index=False)
crash_0701_0831.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/crash_0701-0831_cleanDF.csv", index=False)
crash_0901_1031.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/crash_0901-1031_cleanDF.csv", index=False)
crash_1101_1231.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/crash_1101-1231_cleanDF.csv", index=False)
```

Figure 4. Python Codes for Splitting Large Annual Crash Data File into Bimonthly Datasets

Prune the dataset and retrieve attributes that are relevant to curve-related crash misclassifications

After carefully reviewing all data attributes in the available CRIS data, CTR identified 17 attributes that can assist in the identification of curve-related crash misclassification. These attributes include unique identifiers of crashes, locations in the format of different referencing methods, and horizontal curve information. Table 1 provides more details of these attributes.

Table 1. Attributes Relevant to Curve-Related Crash Misclassification

No.	Attribute Name	Column Name in CRIS	Description	Field Category
1	Crash ID	Crash_ID	System-generated unique identifying number for a crash	CR-3 Reported
2	Located Flag	Located_Fl	Indicates whether the CRIS locator application was able to locate the crash	System Generated
3	Latitude	Latitude	Latitude map coordinate of the crash	System Generated
4	Longitude	Longitude	Longitude map coordinate of the crash	System Generated
5	Street Name	Street_Name	Name of the road crash occurred on, as determined by the Locator application	System Generated
6	DFO	Dfo	The distance from the origin of the highway to the spot where the crash occurred	System Generated
7	On System Flag	Onsys_Fl	Indicates whether the primary road of the crash was on the TxDOT highway system	System Generated
8	Ref. Marker Nbr	Ref_Mark_Nbr	Reference marker number on the primary highway nearest the crash location	System Generated
9	Ref. Marker Displ.	Ref_Mark_Displ	The distance from the reference marker to the crash location	System Generated
10	Roadway Alignment	Road_Algn_ID	The geometric characteristics of the roadway at the crash site	CR-3 Reported
11	Curve Type ID	Curve_Type_ID	Type of curve, for crashes located on the state highway system	Appended Roadway Attributes
12	Length of Curve	Curve_Lngth	Length of curve, for crashes located on the state highway system	Appended Roadway Attributes
13	Curve degrees	Cd_Degr	Curve degrees (N & S type only), for crashes located on the state highway system	Appended Roadway Attributes
14	Curve delta degrees	Dd_Degr	Curve delta degrees (for crashes located on the state highway system)	Appended Roadway Attributes
15	Delta Left/Right ID	Delta_Left_Right_ID	Indicates whether the curve is right or left (for crashes located on the state highway system)	Appended Roadway Attributes
16	At Intersection	At_Intrsct_Fl	Indicates whether the crash occurred at an intersection	CR-3 Reported
17	IF- Intersection Related	Intrsct_Relat_ID	Specifies whether a crash occurred at an intersection, not at an intersection, or if the presence of an intersection contributed to the crash	Interpreted

The outcomes from the previous step (i.e., the bimonthly datasets) are then used to create cleaned datasets that only contain the attributes useful for identifying curve-related crashes. The codes, in Figure 5, show the process of loading six bimonthly datasets. In the code, we used the term "*subset1*" to "*subset6*" for programming purposes. For example, "*subset1*" represents the first bimonthly CRIS data in a year; "*subset6*" represents the last bimonthly data in a year.

```
import pandas as pd
import numpy as np
import multiprocessing
import warnings

import itertools
from pprint import pprint
import random
import os
import gc

warnings.simplefilter('ignore')
%matplotlib inline

#load six bimonthly CRIS subsets
#0101-0228
subset1 = pd.read_csv('C:/Project 0-7050/databases/CRIS/2020/CleanDB/crash_0101-0228_cleanDF.csv')

#0301-0430
subset2 = pd.read_csv('C:/Project 0-7050/databases/CRIS/2020/CleanDB/crash_0301-0430_cleanDF.csv')

#0501-0630
subset3 = pd.read_csv('C:/Project 0-7050/databases/CRIS/2020/CleanDB/crash_0501-0630_cleanDF.csv')

#0701-0831
subset4 = pd.read_csv('C:/Project 0-7050/databases/CRIS/2020/CleanDB/crash_0701-0831_cleanDF.csv')

#0901-1031
subset5 = pd.read_csv('C:/Project 0-7050/databases/CRIS/2020/CleanDB/crash_0901-1031_cleanDF.csv')

#1101-1231
subset6 = pd.read_csv('C:/Project 0-7050/databases/CRIS/2020/CleanDB/crash_1101-1231_cleanDF.csv')

```

Figure 5. Python Codes for Loading Six Bimonthly CRIS Datasets

After loading the dataset, users shall select one subset at a time (start from *subset1* and then the next subset in sequence) to perform subsequent data processing. This can be done by removing the hash mark "#" before the line. For example, in Figure 6, *subset6* is the selected subset. Users should always add the hash mark "#" back when they choose the next subset. Because only one bimonthly dataset can be processed at a time.

```
#iterate: select one subset at a time
#data = subset1
#data = subset2
#data = subset3
#data = subset4
#data = subset5
data = subset6
#data.head()
data.info()
```

Figure 6. Python Codes for Selecting One Bimonthly CRIS Datasets

Then, using the selected subset, users can clean the dataset by only keeping useful attributes that can provide meaningful information on the identification of curve-related crashes in CRIS. This can be achieved by running the code shown in Figure 7.

```
#Prune the dataset and retrieve attributes that
#are relevant to curve-related crash misclassifications
cleanDF = pd.DataFrame({'CrashID': data.Crash_ID})
#cleanDF

#indicates whether the CRIS locator application was able to locate the crash
cleanDF['Located_Fl']=data.Located_Fl

#locating crashes
cleanDF['Latitude']=data.Latitude
cleanDF['Longitude']=data.Longitude

cleanDF['Street_Name']=data.Street_Name
cleanDF['Dfo']=data.Dfo
cleanDF['Onsys_Fl']=data.Onsys_Fl

cleanDF['Ref_Mark_Nbr']=data.Ref_Mark_Nbr
cleanDF['Ref_Mark_Displ']=data.Ref_Mark_Displ

#curve-related info
cleanDF['Road_Algn_ID']=data.Road_Algn_ID
cleanDF['Curve_Type_ID']=data.Curve_Type_ID
cleanDF['Curve_Lngth']=data.Curve_Lngth
cleanDF['Cd_Degr']=data.Cd_Degr
cleanDF['Dd_Degr']=data.Dd_Degr
cleanDF['Delta_Left_Right_ID']=data.Delta_Left_Right_ID

#At_Intrsct_Fl
cleanDF['At_Intrsct_Fl']=data.At_Intrsct_Fl
#Intrsct_Relat_ID
cleanDF['Intrsct_Relat_ID']=data.Intrsct_Relat_ID

cleanDF
```

Figure 7. Python Codes for Pruning CRIS Dataset and Retrieving Useful Attributes

Remove invalid crash records (e.g., crashes missing location information or that are not on on-system roads)

The "*Located Flag*" field in CRIS is an indicator that reveals whether the location of a given crash can be identified by the system. When the "*Located Flag*" attribute equals "N," it means the crash cannot be located by the CRIS locator application. In this project, crash records with no location information are treated as invalid records and removed from the datasets, as these data are not able to provide any useful information for identifying curve-related crash misclassification. In addition to crashes without location information, crashes that are not on an on-system road are also removed from the datasets. These roadways are not designated on the state highway system and are not maintained by TxDOT; thus, the data are beyond the scope of this research.

The following Python codes presented in Figure 8 can be used to remove crashes that have no available location information or that did not occur on on-system highways.

```
#Remove invalid crash records (e.g., crashes missing location
#information or that are not on on-system roads)

#drop duplicated crash records
cleanDF = cleanDF.drop_duplicates(subset='CrashID', keep='first')
#cleanDF.info()

cleanDF=cleanDF.drop(cleanDF[cleanDF.Located_Fl == "N"].index)
cleanDF=cleanDF.drop(cleanDF[cleanDF.Onsys_Fl == "N"].index)

cleanDF = cleanDF[cleanDF['Latitude'].notna()]
cleanDF = cleanDF[cleanDF['Longitude'].notna()]

cleanDF
```

Figure 8. Python Codes for Removing Invalid Crash Records from the CRIS Datasets

Categorize crash data into subsets based on curve-related attributes

After the data is split into smaller datasets, each of the bimonthly datasets is disassembled into four subsets, as illustrated in Figure 9, based on the information provided by Curve Type ID and Road Align ID.

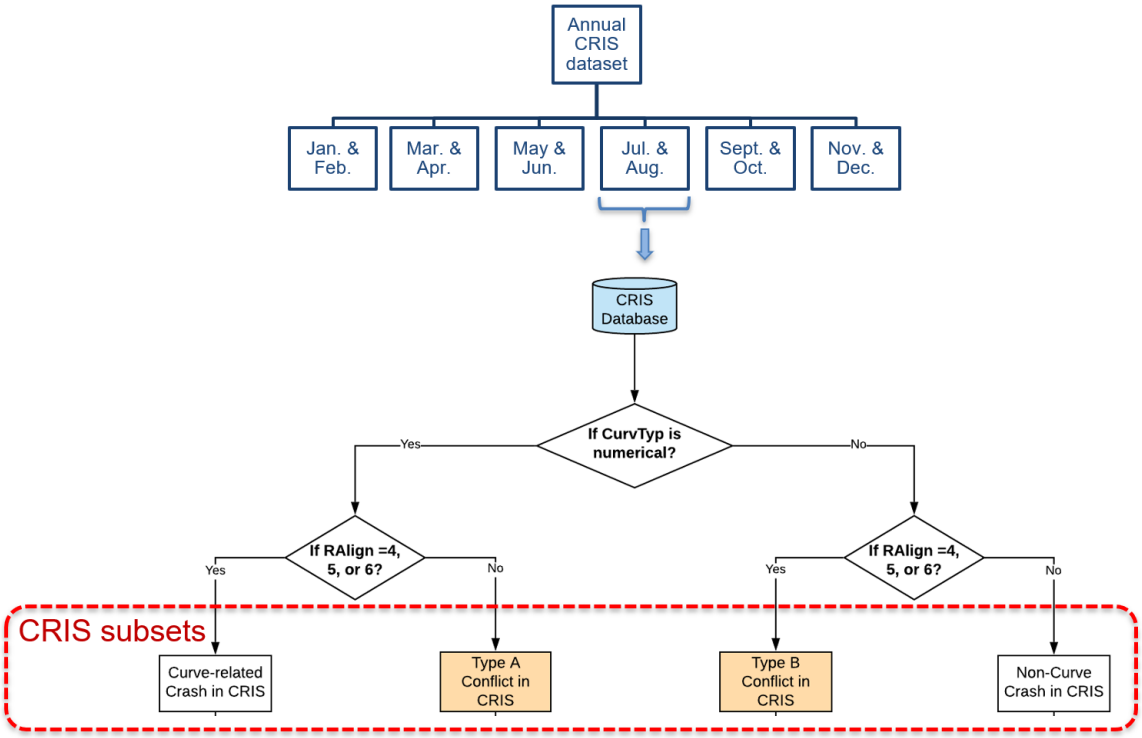

Figure 9. CRIS Data Subsets Based on Regrouped Curve Type and Road Alignment

The Python codes shown in Figure 10 are used to generate the four subsets based on the curve-related attributes in CRIS (i.e., Curve Type ID and Road Align ID).

```
curve_IDs = [4, 5, 6]
cleanDF_RAlign_456 = cleanDF[cleanDF['Road_Algn_ID'].isin(curve_IDs)]
cleanDF_RAlign_notCurv = cleanDF[cleanDF['Road_Algn_ID'].isin(curve_IDs)==False]

cleanDF_RAlignCurve_NUM = cleanDF_RAlign_456[cleanDF_RAlign_456['Curve_Type_ID'].notna()]
cleanDF_RAlignCurve_BLANK = cleanDF_RAlign_456[cleanDF_RAlign_456['Curve_Type_ID'].isnull()]
#cleanDF_RAlignCurve_NUM.info()

cleanDF_RAlignNotCurve_NUM = cleanDF_RAlign_notCurv[cleanDF_RAlign_notCurv['Curve_Type_ID'].notna()]
cleanDF_RAlignNotCurve_BLANK = cleanDF_RAlign_notCurv[cleanDF_RAlign_notCurv['Curve_Type_ID'].isnull()]
#cleanDF_RAlignNotCurve_BLANK.info()
```

Figure 10. Python Codes for Categorizing Crash Data into Subsets Based on Curve-Related Attributes

The final outputs from this phase are saved as plain text files in the format of comma-separated values (CSV) for further analysis. The codes for saving the subsets to CSV files are shown in Figure 11. When working on this step, users should select the correct time range based on the bimonthly dataset used in previous steps.

```
#save data
#iterate: save four subsets at a time

#0101-0228
#cleanDF_RAlignCurve_NUM.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0101-0228_RAlignCurve_NUM.csv",
#cleanDF_RAlignNotCurve_NUM.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0101-0228_RAlignNotCurve_NUM
#cleanDF_RAlignCurve_BLANK.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0101-0228_RAlignCurve_BLANK.c
#cleanDF_RAlignNotCurve_BLANK.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0101-0228_RAlignNotCurve_B

#0301-0430
#cleanDF_RAlignCurve_NUM.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0301-0430_RAlignCurve_NUM.csv",
#cleanDF_RAlignNotCurve_NUM.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0301-0430_RAlignNotCurve_NUM
#cleanDF_RAlignCurve_BLANK.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0301-0430_RAlignCurve_BLANK.c
#cleanDF_RAlignNotCurve_BLANK.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0301-0430_RAlignNotCurve_B

#0501-0630
#cleanDF_RAlignCurve_NUM.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0501-0630_RAlignCurve_NUM.csv",
#cleanDF_RAlignNotCurve_NUM.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0501-0630_RAlignNotCurve_NUM
#cleanDF_RAlignCurve_BLANK.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0501-0630_RAlignCurve_BLANK.c
#cleanDF_RAlignNotCurve_BLANK.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0501-0630_RAlignNotCurve_B

#0701-0831
#cleanDF_RAlignCurve_NUM.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0701-0831_RAlignCurve_NUM.csv",
#cleanDF_RAlignNotCurve_NUM.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0701-0831_RAlignNotCurve_NUM
#cleanDF_RAlignCurve_BLANK.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0701-0831_RAlignCurve_BLANK.c
#cleanDF_RAlignNotCurve_BLANK.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0701-0831_RAlignNotCurve_B

#0901-1031
#cleanDF_RAlignCurve_NUM.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0901-1031_RAlignCurve_NUM.csv",
#cleanDF_RAlignNotCurve_NUM.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0901-1031_RAlignNotCurve_NUM
#cleanDF_RAlignCurve_BLANK.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0901-1031_RAlignCurve_BLANK.c
#cleanDF_RAlignNotCurve_BLANK.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_0901-1031_RAlignNotCurve_B

#1101-1231
cleanDF_RAlignCurve_NUM.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_1101-1231_RAlignCurve_NUM.csv",
cleanDF_RAlignNotCurve_NUM.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_1101-1231_RAlignNotCurve_NUM.
cleanDF_RAlignCurve_BLANK.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_1101-1231_RAlignCurve_BLANK.cs
cleanDF_RAlignNotCurve_BLANK.to_csv("C:/Project 0-7050/databases/CRIS/2020/CleanDB/subsets/crash_1101-1231_RAlignNotCurve_BL

```

Figure 11. Python Codes for Saving CRIS Subsets as CSV Files

After performing all the aforementioned steps, each annual CRIS dataset will generate 24 bimonthly, categorized subsets that formed the customized CRIS dataset, as shown in Figure 2. The customized CRIS dataset is then used as input for the data analysis in ArcGIS Pro.

3. Loading Highway Curves GIS Layer to ArcGIS Pro

In this project, the CTR research team performed a comprehensive investigation of available data sources that contain reliable roadway geometry and inventory information maintained by TxDOT. The data sources include the Geometrics (Geo-HINI) database, the Geospatial Roadway Inventory Database (GRID), and the Highway Curves GIS layer. In addition to identifying the most reliable and accurate data source for roadway geometry and inventory, CTR also examined curve-related attributes that can provide information to assist with the identification of curve-related crashes in the CRIS database. Based on a thorough examination, the research team concluded that the Highway Curves GIS layer can serve as a reliable data source that encompasses both roadway geometric data and horizontal curve information. Therefore, the Highway Curves GIS layer is used as the integrated roadway geometry dataset for verifying curve-related information in CRIS.

Before starting the implementation of the methodological procedure for identifying curve-related crashes in CRIS. The first thing is to download the Highway Curves Geographic Information System (GIS) layer (available at http://arcg.is/1SPG8i) and open it in ArcGIS Pro, as shown in Figure 12 and Figure 13, respectively. The Highway Curves GIS layer is a feature layer developed by TxDOT, which provides a visualization of horizontal curves on highways across the state and contains curve-related information, as illustrated in Figure 14.

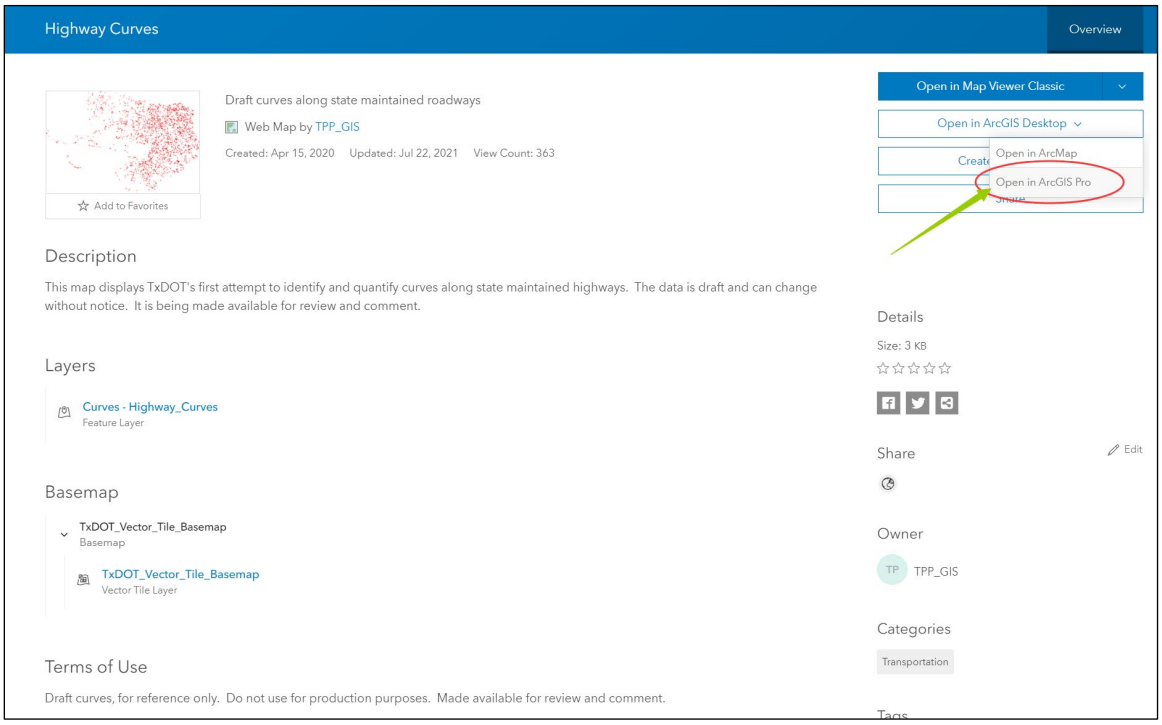

Figure 12. Download the Highway Curves GIS Layer

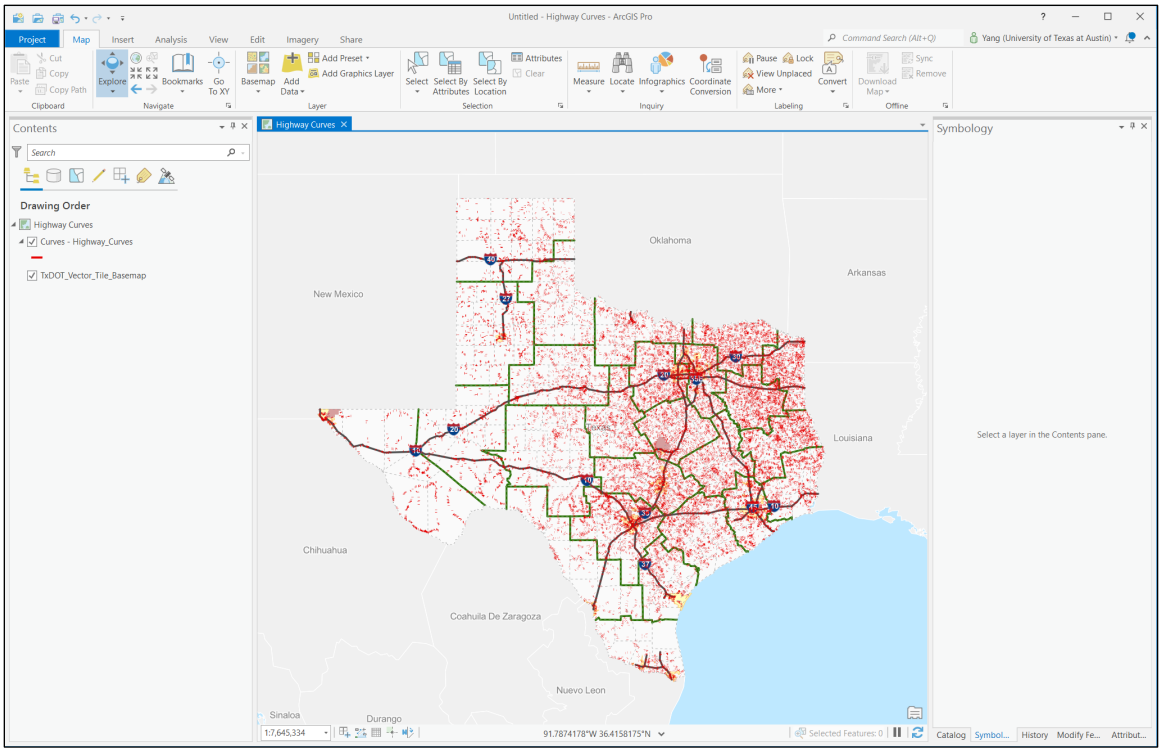

Figure 13. Open the Highway Curves GIS Layer Using ArcGIS Pro

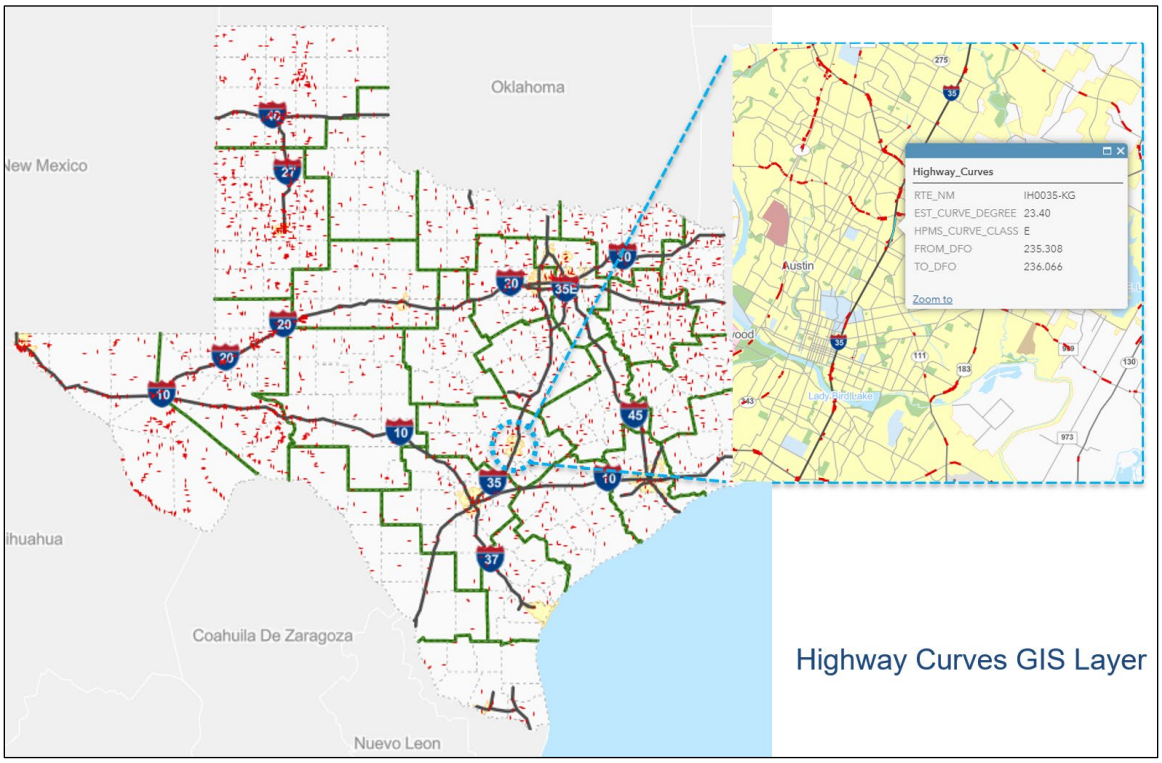

Figure 14. Screenshot of Texas Highway Curves GIS Layer

4. Data Integration, Processing, and Analysis

In the second phase of the methodological procedure, the CTR research team systematically performed data processing and analysis in ArcGIS Pro using the customized CRIS data as inputs. In particular, by leveraging Python programming language and ArcGIS Python libraries, the CTR research team accomplished the automation of two major tasks — visualization of the customized CRIS data in ArcGIS Pro and verification of curve-related crash classification using the Highway Curves GIS layer as a reference.

All the activities in the second phase are implemented in ArcGIS Notebooks, which are built-in Python notebooks that provide users with a convenient real-time environment to manage (e.g., create, edit, and save) their Python codes. By integrating ArcGIS Notebooks, ArcGIS Pro allows users to access GIS map content, conduct real-time data analysis, and obtain instant results that can be visualized in a geographic context (ESRI, 2022). ArcGIS Notebooks provide an efficient approach that can significantly relieve users of the burden of repetitive operations by automatically executing the workflow. In addition, ArcGIS Notebooks can be saved and shared within a project team, thus boosting the efficiency of collaborations and communication (Gimmler and Kalisky, 2020). Other usages of ArcGIS Notebooks include data cleaning, numerical simulation, statistical modeling, and machine learning, among others (ESRI, 2022).

Create an ArcGIS Notebook

In order to automate the methodological procedure in ArcGIS Pro, an ArcGIS Notebook is first created, as shown in Figure 15. To create a new notebook in ArcGIS Pro, click the "**Insert**" tab, and then click the "**New Notebook**" button as shown in Figure 15. For more information about how to use ArcGIS notebooks in ArcGIS Pro, please visit https://pro.arcgis.com/en/pro-app/2.8/arcpy/get-started/pro-notebooks.htm.

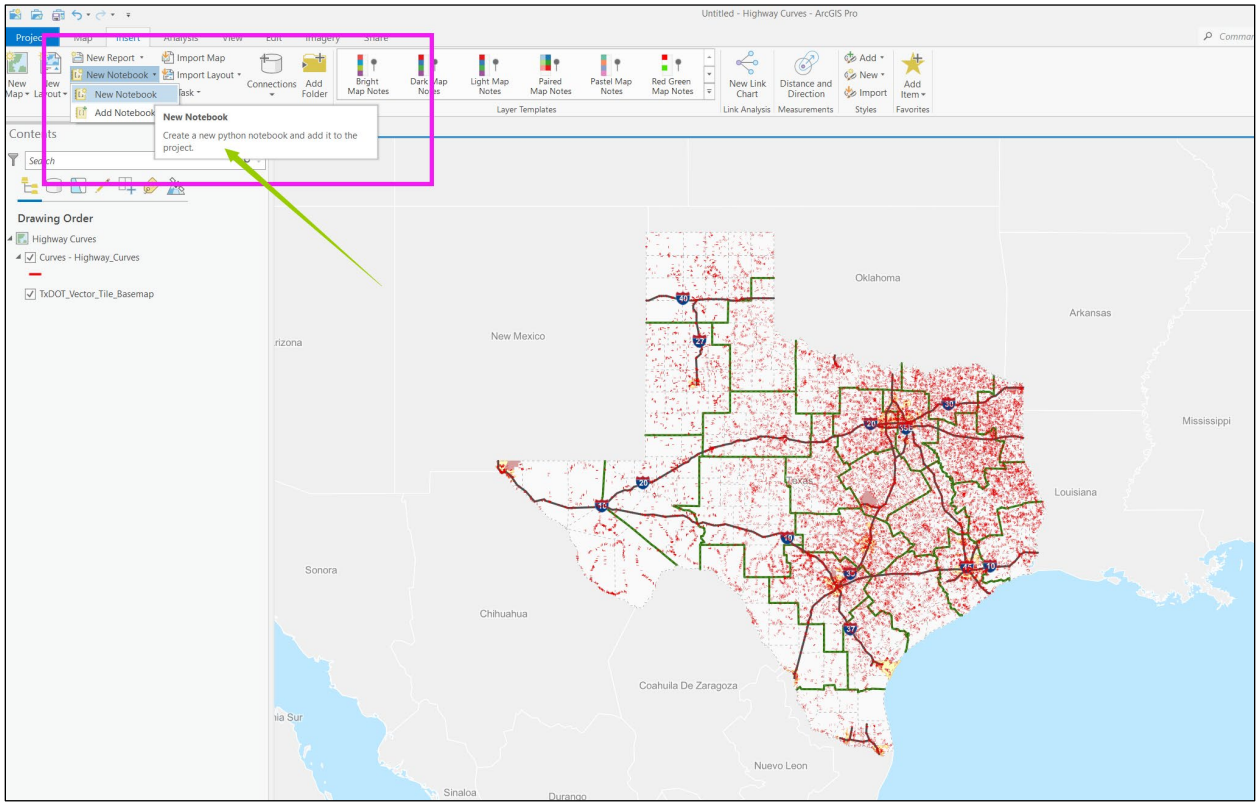

Figure 15. Create an ArcGIS Notebook

After clicking the "**New Notebook**" button under the "**Insert**" tab, the Notebook view window will pop up automatically, as shown in Figure 16.

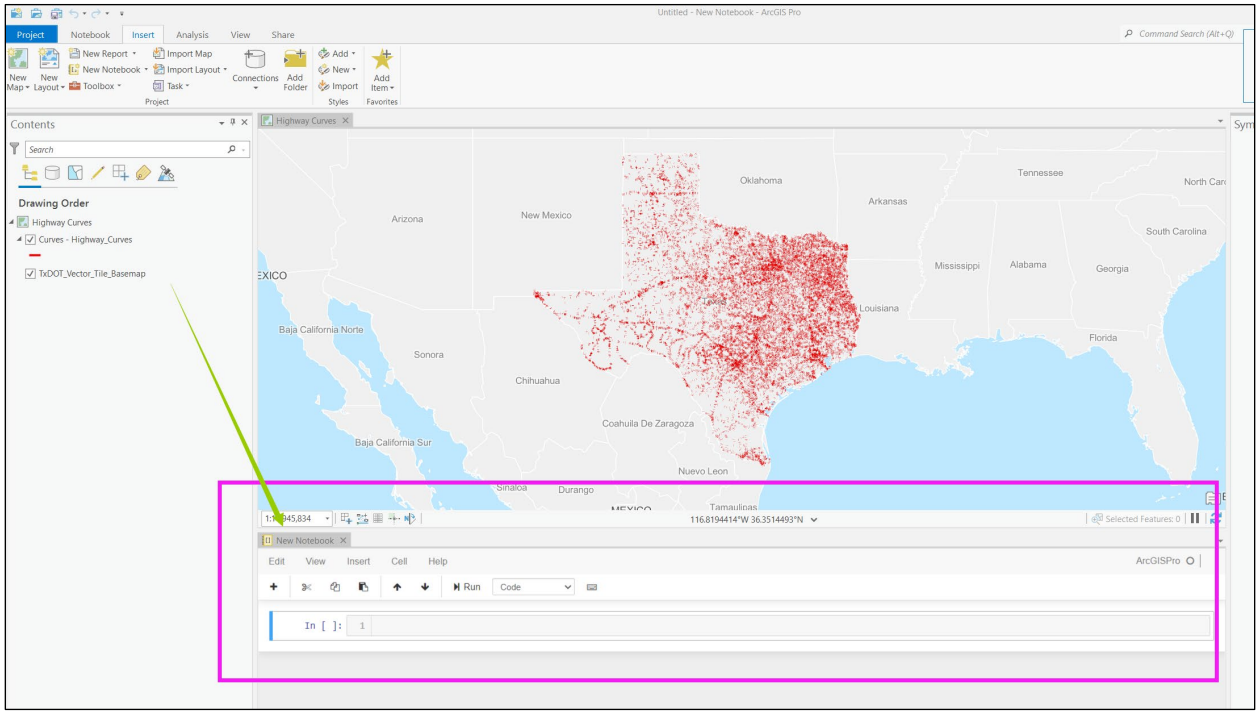

Figure 16. An Illustration of the Notebook View in ArcGIS Pro

After this step, users can start working on the second phase of the automated methodological procedure using the ArcGIS Notebook created through the previous steps. The codes are very compact and straightforward with describing notes. To run the codes, users can copy and paste them into the Notebook. The following sections provide step-by-step guidance on how to run the codes.

Step 1 – Generate new layers via ArcGIS Notebooks

The Python codes shown in Figure 17 are used to generate new layers using the customized bimonthly CRIS datasets generated from data preparation. In particular, the Python codes first load a bimonthly dataset (e.g., 11/01–12/31 from CRIS 2020) into ArcGIS Pro. For each bimonthly dataset, the imported data includes four subsets that are classified based on curve-related attributes (i.e., Curve Type ID and Road Align ID). For each of these subsets, the codes, as shown in Figure 17, will automatically create a new layer to visualize the crashes on the Highway Curves GIS map.

```
################################################################
#################### XYTableToPoint ############################
## Description: Creates a point feature class from input table ##
################################################################

# import system modules
import arcpy

# Set the local variables
in_table_1 = r"C:\Users\young.x\Desktop\Project 0-7050\databases\CRIS\2020\CleanDB\subsets\crash_1101-1231_RAlignCurve_NUM.csv"
in_table_2 = r"C:\Users\young.x\Desktop\Project 0-7050\databases\CRIS\2020\CleanDB\subsets\crash_1101-1231_RAlignNotCurve_NUM.csv"
in_table_3 = r"C:\Users\young.x\Desktop\Project 0-7050\databases\CRIS\2020\CleanDB\subsets\crash_1101-1231_RAlignCurve_BLANK.csv"
in_table_4 = r"C:\Users\young.x\Desktop\Project 0-7050\databases\CRIS\2020\CleanDB\subsets\crash_1101-1231_RAlignNotCurve_BLANK.csv"

# Define the names of the output feature layer
out_feature_class_1 = "CTnumRAcurv"
out_feature_class_2 = "CTnumRAnotCurv"
out_feature_class_3 = "CTblkRAcurv"
out_feature_class_4 = "CTblkRAnotCurv"

x_coords = "Longitude"
y_coords = "Latitude"
#GCS_North_American_1983 = 4269
SR = arcpy.SpatialReference(4269)

# Make the XY event layer...
# from 4 to 1, no need to rearrange the order of the layers
arcpy.management.XYTableToPoint(in_table_4, out_feature_class_4,
                                x_coords, y_coords, "", SR)
arcpy.management.XYTableToPoint(in_table_3, out_feature_class_3,
                                x_coords, y_coords, "", SR)
arcpy.management.XYTableToPoint(in_table_2, out_feature_class_2,
                                x_coords, y_coords, "", SR)
arcpy.management.XYTableToPoint(in_table_1, out_feature_class_1,
                                x_coords, y_coords, "", SR)
```

Figure 17. Python Codes for Generating New Layers via ArcGIS Notebooks

Step 2 – Compute the number of crashes in CRIS subsets

Then, the number of crashes that belonged to each of the four CRIS subsets are computed using the codes illustrated in Figure 18.

```
# Compute the number of crashes in each of the four CRIS subsets

result1 = arcpy.GetCount_management(out_feature_class_1)
result2 = arcpy.GetCount_management(out_feature_class_2)
result3 = arcpy.GetCount_management(out_feature_class_3)
result4 = arcpy.GetCount_management(out_feature_class_4)

totCrashCount1 = int(result1.getOutput(0))  #CurveTpye=Num, RoadAlign=Curve
totCrashCount2 = int(result2.getOutput(0))  #CurveTpye=Num, RoadAlign=NOTCurve
totCrashCount3 = int(result3.getOutput(0))  #CurveTpye=Blk, RoadAlign=Curve
totCrashCount4 = int(result4.getOutput(0))  #CurveTpye=Blk, RoadAlign=NOTCurve

print("(CTnumRAcurv, CTnumRAnotCurv, CTblkRAcurv, CTblkRAnotCurv) = ")
totCrashCount1, totCrashCount2, totCrashCount3, totCrashCount4
```

Figure 18. Python Codes for Computing the Number of Crashes in CRIS Subsets

Step 3 – Obtain one CRIS subset as the input feature class

In this step, users are expected to select one subset from the four CRIS subsets by entering the name of the subset into the field highlighted in Figure 19. This selected subset will be used as the input feature class for the data analysis performed in the next step.

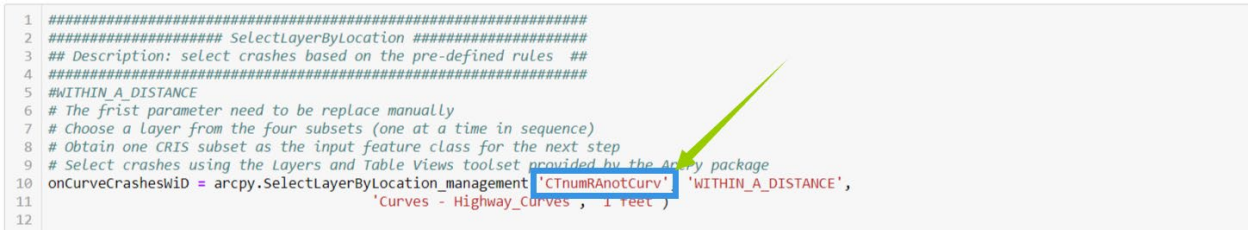

Figure 19. Python Codes for Selecting One CRIS Subset as Input Feature Class

Step 4 – Select crashes using the ArcPy package

As shown in Figure 20, the Python codes employ the *Layers and Table Views* toolset provided by the ArcPy package to select crashes based on the rules entered by users. Parameters defined in this step include input features, selecting features, relationship between selected features, search distance (set at 1 foot in this task), selection type, etc. Users only need to input the first parameter as described in Step 3.

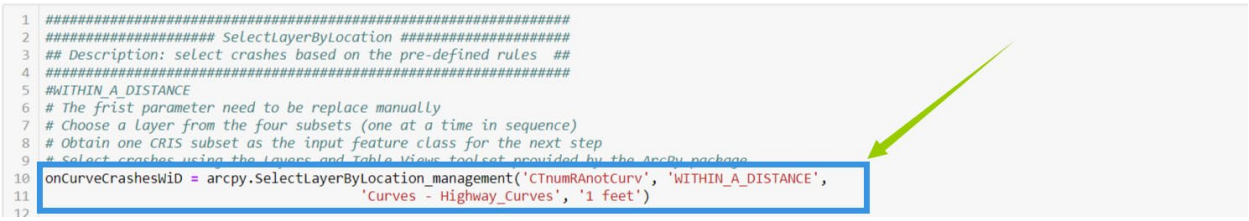

Figure 20. Python Codes for Selecting Curve-Related Crashes

Step 5 – Calculate the number of on-curve crashes

In this step, the following Python codes presented in Figure 21 are used to calculate the number of on-curve crashes.

```
# Calculate the number of on-curve crashes
onCurveCount = int(arcpy.GetCount_management(onCurveCrashesWiD)[0])
```

Figure 21. Python Codes for Calculating the Number of On-Curve Crashes

Step 6 – Compute the number of not-curve-related crashes

Similarly, the codes shown in Figure 22 can be used to obtain the number of crashes that are not curve-related.

```
# Compute the number of crashes that are not curve-related
notCurveCount = totCrashCount2 - onCurveCount                    #!!! use proper totCrashCountX

#print results
print("(# of on-curve crashes, # of NOT curve crashes) = ")
onCurveCount, notCurveCount
```

Figure 22. Python Codes for Computing the Number of Not-Curve-Related Crashes

Step 7 – Repeat Step 3 to Step 6 for all four CRIS subsets

For each bimonthly dataset, the imported data includes four CRIS subsets. By running the Python codes presented in Steps 3 through 6, users can obtain the number of curve-related and not-curve-

related crashes belonging to one of the four subsets. Hence, to obtain all data for a bimonthly CRIS dataset, users will need to repeat the operations for each subset.

Step 8 – Obtain the number of crashes in each type of classification

After the first seven steps are completed, the number of each type of six misclassifications and two correct classifications can be obtained, as illustrated in Figure 23.

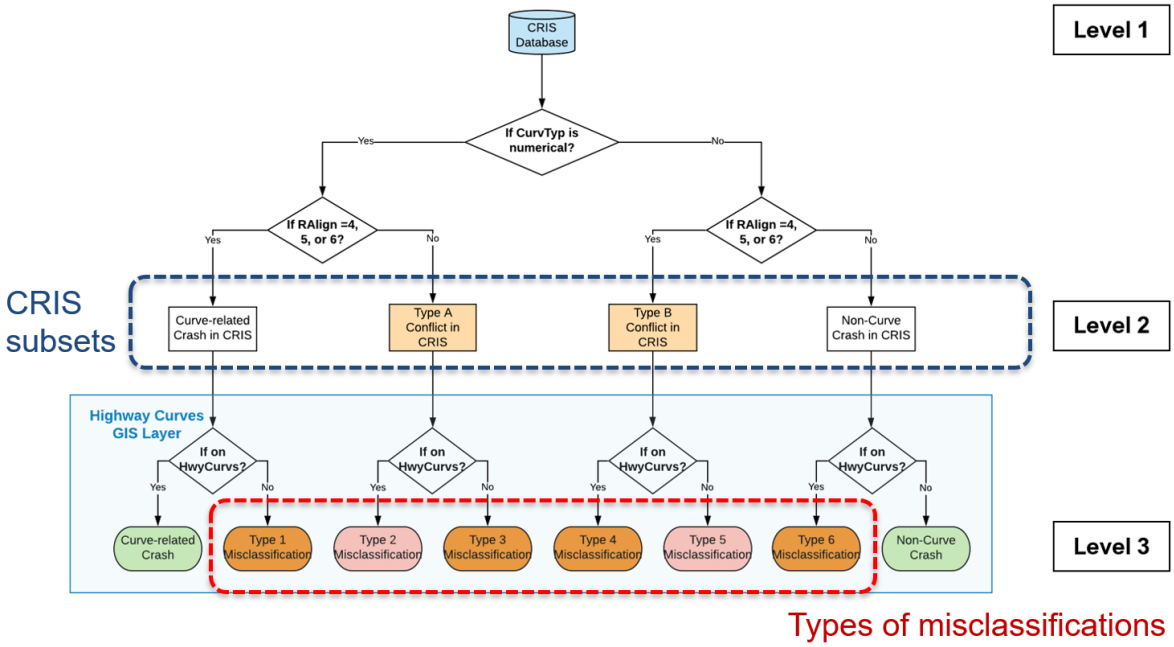

Figure 23. Tree Chart for Six Types of Curve-Related Crash Misclassifications in CRIS

References

Environmental Systems Research Institute (ESRI). 2022. Notebooks in ArcGIS Pro. Accessed on April 26, 2022. Available at: https://pro.arcgis.com/en/pro-app/2.7/arcpy/get-started/pro-notebooks.htm

Gimmler, L. and S. Kalisky. 2020. Introducing ArcGIS Notebooks in ArcGIS Pro. February 04, 2020. Accessed on April 26, 2022. Available at: https://www.esri.com/arcgis-blog/products/arcgis-pro/analytics/introducing-arcgis-notebooks-in-arcgis-pro/