# IDAHO TRANSPORTATION DEPARTMENT

# RESEARCH REPORT

## Off-System Public Roads Annual Average Daily Traffic (AADT) Estimation and Validation Tools

RP 314

By

Bryce Miller, Mark Egge

High Street

Prepared for

Idaho Transportation Department

ITD Research Program, Planning and Development Services

Highways Division

April 2025

# Disclaimer

This document is disseminated under the sponsorship of the Idaho Transportation Department and the United States Department of Transportation in the interest of information exchange. The State of Idaho and the United States Government assume no liability of its contents or use thereof.

The contents of this report reflect the view of the authors, who are responsible for the facts and accuracy of the data presented herein. The contents do not necessarily reflect the official policies of the Idaho Transportation Department or the United States Department of Transportation.

The State of Idaho and the United States Government do not endorse products or manufacturers. Trademarks or manufacturers' names appear herein only because they are considered essential to the object of this document.

***This report does not constitute a standard, specification or regulation.***

# Technical Report Documentation Page

| | | |
|---|---|---|
| 1. Report No.<br>FHWA-ID-24-314 | 2. Government Accession No. | 3. Recipient's Catalog No. |
| 4. Title and Subtitle<br>Off-System Public Roads Annual Average Daily Traffic (AADT) Estimation and Validation Tools: Literature Review and Research Report | | 5. Report Date<br>July 24, 2024<br>6. Performing Organization Code |
| 7. Author(s)<br>Mark Egge<br>Bryce C. Miller | | 8. Performing Organization Report No. |
| 9. Performing Organization Name and Address<br>High Street Consulting Group<br>6937 Blenheim Ct<br>Pittsburgh, PA 15208 | | 10. Work Unit No. (TRAIS)<br>11. Contract or Grant No.<br>T003165 |
| 12. Sponsoring Agency Name and Address<br>Idaho Transportation Department (SPR)<br>Highways Division, Planning and Development Services, Research Program<br>PO Box 7129<br>Boise, ID 83707-7129 | | 13. Type of Report and Period Covered<br>Final Report<br>[05/01/2024 – 05/31/2025]<br>14. Sponsoring Agency Code<br>RP-314 |
| 15. Supplementary Notes<br>This project is performed in cooperation with the Idaho Transportation Department and Federal Highway Administration. | | |
| 16. Abstract<br>This report describes a process for estimating AADT on non-Federal-Aid public roads (collectively called "off-system" roads) in Idaho. This will provide additional data for ITD and regional and local transportation agencies to conduct analysis, and it will meet new federal requirements for estimating AADT on all public roads. The report includes a review of federal guidance, documentation of ITD's current AADT estimation process, an overview of spatial interpolation via Kriging, a user guide for the Rural AADT Estimation toolbox in Esri's ArcGIS Pro software environment, and end materials including works cited, a list of independent variables considered, and the Python code used to run the toolbox. | | |
| 17. Key Words<br>Annual average daily traffic, traffic volume, traffic estimation, non-Federal-Aid highways local roads, interpolation, geospatial data | 18. Distribution Statement<br>Copies available from the ITD Research Program | |

| | | | |
|---|---|---|---|
| 19. Security Classification (of this report)<br>Unclassified | 20. Security Classification (of this page)<br>Unclassified | 21. No. of Pages<br>183 | 22. Price<br>None |

# Acknowledgments

This project has benefited from the feedback, experience, assistance, and insights of many organizations and people. These include but are not limited to—

- The Technical Advisory Committee (TAC) members, including those from the Idaho Transportation Department, the Local Highway Technical Assistance Council (LHTAC), and the Federal Highway Administration (FHWA).

# Technical Advisory Committee

Each research project is overseen by a Technical Advisory Committee (TAC), which is led by an ITD project sponsor and project manager. The TAC is responsible for monitoring project progress, reviewing deliverables, ensuring that study objectives are met, and facilitating implementation of research recommendations, as appropriate. ITD's Research Program Manager appreciates the work of the following TAC members in guiding this research study.

- Project Sponsor: Idaho Transportation Department
- Project Manager: Vicky Calderon
- TAC Members:
    - Idaho Transportation Department (ITD): Vicky Calderon, David Coladner, Taylor Emmons (former), Nicole Hanson, Amanda Laib, Margaret Pridmore
    - Federal Highway Administration (FHWA): Lance Johnson
- FHWA-Idaho Advisor: Lance Johnson

# Table of Contents

# List of Tables

# List of Figures

# Acronyms

| | |
|---|---|
| AADT | Annual average daily traffic |
| AADTT | Annual average daily truck traffic |
| AASHTO | American Association of State Highway and Transportation Officials |
| AC | Asphalt-concrete |
| ACS | American Community Survey |
| ATR | Automatic traffic recorder |
| CAADT | Commercial AADT |
| CFR | Code of Federal Regulations |
| COMPASS | Community Planning Association of Southwest Idaho |
| CRCP | Continuously reinforced concrete pavement |
| CSV | Comma separated values |
| DMV | Department of Motor Vehicles |
| DOT | Department of transportation |
| FC | Functional classification |
| FHWA | Federal Highway Administration |
| FIPS | Federal Information Processing Standards |
| GIS | Geographic information systems |
| GPS | Global positioning system |
| GWR | Geographically weighted regression |
| HPMS | Highway Performance Monitoring System |
| HQ | Headquarters |
| HSIP | Highway Safety Improvement Program |
| IAHD | Idaho Association of Highway Districts |
| ITD | Idaho Transportation Department |
| ITE | Institute of Transportation Engineers |
| JPCP | Jointed plain concrete pavement |
| JRCP | Jointed reinforced concrete pavement |
| IAHD | Idaho Association of Highway Districts |
| LCVMPO | Lewis-Clark Valley Metropolitan Planning Organization |
| LEHD | Longitudinal Employer Household Dynamics |
| LHTAC | Local Highway Technical Assistance Council |
| LODES | LEHD Origin-Destination Employment Statistics |
| LRI | Local road inventory |
| LRS | Linear referencing system |
| ME | Mean error |
| MAPE | Mean absolute percent error |
| MIRE | Model Inventory of Roadway Elements |
| ML | Machine learning |
| MPO | Metropolitan planning organization |
| MSE | Mean squared error |
| NAICS | North American Industry Classification System |
| NCHRP | National Cooperative Highway Research Program |
| NHS | National Highway System |
| NLCD | National Land Cover Database |
| NPMRDS | National Performance Management Research Data Set |

| | |
|---|---|
| PCC | Portland cement concrete |
| PMID | Pavement management ID |
| RMSE | Root-mean-square error |
| RPC | Regional Planning Council |
| SVM | Support vector machines |
| TAC | Technical advisory committee |
| TDM | Travel demand model |
| TMC | Traffic message channel |
| VMT | Vehicle miles traveled |
| WAC | Workplace area characteristics |
| WIM | Weigh-in-motion |

# Definitions

**Annual average daily traffic (AADT):** The AADT measure is an estimate of the average daily volume of vehicle traffic on a section of roadway over a full year. Discussion of how ITD currently calculates AADT and examples of equations from federal guidance for calculating AADT based on different types and durations of traffic counts.

**Federal-Aid roads:** Federal-Aid roads are those belonging to the National Highway System, the Eisenhower National System of Interstate and Defense Highways ("Interstate Highways"), and all other public roads not classified as local roads (functional classification 7) or rural minor collectors (functional classification 6) (23 CFR § 470.103).

**Local roads:** As defined by the Idaho Transportation Department Systems Procedures, "local roads provide direct access to residential neighborhoods, local businesses, agricultural properties and timberlands. Volumes typically range from less than a hundred to possibly thousands of vehicles per day. Roads not classified as arterials or collectors are considered local roads" (Idaho Transportation Department 2016).

**Off-system roads:** Off-system roads are public roads that are not a part of the Federal-Aid system. In Idaho, these are generally local roads (functional classification 7) and rural minor collectors (functional classification 6).

**On-system roads:** On-system roads are public roads that are part of the Federal-Aid System.

**Public roads:** Public roads are those "under the jurisdiction of and maintained by a public authority and open to public travel." A public authority is a "federal, state, county, town, or township, Indian tribe, municipal, or other local government or instrumentality with authority to finance, build, operate, or maintain toll or toll-free facilities" (23 USC § 101 (22-23)).

**Urban roads:** Urban roads are in an area where the population exceeds 5,000 (INSIDE Idaho 2020).

# 1. Introduction

This report describes a method to estimate traffic on Idaho's off-system roads where the Idaho Transportation Department's (ITD) does not routinely perform traffic counts. It augments ITD's existing process for estimating annual average daily traffic (AADT) on all Federal-Aid roads in Idaho with estimates of AADT for remaining non-Federal-Aid public roads (collectively referred to as "off-system" roads in this report) created through a combination of spatial interpolation and linear regression. The expanded AADT estimates will facilitate crash analytics by ITD's Traffic Safety group and respond to new federal recommendations for data collection described in the Model Inventory of Roadway Elements (MIRE). Additionally, the Highway Safety Improvement Program (HSIP) will require every paved road to have estimates of AADT made available by 2026 (Tsapakis, Holik, et al. 2020). This methodology and process aims to support ITD's goals and processes (including that it produces the most accurate estimates possible, be executable with in-house resources, and be relatively straightforward to explain and understand), and was developed with ITD's guidance and the technical advisory committee's (TAC) input.

This report has three primary chapters: Literature Review, Estimation Methodology, and User Guide. "Literature Review" reviews existing research for estimating AADT on public off-system or similar roads, describes ITD's existing AADT estimation methodology, and reviews FHWA requirements and recommendations around estimating AADT. "Estimation Methodology" goes into detail on the overall process for gathering, analyzing, and evaluating information to create useful traffic volume estimates. The "User Training Guide" walks through the Esri ArcGIS Pro Toolbox step-by-step in creating and assessing AADT estimates in Idaho, acting as a direct training and reference guide for ITD staff and as a jumping-off point for other organizations. The "Project User Guide" provides details on the operation of the underlying Python script for each tool within the Off-System AADT Estimation toolbox. The Works Cited and Appendices provide detailed references to the resources that form the basis for using geospatial interpolation to better estimate traffic volumes, the set of independent variables considered throughout the project to evaluate their ability to support better AADT estimation, and the underlying Python code used within the tools to create estimates in Esri's ArcGIS Pro software environment.

# 2. Literature Review

This section reviews traffic counting federal requirements and guidance, current ITD practices, and literature from state departments of transportation (DOTs) and other research organizations that are studying spatial AADT estimation to provide the context within which this project's toolbox will operate. This review indicates a broad leeway for ITD to pursue AADT estimation techniques that make sense for its needs since there are few federal requirements around how to estimate AADT on off-system public roads. There is no industry-wide consensus around best practices.

This section addresses three topics. The first covers federal regulations and rules contained in the *Traffic Monitoring Guide* related to AADT estimation. The second is a summary of methods for estimating AADT on off-system public roads, including describing pros and cons of each method and locations where the method has been implemented; academic research, a survey of states, and instances of practical application inform this methodological review. Finally, the third is ITD's current process for estimating segment-level AADT and statewide vehicle miles traveled (VMT).

## Federal Regulations & Rules

The *Traffic Monitoring Guide* (TMG) published by the Federal Highway Administration (FHWA) provides guidance to state highway agencies about the policies, standards, procedures, and equipment for traffic monitoring programs. The *Traffic Monitoring Guide* primarily consists of recommendations rather than requirements about how to conduct traffic monitoring, and the recommendations cover statewide traffic monitoring programs rather than being specifically tailored to AADT estimation off of the Federal-Aid System. However, it remains the most authoritative source for standards about collecting traffic monitoring. This section is based on the 2016 *Traffic Monitoring Guide* (Federal Highway Administration 2016) and is supplemented by the newest version of the *Traffic Monitoring Guide* released in December 2022 (Federal Highway Administration 2022). FHWA's *Traffic Data Computation Method* Pocket Guide provides additional guidance on calculating AADT (Federal Highway Administration 2018).

This subsection addresses the recommendations that the *Traffic Monitoring Guide* provides related to AADT estimation, including traffic count programs, calculations to derive AADT from traffic counts, and related data submission requirements. It documents best practices related to traffic count programs, but few requirements and no best practices specifically geared toward AADT estimation for off-system roads.

The 2016 *Traffic Monitoring Guide* contains several relevant chapters to AADT estimation. Recommendations around the traffic count program are in Chapter 2 ("Traffic Monitoring Program") and Chapter 3 ("Traffic Monitoring Methodologies"). Recommendations related to calculations for deriving AADT are also in Chapter 3 ("Traffic Monitoring Methodologies"). Recommendations related to data submission are in Chapter 6 ("HPMS Requirements for Traffic Data") and in Chapter 7 ("Traffic Monitoring Formats"). Other chapters address theory (Chapter 1), the traffic monitoring program

(Chapter 2), traffic monitoring for non-motorized traffic (Chapter 4), and transportation management and operations (Chapter 5). The 2022 *Traffic Monitoring Guide* discusses the traffic count program in Chapter 1, 2, and 3 ("Traffic Monitoring Program Introduction," "Traffic Data Collection Technology and Equipment," and "Methodologies for Traffic Data Collection and Processing" respectively). Recommendations related to calculations for deriving AADT are in Chapter 3 ("Methodologies for Traffic Data Collection and Processing"). Other chapters address data formats (Chapter 4), data reporting requirements (Chapter 5), and third-party traffic data (Chapter 6).

## Traffic Count Programs

FHWA recommends for state DOTs to evaluate their traffic monitoring programs at least once every five years, and that the evaluation should cover the entire program including equipment, selection of data collection sites, validation, and data analysis (2016 *Traffic Monitoring Guide* Section 2.3, 2022 *Traffic Monitoring Guide* Section 1.6). Most traffic monitoring programs will include continuous data programs for sites for which data is continuously collected and short-duration count sites. The *Traffic Monitoring Guide* provides methods for collecting and processing traffic data including data on traffic volume (source: 2016 *Traffic Monitoring Guide* Figure 3-2, 2022 *Traffic Monitoring Guide* Figure 1-1). Continuous traffic count locations need to be grouped based on identified traffic patterns for time of travel consistency such as hour-of-day distributions, day-of-week distributions, and monthly factors. The 2016 and 2022 versions of the *Traffic Monitoring Guide* both present several methods that transportation agencies may follow for grouping count locations, including the 'traditional approach' (which uses general knowledge of the road system and monthly graphs to identify patterns and define groupings), cluster analysis, and volume factor groups. The *Traffic Monitoring Guide* also provides guidance on data collection for vehicle classification (e.g., axles and length), speed, weight, and lane occupancy.

Short-duration counts furnish most of the geographic coverage for traffic monitoring programs. Recommended durations of counts can vary from 48 hours to a week or more, with longer-duration counts having been shown to produce more accurate volume estimates (2016 *Traffic Monitoring Guide* Page 3-77, 2022 *Traffic Monitoring Guide* page 3-43). Since short-duration count locations are by definition not continuous, the frequency of counts needs to be set, and the TMG recommends that counts be conducted at least once every six years or less for all non-NHS lower functional system roadway sections (2016 *Traffic Monitoring Guide* Page 3-80, 2022 *Traffic Monitoring Guide* page 3-43), and potentially more often for some roads such as those experiencing growth in traffic volumes or that belong to a higher functional class. Some short-duration count locations should also have vehicle classification equipment to allow for estimation of annual average daily truck traffic (AADTT) in addition to other purposes. At least 25 – 30% of short-duration count locations should include classification counting equipment, or more when agency resources allow (2016 *Traffic Monitoring Guide* Page 3-78, 2022 *Traffic Monitoring Guide* Table 3-18). The *Traffic Monitoring Guide* recommends 48 consecutive hours for vehicle classification counts with variation on a case-by-case basis (2016 *Traffic Monitoring Guide* Page 3-56, 2022 *Traffic Monitoring Guide* Page 3-87).

## Calculations for Deriving AADT

Estimating AADT and AADTT from short-duration traffic count locations requires using factors to adjust for traffic patterns by time of day, day of week, and seasons. Figure 2-3 shows an example of the day-of-week traffic distributions that may be observed at count locations, showing both traditional urban traffic patterns with higher weekday than weekend travel and traffic patterns more typical of roads used for recreation, which usually show slightly higher weekend than weekday travel. For extremely short-duration traffic counts (less than 24 hours), there are additional adjustments to be made using data collected from continuous count locations (2016 *Traffic Monitoring Guide* Page 3-95). The 2016 *Traffic Monitoring Guide* provides procedures for computing the following statistics. They are not summarized here since none of these are directly related to calculating AADT on off-system roads.

- Average daily truck traffic (ADTT)
- Annual average daily truck traffic (AADTT)
- Axle correction factors
- Factors for converting daily truck traffic counts into estimates of AADTT (by class)
- Factors that allow conversion of AADTT estimates (by class) into average day of week estimates for use in the draft National Cooperative Highway Research Program (NCHRP) 1-37A Pavement Design Guide
- Sum of FHWA heavy vehicle classes 4 – 13 for 24 hours (Vehicle classes are shown in Figure 2-4)
- % Single Unit
- % Combination Unit

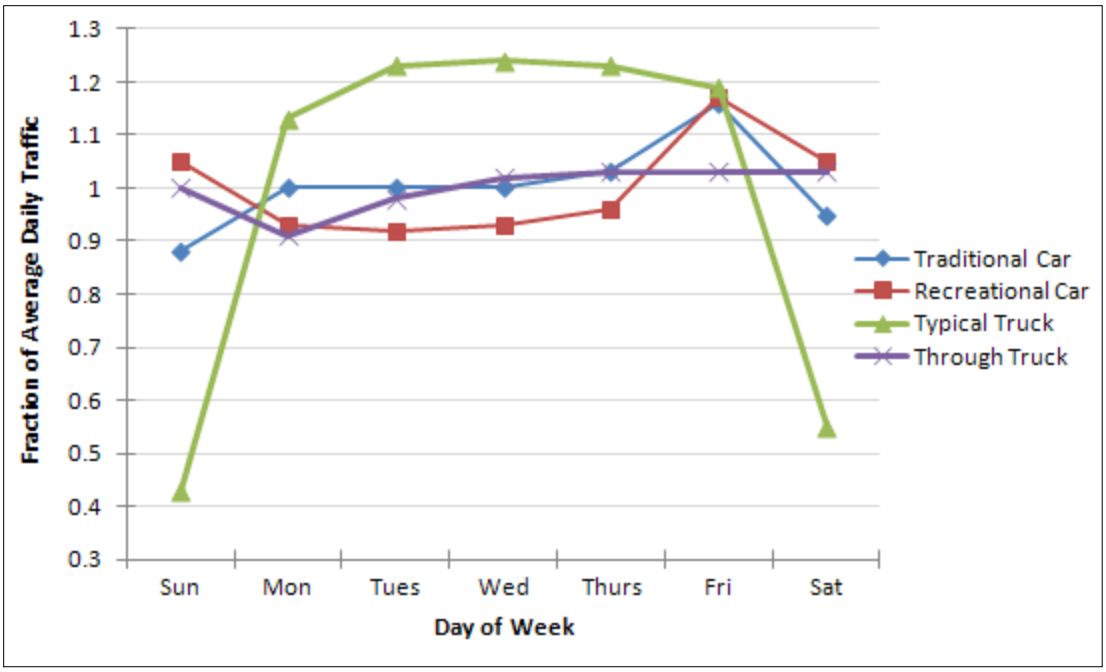


**Figure 2-1. Example of Day of Week Travel Patterns**

**Source: Traffic Monitoring Guide, Figure 1-4, Federal Highway Administration (2016)**

**Note: This graphic shows example travel patterns observed in many locations in the United States and is not specific to any particular location in Idaho.**

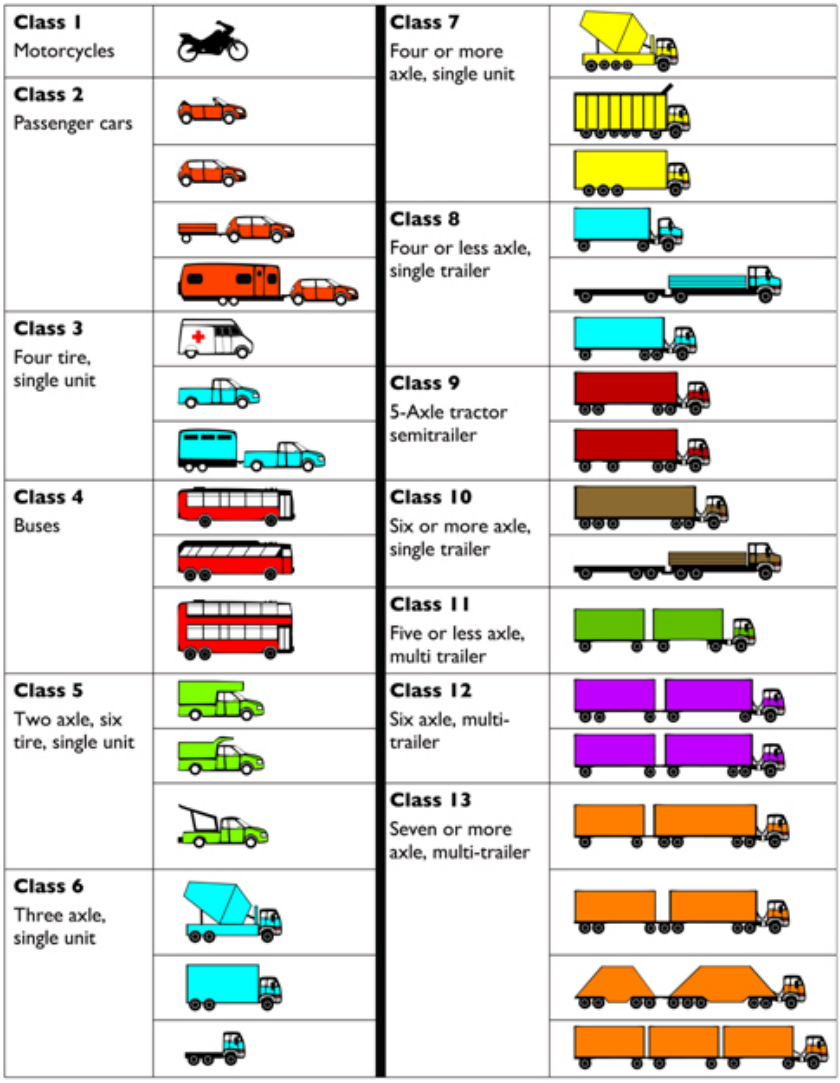


**Figure 2-2. FHWA Vehicle Category Classification**

**Source: *Traffic Monitoring Guide*, Figure C-1, Federal Highway Administration (2016)**

Additionally, the 2022 *Traffic Monitoring Guide* provides commonly used equations for estimating AADT based on continuous and short-duration counts, including Equation 1 and Equation 2 for calculating AADT based on continuous counts, and Equation 3 for estimating AADT from short-duration counts. While these equations or closely related versions of them are often used to calculate AADT where traffic counts are available, they cannot be used in their existing forms to estimate AADTs in the absence of traffic counts, which are not available for most off-system public roads in Idaho.

**Equation 1. Monthly average daily traffic calculation from continuous counts**

$$MADT_m = \frac{\sum_{j=1}^{7} w_{jm} \sum_{h=1}^{24} \left[ \frac{1}{n_{hjm}} \sum_{i=1}^{n_{hjm}} VOL_{ihjm} \right]}{\sum_{j=1}^{7} w_{jm}}$$

**Equation 2. AADT calculation from continuous counts**

$$AADT = \frac{\sum_{m=1}^{12} d_m * MADT_{HP_m}}{\sum_{m=1}^{12} d_m}$$

Where,

- $AADT$ is annual average daily traffic.
- $MADT_{HPm}$ is monthly average daily traffic for month m.
- $VOL_{ihjm}$ is total traffic volume for *i*th occurrence of the *h*th hour of day within *j*th day of week during the *m*th month.
- $i$ is the occurrence of a particular hour of day within a particular day of the week in a particular month (*i*=1,…$n_{hjm}$) for which traffic volume is available.
- $h$ is the hour of the day (*h*=1,2,…24) – or other temporal interval.
- $j$ is the day of the week (*j*=1,2,…7).
- $m$ is the month (*m*=1,…12).
- $n_{hjm}$ is the number of times the *h*th hour of day within the *j*th day of week during the *m*th month has available traffic volume ($n_{hjm}$ ranges from 1 to 5 depending on hour of day, day of week, month, and data availability).
- $w_{jm}$ is the weighting for the number of times the *j*th day of week occurs during the *m*th month (either 4 of 5); the sum of the weights in the denominator is the number of calendar days in the month (i.e., 28, 29, 30, or 31).
- $d_m$ is the weighting for the number of days (i.e., 28, 29, 30, or 31) for the *m*th month in the particular year.

**Equation 3. AADT calculation from short-duration counts**

$$AADT_{hi} = VOL_{hi} \times M_h \times D_h \times T_h \, x \, A_i \times G_h$$

Where,

- $AADT_{hi}$ is the annual average daily travel at location *i* of factor group *h*.

- $VOL_{hi}$ is the 48-hour axle volume at location *i* of factor group *h.*
- $M_h$ is the applicable monthly (seasonal) factor for factor group *h.*
- $D_h$ is the applicable DOW factor for factor group *h* (if needed).
- $T_h$ is the applicable TOD factor for factor group *h* (if needed for any partial day counts).
- $A_i$ is the applicable axle-correction factor for location *i* (if not a traffic volume or class count, i.e., for counts collected using a single pneumatic road tube).
- $G_h$ is the applicable yearly change (i.e., growth or decline) rate factor for factor group *h* (if needed).

For traffic counts submitted to HPMS, FHWA recommends longer counts for lower-volume roads. Specifically, "the TMG recommends a minimum of a 24-hour monitoring period for roads with traffic volumes of greater than 5,000 AADT and a 48-hour monitoring period for lower-volume roads" (page 5-10). The longer recommended duration for lower-volume roads is intended to increase the accuracy of AADT estimates (page 3-44). FHWA does not recommend counts below 24 hours, though if they are collected then hour-of-day adjustment factors can be used to estimate AADT (page 3-56) (Federal Highway Administration 2016).

## Data Submission

While the *HPMS Field Manual* provides authoritative requirements for data submission, the *Traffic Monitoring Guide* also provides guidance to states in meeting the HPMS program traffic reporting requirements. One broad recommendation is to make the state's traffic monitoring program mirror HPMS data reporting so that information published by FHWA and the state remain as similar as possible.

# Literature Review

This section reviews academic literature and documented practices related to geospatial interpolation models and how they can be applied to estimating transportation system characteristics and performance metrics such as AADT on off-system public roads and roads with similar characteristics. This information serves as the theoretical foundation for the estimation and validation tools developed for ITD to use to estimate traffic volumes on off-system public roads in Idaho.

Previous research (RP 303: "Off-System Public Roads Annual Average Daily Traffic Estimation Study", 2022) presented a comprehensive overview of methods for estimating off-system AADT researched in the United States or implemented by state DOTs. The project team referenced Transportation Research Board research, NCHRP projects, state DOT documents, and FHWA guidance to ensure it comprehensively identified efforts over the last 20 years regarding AADT estimation, off of the state-owned highway system or off of the Federal-Aid System. The project team also conducted searches on academic search engines (e.g., Google Scholar) that included journals such as the *Transportation Research Record* and the *Journal of Transport Geography*, and it reviewed the chain of citations for published work to identify relevant peer-reviewed academic research.

Methods reviewed include improvements to existing sampling methods, regression models, geospatial interpolation models, machine learning supported techniques, travel demand modeling, and network centrality analysis. Through discussion with technical experts and ITD staff, geospatial interpolation was selected as the preferred method for a future toolbox to analytically supplement direct traffic observations.

The remainder of this section highlights relevant examples in practice from the previous report, describes underlying theory and algorithms used in the specific type of geospatial interpolation selected in the previous project (Kriging), and summarizes the validation process and unique challenges in this specific situation.

## State-Level Analyses & Practices

### Surveys and Assessments in Literature

Publicly available information on current state DOT practices for estimating AADT is primarily focused on facilities that are counted for short periods (1 – 3 days) on a rotating basis, focusing on the process for estimating annual values for years between counts. A 2018 report by Portland State University and Oregon DOT reported on several other states' practices for estimating AADT on roads that directly count data, some of which included methods for non-state-owned facilities. This information was gathered as part of a survey (Unnikrishnan, et al. 2018), and the results indicate that state DOTs' approaches to estimating local access AADT are typically ad-hoc with reliance on grouping facilities by functional characteristics or land use intensities.

More recently, Huynh et al. (2021) surveyed 17 state DOTs over the summer of 2020 on their methods for estimating AADT on facilities that are not directly counted. Although the names of the participating states are not listed in the publication, the most common processes for locations with no recent counts within the prior ten years on the facility in question or on nearby facilities were multiple linear regression, visual estimation, and default values (Huynh, et al. 2021).

The two reports summarized above indicate that there is little industry consensus in estimating AADT on off-system public roads. This is further supported by conversations with ITD staff about feedback on their presentations at a recent conference on the results of the previous report's findings. Overall, there was substantial interest in seeing the effectiveness of a spatially driven estimation method, especially in a more rural state like Idaho.

## AADT Estimation Research

This section summarizes efforts by various state DOTs and academic institutions to estimate traffic volume on off-system public roads across a wide range of climates, terrains, population densities, and other state characteristics occurring since 2000. These projects were for research purposes and had not appeared to have been adopted at the time of this report. Table 2-1 summarizes the locations of prominent research efforts reviewed in the previous project. Notably, various forms of geospatial interpolation research have been conducted in several states, but research did not show that any state agencies had implemented such a process into their operating procedures.

**Table 2-1. Summary of Estimation Method by Analysis Location**

| State | Sampling | Regression | Geospatial | Machine Learning | Travel Demand Model (TDM) | Network Analysis |
|---|---|---|---|---|---|---|
| **Alabama** | -- | X | -- | -- | -- | -- |
| **Florida** | -- | X | X | -- | X | -- |
| **Georgia** | X | -- | -- | -- | -- | -- |
| **Idaho** | -- | -- | -- | X | -- | X |
| **Indiana** | -- | X | -- | -- | -- | -- |
| **Kentucky** | -- | X | -- | -- | -- | -- |
| **Louisiana** | -- | -- | -- | X | -- | -- |
| **North Carolina** | -- | -- | X | -- | -- | -- |
| **Ohio** | X | -- | -- | -- | -- | -- |
| **South Carolina** | -- | -- | X | -- | -- | X |
| **Tennessee** | -- | -- | -- | X | -- | -- |
| **Texas** | -- | -- | X | -- | -- | -- |
| **Vermont** | -- | -- | -- | X | -- | -- |
| **Washington** | -- | -- | X | -- | -- | -- |
| **Wyoming** | -- | X | -- | -- | -- | -- |

*Note: A dashed line indicates that the estimation method has not been tested in the state, and an "X" indicates that the estimation method has been tested in the state.*

## Review of Geospatial Models

Underpinning the use of geospatial data in statistical analysis is the concept summarized by Waldo Tobler's first law: "Everything is related to everything else, but near things are more related than distant things" (Tobler 1970). The statistic which typically represents this phenomenon is Moran's I, calculated as the correlation coefficient between a variable for a given geography and the same variable for neighboring geographies. Neighbors can be classified with any set of arbitrary rules, but some common methods include adjacency, k-nearest neighbors, and distance thresholds. If an independent variable or set of variables can be confirmed to have statistically significant spatial correlation with the dependent variable in question, it can be used in geospatially weighted regression models and supplemented with various interpolation techniques.

Geospatial models and their interpolative methods fall into two general categories: deterministic models which use arbitrary values, and statistical models which choose parameters based on data metrics. Regardless of the category, spatial analysis uses weighted averages to predict values where measurements are absent.

A simple form of estimating spatial values across a geography from a set of discrete samples is proximity interpolation. The method generates polygons where edges lie along the midpoints between sampled locations. These "Thiessen" polygons then inherit the attributes of the sample point which they enclose, but, without a large sample size, the resulting surface does not generally reflect natural distributions of data.

The next expansion of this process is to include multiple sample points for an unsampled location. For example, the five nearest points can be averaged (with or without weighting) to estimate a value. This is referred to as "k-nearest neighbor" interpolation.

The inverse distance weighted technique calculates values for unsampled locations with weighted averages from nearby locations based on a power coefficient. Larger coefficients create greater drop-offs in influence as the distance to other locations increases. The power coefficient selection is a subjective process unless the analyst optimizes to a selected evaluator measure, e.g. RMSE for existing AADT counts.

Trend surfaces use polynomials in a similar way to linear regression to estimate two-dimensional relationships between sampled points. A "$0^{th}$" order trend surface is the mean value of all sampled points. A first order trend surface improves this to a trend direction, and a second order trend surface generates a parabolic relationship. The continued addition of higher order terms may increase accuracy, but the success of this method greatly depends on the phenomena under consideration.

Kriging, named after Danie Krige a statistician and mining engineer from South Africa, is a Gaussian interpolation process governed by prior covariances. In practice, this is implemented by extracting spatial trends via the trend surface process described previously, computing a semivariogram to measure spatial autocorrelation, selecting a model to characterize the semivariogram (illustrated in

Figure 2-7), and predicting values at unsampled locations. Several varieties exist, but the most common are Ordinary Kriging (constant unknown mean) and Universal Kriging (general polynomial trend model). Although it requires several more steps, this technique addresses weaknesses due to subjectivity in other spatial interpolation methods. Equation 4 shows the equation for a semivariogram (Mathew 2020).

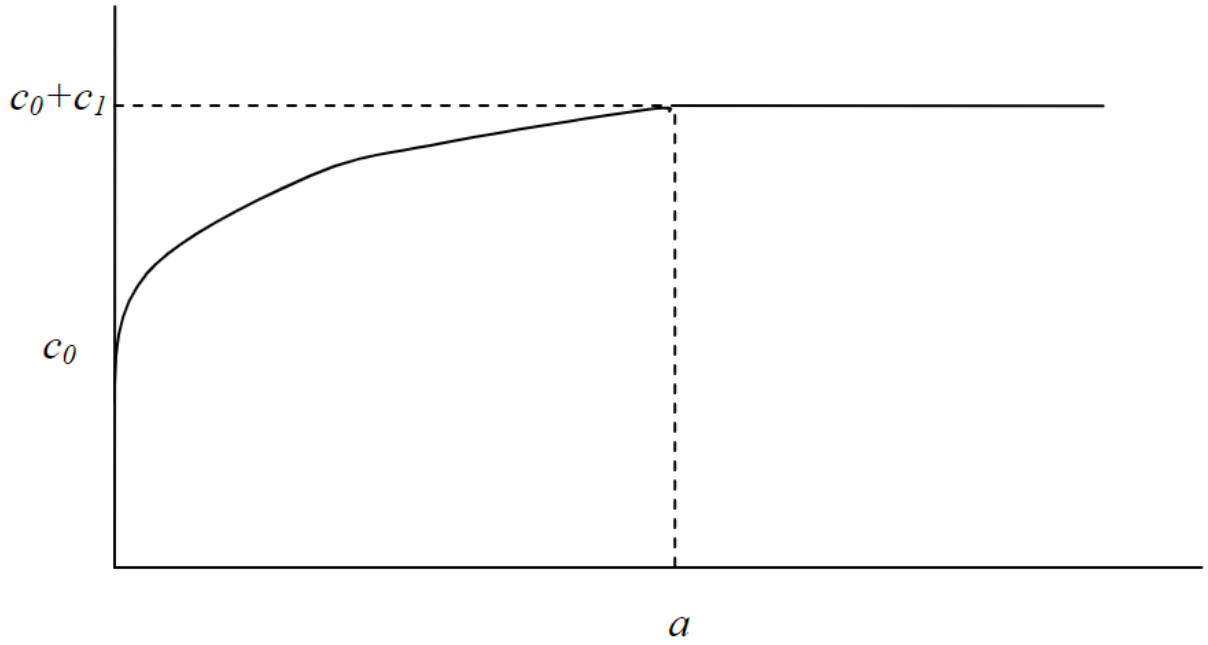


**Figure 2-3. Illustration of a Semivariogram**

**Source: Wang and Kockelman (2009)**

**Equation 4. Semivariogram formula**

$$S(h) = \frac{1}{2 \times N(h)} \times \sum \left(z_i - z_j\right)^2$$

Where,

- $S(h)$ is the semivariance for a given distance or lag, $h$.
- $N(h)$ is the number of pairs of observations separated by the distance $h$.
- $z_i$ and $z_j$ are the values of the variable being studied at two locations $i$ and $j$ that are separated by the distance $h$.

Semivariograms describe the similarity or difference between observations based on the distance between them and other observations in the dataset. While semivariograms are used across many types of spatial interpolation, Kriging also considers the arrangement of observations, quantifying a matrix of

weights for each point that define how much or how little they contribute to the model that estimates values for unobserved points.

Once the semivariance and related distances have been calculated to form the points of a semivariogram, a model of best fit, like the one shown in Figure 2-7. Illustration of a Semivariogram is selected to act as the underlying model for the Kriging process. Common model types include circular, spherical, exponential, gaussian, and linear. However, research has shown that which model best reflects the data can change from year to year and between types of Kriging models (Owaniyi 2019).

Most geospatial analysis software packages have the capacity to perform any of these processes, and spatial interpolation has been applied in several transportation related or adjacent fields such as transit ridership estimation (Marques and Pitombo 2021), geotechnical surveying (Liu, et al. 2021), and air quality modeling (Kumar, Mishra and Sarma 2020). Despite the need for greater data input density and computational resources, research on estimating rural facilities like the study conducted by Huynh et al. in South Carolina (2021) saw considerable improvements when compared to statewide or county-level average AADT value assignments. These improvements reflect geospatial interpolation's ability to reflect the inherently spatially-dependent nature of the transportation system and the communities it serves.

**References for Geospatial Interpolation and Weighted Regression:** Zhao and Chung (2001), Zhao and Park (2004), Eom et al. (2006), Wang and Kockelman (2009), Pulugurtha and Kusam (2012), Selby and Kockelman (2013), Shamo, Asa and Membah (2015), Owaniyi (2019), Huynh et al. (2021), Baffoe-Twum et al. (2023)

## Explanatory Factors Considered in Literature

The real-world information provided to the models reviewed varied widely in the literature. However, the information referenced by these methods can be divided into four broad groups: information about the physical condition and design of the facility, the relationship between the facility and the wider network, socio-economic data about the surrounding area, and the demographics of nearby populations. A summary of methodologies' minimum data requirements is shown in Table 2-8, and recurrent data points from the literature reviewed is displayed in Table 2-9.

**Table 2-2. Minimum Data Requirements by Methodology**

| Estimation Method | Observed AADT | Count Site Location | Road Network | Functional Classes | Population | Workers |
|---|---|---|---|---|---|---|
| **Sampling** | X | -- | -- | X | -- | -- |
| **Regression** | X | -- | -- | X | -- | -- |
| **Geospatial** | X | X | -- | -- | -- | -- |
| **Machine Learning** | X | -- | -- | X | -- | -- |
| **TDM** | X | X | X | X | X | X |
| **Network Analysis** | X | X | X | X | -- | -- |

*Note: A dashed line indicates that the data type is not strictly required to minimally use the method, and an "X" indicates that the data type is strictly required to minimally use the method.*

**Table 2-3. Common Explanatory Factors Used in Literature**

| Road Characteristics | Network Characteristics | Socio-Economic Data | Demographic Factors |
|---|---|---|---|
| Functional Classification | Distance to intersection | Urban / Rural Designation | Population |
| Number of Lanes | Accessibility (to primary or secondary roads) | Workers | Dwelling Units |
| Speed Limits | Centrality Measures | Employment by Industry | Vehicle Registration |
| Surface Material | Road Mileage Density | Median Income | |
| | | Poverty Rates | |

## Validation of AADT Estimates for Off-System Public Roads

Most practitioners and researchers validate their results using cross-validation—a process of reserving a portion of their locations from the model calibration process and using it to compare to the estimated AADT produced by their process. This process is often repeated across multiple defined subsets of the data. From the literature reviewed that estimates AADT for off-system public roads, reserving 25% of off-system road locations is most common (Mathew 2020, Staats 2016, Tsapakis, Holik, et al., Informational Guide on Data Collection and Annual Average Daily Traffic (AADT) Estimation for Non-Federal Aid-System (NFAS) Roads 2020, Pulugurtha and Mathew 2020), with one study reserving 30% for validation (Raja, Doustmohammadi and Anderson 2018).

There are several plots and statistics that are commonly used for validation, including visually examined scatter plots (Raja, Doustmohammadi and Anderson 2018), mean absolute error (MAE) (Staats 2016), mean absolute percentage error (MAPE) (Tsapakis, Holik, et al., Informational Guide on Data Collection and Annual Average Daily Traffic (AADT) Estimation for Non-Federal Aid-System (NFAS) Roads 2020, Mathew 2020, Staats 2016, Pulugurtha and Mathew 2020), mean percent error (Mathew 2020, Pulugurtha and Mathew 2020), root mean squared error (RMSE) (Mathew 2020, Pulugurtha and Kusam 2012), R-square (Raja, Doustmohammadi and Anderson 2018), and the Nash-Sutcliffe coefficient (Raja, Doustmohammadi and Anderson 2018). While less used in the literature, an R-squared value or information criterion (such as the Akaike information criterion (AIC) or the Bayesian information criterion (BIC)) may also be used.

## Challenges for Estimating AADT on Off-System Public Roads

The requirement for state DOTs to estimate AADT on all local public roads comes in part from HSIP, which is requiring every public paved road to have AADTs made available by 2026. There are several challenges facing ITD in preparing estimates for this requirement, many of which align with the challenges noted in the literature such as institutional knowledge and data availability.

At 22.3 people per square mile in 2020, Idaho is the 44$^{th}$ least population dense state in the U.S., and a few rural roads on state highway system are unpaved (< 0.1%). However, unpaved local roads are often used for going to a site (like a campground) at the end of the road and for passing through areas where a paved road cannot be maintained. Therefore, use patterns on unpaved state system roads are not analogous to paved local roads.

# 3. Estimation Methodology

Based on TAC feedback in the previous project, Kriging geospatial interpolation method was selected because it provides lower data collection requirements, a high degree of flexibility and adaptability, high ease of explanation about the process, and relatively high accuracy (Hylton and Miller 2023). Table 2-1 details how the methodology will meet ITD's requirements for estimating AADT. The remainder of this chapter describes the proposed AADT estimation methodology with step-by-step guidance for development of the toolbox. Sections include data collection, pre-processing, Kriging interpolation, and post processing.

**Table 2-1. Summary of Kriging-Interpolation Requirements**

| Requirement | Solution |
|---|---|
| Incorporate Idaho's urban vs rural road segment designations (e.g., Urban Cluster, Small Urban Area, and Urbanized Area) | The interpolation should be run separately for urban and rural roads or include a variable distinguishing urban and rural roads. |
| Incorporate Idaho's paved vs unpaved roadway characteristics | The interpolation includes a variable distinguishing paved and unpaved roads, and independent models will also be developed for each surface type. |
| Balance with the statewide annual VMT | The raw AADT estimates will be scaled to balance with statewide annual VMT. |
| Take into consideration any other inputs that the survey and the review of literature may suggest | Universal or Cokriging will be used to incorporate independent variables identified in the literature and by the previous project that are correlated with facility-level AADT. |
| Be feasible to implement using geospatial tools that are accessible to ITD | The methodology will be implemented in ArcGIS Pro. |
| Span the entire state network | Roadway, network, socioeconomic, and demographic factors will be analyzed at the state level. |
| Produce estimates in conformance with the FHWA *Traffic Monitoring Guide* | No inconsistencies |
| Be possible to validate | The validation methods discussed in this report will compare outputs with selected count locations. |

## Data Collection

### Step 0: Gather Data

Based on the list of variables provided in this report's section on data requirements, spatial data will be collected from both ITD and federal sources to describe the road network of interest and its local contexts. These files should all be saved in the same directory within the project folder where the analyst is working.

# Pre-Processing

## Step 1: Import and Clean Data

Geospatial interpolation estimates AADT at locations where they are unknown based on AADT at nearby locations where it is derived from traffic counts. Therefore, the first step is to collect count-derived AADT. Counts from prior years can be used if they are updated with a growth factor to estimate current year AADT. For instance, DeVine (2020) used ten years of AADT data. Data from prior years should be updated to the current year based on growth rates and seasonality factors if applicable. Only the most recent AADT should be used at a given location. If duplicate geometries exist within the observed data set, errors will likely arise from the underlying statistical functions. It may also be necessary to remove extreme AADT outliers, which can be done using the interquartile method described by DeVine (2020), who removed outliers from short-duration counts that skewed toward high AADTs.

Off-system public roads without count-derived AADTs are imported into an estimation data set. Any additional data included in these files that will be utilized in the analysis process should be cleaned at this point, substituting “NA” or “NULL” values with appropriate assumptions (e.g., unpaved for the surface type of remote facilities), removing or fixing invalid geometry, and checking for data entry errors.

Additional data sets are also cleaned and prepared for conflation with the roadway data set at this stage. These additional data sets may include roadway characteristics (e.g., number of lanes, functional class, surface type), economic and demographic characteristics (e.g., employment density, population density), and network characteristics (e.g., network centrality).

## Step 2: Prepare Data Set for Estimation

Off-system public roads without count-derived AADTs are imported into an estimation data set. Some studies have conducted cleaning of this data set, such as removing dead-end roads and driveways, and combining multiple sequential segments into a single segment (DeVine 2020). This tool will estimate AADT segment mid-points.

Additional characteristics are also associated with roadway segments in this step. Each data set that may be used as a variable in the regression should be associated with the entire network. The data sets should be converted to variables and only the variable values associated with the network rather than the raw values. For instance, raw roadway surface types are condensed to just two values: paved and unpaved. Similarly, raw population values within Census block groups may need to be converted to population density by dividing by land area.

The entire network includes both on-system and off-system public roads since there may not be enough off-system traffic counters to derive the relationship with AADT purely from off-system public roads. This means that segment-based data whose extent or segmentation differs from the final network

should be assigned to the network for which AADT will be estimated. Location-based variables and variables derived from network analysis should also be calculated for segments in that network.

Table 2-2 below summarizes how to compile data into a single data set aligned with roadway segments. All variables that will be considered for the analysis should be included in this final data set, including AADTs that are directly derived from traffic counters. AADTs produced through the ITD's existing traffic smoothing process should be omitted so that the relationship between AADT and other variables is derived from known AADTs only. The result of this step is a geospatial file (e.g., geodatabase, shapefile) with the dependent variable (i.e., AADT) where derived from count locations and with all independent variables that may be used in the model.

**Table 2-2. Steps for Compiling Data by Data Type**

| Data Type | Applicable Data Types | Approach |
|---|---|---|
| Segment-based data with a common unique identifier field and the same segmentation as the final network | Roadway Characteristics | Tabular join using shared unique identifiers as a join field |
| Segment-based data without a common unique identifier field but with route IDs and measures | Roadway Characteristics | Use the Overlay Route Events tool in Esri's ArcGIS Pro to join based on route IDs and measures. |
| Area-based data (such as data for Census tracts or block groups) | Demographic and economic characteristics | Geospatial assignment (e.g., intersection, buffers) |
| Variables derived from network analysis (e.g., betweenness, centrality) | Network characteristics | Calculate directly from the network |

Part of calculating variables will depend on treating missing data correctly when data is missing for some road segments. Data can be filled in if it is possible to deduce what the data would likely be. For instance, missing numbers of through lanes may be assumed to be two (one in each direction), and roads without surface type data may be assumed to be unpaved if in a rural area and paved if in an urban area. Table 2-3 below provides guidance for filling in missing data for key variables. The entries should be left null for variables where there is no default value or where the value of missing values cannot be reasonably deduced by examining the pattern of missing data.

**Table 2-3. Guidance for Filling in Missing Values**

| Variable | How to Fill in Missing Data |
|---|---|
| Number of through lanes | Assume two through lanes (one in each direction) if data is missing. |
| Paved vs. unpaved | If data on surface material is missing, assume unpaved if the segment is in a rural area and paved if it is in an urban area. |
| Functional class | Assume "off-system" designation for all segments for which functional class is unknown or that have not been functionally classified. |
| Urban vs. rural | If missing, use MPO boundaries, assuming urban inside of MPO boundaries and rural outside. |

The project team recommends beginning the analysis with the variables listed in this report's section on data requirements. Additional variables may be added as new or improved data sources become available; however, if there is no statistical or geospatial relationship between a variable and AADT counts, it should not be included unless the analysts or other team members identify a reasonable relationship between that factor and underlying trends that influence AADT. If multiple variables are found to be collinear, the variable with the greatest correlation to AADT can be retained and the others discarded.

The presence of heteroskedasticity in any factor indicates that a transformation (e.g., log, power) may be necessary or that the variable cannot be used. Log transformation of AADT values will be applied on a group-by-group basis to evaluate potential accuracy improvements.

Network characteristics have shown the capacity in the literature to greatly increase AADT estimation accuracy (Kehan 2017). Simple measures such as "miles of functional class 6 within five miles" can be calculated directly from a network shapefile without further data cleaning. However, more complex network measures such as "betweenness centrality" or "travel distance to the nearest primary arterial facility" require more extensive data preparation. A network graph cannot be created by assessing facility overlap since some facilities that overlap are not connected (e.g., highway overpasses). Therefore, if elevation values are not available within the roadway geometries, network preparation will only add intersection nodes when facilities of the same functional class intersect. Connection will also be made when endpoints connect to other endpoints or are immediately adjacent to another facility.

This network will then be used to calculate network centrality measures: closeness centrality, how close the facility is to all other facilities; betweenness centrality, how often a facility is part of a shortest route between origin-destination (OD) pairs; and degree centrality, the number of connecting facilities. The calculation of distances and centrality measures will be based on sets of origins and destinations provided by the user. For example, one set could provide census tract centroids as origins and county centroids as destinations, or another could provide county centroids as origins and recreational sites (campgrounds, national parks, etc.) as destinations.

Another network characteristic is the overall network density, e.g., the total length of the network or a subset of the network (such as local roads or interstates) within a given buffer area. A range for this buffer's radius will be identified during the toolbox development, but it is expected that including multiple values will be beneficial for annual updates to ensure that the best fit can be developed each year.

Once network characteristics have been prepared, routes will be segmented such that each portion is associated with a unique value from each characterizing data set. For example, routes that have paved and unpaved portions should be split by surface type, and routes that span multiple counties should be split to have unique numbers of registered vehicles at the county level. Once all routes have been split according to facility characteristics and external geographies, all external data sets can be joined to the

facilities of interest. If splitting routes is not sufficiently reliable, then characteristics will be assigned based on the extent of overlap.

The last step for the network will be calculating the midpoint for each segment. The resulting point geometries will retain the information associated with their respective segments. These points will serve as the basis for network-related independent variables' inputs into the Kriging process.

The Kriging interpolation process requires discrete dependent variables and continuous independent variables. Therefore, AADT estimates will be associated with the coordinates of their respective count locations, and all other information will be converted to raster formats by the tool. For the network, this will be accomplished by calculating segment midpoints and using the nearest value for categorical or ordinal variables and linear spatial interpolation (e.g., inverse distance weighting with a power of one) for continuous variables. For polygon-based data such as population density, conversion occurs directly with no interpolation required. This overlapping "stack" of rasters will be considered together in the Kriging process with an iterative loop that test removing or adding individual variables to identify the subset that best fits existing AADT counts and estimates.

For this iterative process, facilities will be grouped based on their location, road category, and surface type and then split into "training" and "testing" sets. The literature has often used approximately 80% of count locations for training the model and retained the remaining 20% used for testing or 'validating' estimates, although training sets have been observed to vary between approximately 75% to 90% of the data (Staats 2016, DeVine 2020, T. Wang, Improved Annual Average Daily Traffic (AADT) Estimation for Local Roads using Parcel-Level Travel Demand Modeling 2012). Subsequent analysis is performed on the training set, and the testing set is to be reserved for evaluation of accuracy in the Validation Plan. Locations included in the training and testing data sets should be randomly selected to avoid the possibility of bias.

## Kriging Interpolation

Much like regression analysis, Kriging interpolation utilizes 'best linear unbiased predictions' to compute weighted averages in spatial contexts. First, a formula of dependent and independent variables is used to calculate an empirical semivariogram. The resulting values are then associated spatial distances and semivariance (describing how random the relationship appears to be). A model is fitted to this relationship and then used to make predictions given a broader set of points of interest.

Explanatory variables such as population density, network connectivity, and roadway surface type are used directly in "Universal Kriging" and "Cokriging" where predictions are made based on a raster of values for those variables. Equation 1 summarizes Universal Kriging. The first summation of Equation 1 models the deterministic component and the second models the stochastic component.

**Equation 5. Generalized method for Universal Kriging**

$$\hat{Z}(s_0) = \sum_{k=0}^{P} \widehat{\beta_k}\, q_k(s_0) + \sum_{i=1}^{N} \lambda_i\, e(s_i)$$

Where,

- $\widehat{\beta_k}$ are the estimated deterministic model coefficients.
- $q_k$ is the matrix of predictors at the known location.
- $\lambda_{i\prime}$ are the Kriging weights determined by the spatial dependence structure of the residual.
- $e(s_i)$ is the residual at a known location.

Universal Kriging interpolation has the capacity to integrate independent variables directly into the semivariogram and modeling processes. This means that ordinal factors that would be used for grouping with other techniques, such as number of through lanes, can be incorporated directly. Categorical variables, such as functional class and surface type, should be integrated with caution to ensure that the model is treating them as categorical variables rather than interval or continuous variables.

### Step 3: Interpolate AADT

Once independent variables and model parameters have been set, the main Kriging tool can be run to estimate AADT at each uncounted segment's midpoint. The outputs of this process are the toolbox's primary output, representing a modeled number of vehicles on off-system roads.

This process will be iterated several times to test different combinations of independent variables and model parameters and their impact on AADT estimates. This iterative validation process will be used to determine the minimum set of information that creates the best fit to the Statewide annual VMT Estimate.

## Post Processing

### Step 4: Scale Resulting AADT Forecasts to Sum to Statewide Annual VMT

The resulting AADTs should be adjusted so that they sum with on-system AADT estimates to equal the statewide VMT estimate. A multiplier is applied to all off-system AADT estimates to adjust them enough so that they sum with on-system AADT estimates to equal the statewide annual VMT estimate. Equation 2 and Equation 3 describe how to calculate that multiplier.

**Equation 6. Off-system VMT**

$$Off - System\ Annual\ VMT = 365 \times \sum_i AADT_i \times length_i$$

Where,

- $i$ is an off-system road segment.
- $AADT_i$ is the unadjusted AADT estimate for segment $i$.
- $length_i$ is the length in miles of segment $i$.

**Equation 7. Scaling multiplier**

$$multiplier = \frac{statewide\ VMT}{on\ system\ VMT\ + off\ system\ VMT}$$

Where,

- $on\ system\ VMT$ is the estimated VMT for road segments currently estimated through the count program and smoothing process.
- $statewide\ VMT$ is the estimate of statewide VMT through existing processes.

## Step 5: Round Off-System AADT Estimates

Off-system AADT estimates should be rounded based on existing rounding rules described in the Task 2 report on current practice at ITD with the exception that values below 10 are permitted. Off-system roads may in some cases have very low average traffic volumes and rounding them up to ten may significantly overstate the true average traffic volume. Table 2-4 shows the rules for rounding off-system AADT estimates.

**Table 2-4. Rounding Rules for AADT**

| AADT Range | Rounding Precision |
|---|---|
| Less than 10 | Round to the nearest whole number |
| Less than 1,000 | Round to nearest ten |
| Equal to or greater than 1,000 and less than or equal to 10,000 | Round to nearest hundred |
| More than 10,000 | Round to nearest five hundred |

**Source:** Based on script in Smoothing Toolbox called AADTDataDeconstructor.py, with modifications for rounding of estimates below 10.

# Validation Plan

Validation is part of the data process for estimating AADT on off-system public roads. Validation occurs both to help select an optimal model for the geospatial interpolation approach to estimating AADT and then to confirm that the selected model produces AADT estimates that meet acceptable levels of accuracy. Since validation involves comparing estimated AADTs with AADTs calculated from observed counts, this plan also addresses the possibility of ITD eventually expanding its count program to collect counts in locations that are currently underrepresented among off-system counts. This expansion is unlikely to happen in the near future because there is a lack of additional resources with which to expand the count program.

## Assumptions

The following assumptions will be presented to relevant members of TAC to check and refine them to the extent possible.

- Functional classification is assumed to be class 7 (local) where it is unknown.
- Surface type will be assumed to be paved in urban areas and unpaved in rural areas where it is unknown.
- The number of through lanes is assumed to be two (one in each direction) where it is unknown.
- If roads are not designated as either urban or rural, then they are assumed to be urban if inside of MPO boundaries and rural otherwise.
- Road network density will be measured as the miles of roadway within **5** miles.

## Statistics

This subsection provides guidance for validating AADT estimates with measures assessing accuracy and bias. How these statistics will be used to evaluate model variations will be determined in coordination with the TAC.

Fundamentally, to assess accuracy is to compare each AADT estimate that is part of the validation group with AADT calculated from observed counts. While this can be done individually, there are statistics that exist to summarize accuracy for a set of data points. While accuracy refers to how close the estimated AADT is to the AADT calculated from observed counts, bias refers to whether AADT is generally overestimated or underestimated. Bias exists when errors are not comprised of overestimation and underestimation fairly equally. Validation is expected to occur using only traffic counts that ITD already regularly collects. Table 3-1 below presents these statistics.

**Table 3-4. Key Statistics for Validation**

| Name | Purpose | Interpretation | Sources |
|---|---|---|---|
| Mean error (ME) | To quantify bias in estimates | Zero signifies no bias in the estimates regardless of accuracy. A negative value indicates that observed AADT generally exceeds estimates, while a positive value indicates the reverse. | Morley (2016) |
| Mean absolute percentage error (MAPE) | To quantify accuracy of estimates | Zero signifies that estimates perfectly match observed AADT. A higher number means that there is more difference between estimated and observed AADT for an average location regardless of direction. Negative numbers are not possible. | Staats (2016), Pan (2008) |
| Root-mean-square error (RMSE) | To quantify accuracy of estimates | Values of zero or negative values are not possible. Values closer to zero signify more accurate estimates than higher values. Compared with MAPE, RMSE adds an extra penalty for estimates being farther away from observed AADT. | Moody (2019) |

While both mean absolute percentage error (MAPE) and root-mean-square error (RMSE) can be used to examine error, RMSE is more sensitive to the size of the difference between estimates and observed AADT. Thus, if there are two models whose estimates receive the same MAPE value, the one that has more uniform deviations between estimates and observed AADT will likely have a lower RMSE value while the one with a mix of very low and very high deviations will receive a greater penalty for those very high values and have a higher RMSE value (Morley 2016).

As shown in Equation 4, Mean Error (ME) can account for bias because it treats instances of overestimation differently from instances of underestimation. This contrasts with MAPE and RMSE, which use absolute values and square the difference between estimates and observed AADT respectively to ensure that instances of overestimation and underestimation are both positive. Equation 5 and Equation 6 describe MAPE (Pan 2008) and RMSE respectively (Moody 2019), and Python code facilitating the calculation of each statistic is included directly under each equation.

## Mean Error (ME)

**Equation 8. ME**

$$ME = \frac{1}{n}\sum_{i=1}^{n}(AADT_{Mi} - AADT_{Fi})$$

Where,

- $AADT_{Fi}$ is the i[th] count-derived AADT value.
- $AADT_{Mi}$ is the i[th] AADT value estimated by the estimation method.
- $n$ is the number of count locations retained for validation.

**Code to calculate ME in Python**

```
# Source: Statology (2020)

# Creating a Function for ME

import numpy as np

def mape(y_test, pred):

   y_test, pred = np.array(y_test), np.array(pred)

   me = np.mean(y_test - pred)

   return me
```

## Mean Absolute Percentage Error (MAPE)

**Equation 9. MAPE**

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{AADT_{Mi} - AADT_{Fi}}{AADT_{Fi}}\right|$$

**Code to calculate MAPE in Python**

```
# Source: datagy (2022)

# There is no built-in Python package to calculate MAPE in Python, so this function
# can be defined in the numpy package to calculate MAPE, using the following code.
# Creating a Function for MAPE
import numpy as np

def mape(y_test, pred):

   y_test, pred = np.array(y_test), np.array(pred)

   mape = np.mean(np.abs((y_test - pred) / y_test))

   return mape
```

## Root-Mean-Square Deviation (RMSE)

**Equation 10. RMSE**

$$RMSE = \sqrt{\sum_{i=1}^{n}\left(\frac{(AADT_{Mi} - AADT_{Fi})^2}{n}\right)}$$

**Code to calculate RMSE in Python**

```
# Source: Statology (2020)

#import necessary libraries

from sklearn.metrics import mean_squared_error

from math import sqrt

#calculate RMSE

rmse = sqrt(mean_squared_error(actual, pred))

return rmse
```

## Relative Accuracy

The purpose of assessments of relative accuracy is to determine which model or which version of a model produces the most accurate results and merits being retained for further development and ultimately for calculating the final estimates of off-system AADT. When, in the course of estimating off-system AADT, different versions of the geospatial interpolation models are tested containing different variables or with different model parameters, the versions' accuracy can be assessed by calculating these statistics for all versions and selecting the version that is most accurate according to the statistics described in Table 3-1 for further development. This may be an iterative process that occurs in several rounds. If there are certain types of count locations that are particular concerns for accuracy because of relatively the small number of count locations on these roads (e.g., low-volume roads, unpaved roads), then they can be split off from the rest of the data set after AADT estimates are produced and accuracy statistics can be calculated for them separately from other roads.

## Absolute Accuracy

Absolute accuracy demonstrates if there is bias in the final AADT estimates (i.e., the model is consistently underestimating or overestimating AADT) and how close estimates are to observed AADTs. The same statistics used for relative accuracy (shown in Table 3-1) are also used to assess absolute accuracy. There is no universal threshold for what constitutes a 'good' or acceptable score for each statistic for all use cases. What is acceptable depends on the use case. Table 3-2 below summarizes the values that other similar studies using geospatial or regression-based approaches have found as a basis for deciding what is acceptable. Regression-based models are included since they have performed with similar levels of accuracy to geospatial techniques in some research (Selby and Kockelman, Spatial

prediction of traffic levels in unmeasured locations: applications of universal kriging and geographically weighted regression 2013, Pulugurtha and Mathew 2020). A larger list of errors associated with different studies by other methods is included in research conducted by DeVine (2020).

**Table 3-5. Error Ranges from Prior Similar Analysis**

| Statistic | Range |
|---|---|
| ME | ME is not assessed in any of the reviewed studies. However, Wang and Kockelman (2009) obtained a median percentage error value of 33% using Kriging approaches, indicating slight overestimation of AADTs. |
| MAPE | Ranging from 32% to 160% depending on road type, with higher values for lower-volume roads (applied to all roads statewide from regression-based model) (Pan 2008).<br>Ranging from 85% to 100% depending on the region (applied to local roads from regression-based model) (Staats 2016). |
| RMSE | 73% for low-volume roads (ADT < 400) (based on regression) (Apronti, et al. 2016).<br>56% to 95% depending on the year (based on Kriging techniques) (Shamo, Asa and Membah 2015). |

In addition to reviewing accuracy for all roads, it may also be useful to assess accuracy for road types with relatively few count locations, such as low-volume roads and unpaved roads. To do this, count locations in the validation set can be observed separately for unpaved roads and for roads by AADT ranges, such as 0-99, 100-999, 1,000-9,999, and 10,000 or more. If bias or unacceptable inaccuracies are observed, then model adjustments may be required such as adding a variable to a Universal Kriging model that is closely associated with AADT, such as roadway intersection density, roadway mileage density, or population and employment density if not already included.

While the right model can produce reasonable and useful results with a high degree of accuracy, there will inevitably remain differences between estimates and observed AADT for reasons relating to statistical limitations, data availability, inaccuracies in input data, and the number of variables that it is possible to include in the model. These differences do not undermine the model results or their utility but rather point to the limits of estimates compared with counts. In percentage terms, these differences are likely to be greatest for low-volume roads where smaller deviations in absolute terms produce larger percentage variations. Additionally, the state of the art has not yet achieved perfect accuracy. Even the most recent research retains moderate deviations between predictions and observations, especially for lower-volume roads (Mathew 2020).

Even when results are highly accurate, it is important to properly explain them so that there are not apparent inaccuracies that simply reflect misunderstanding of what the results are intended to show. AADT represents average traffic over an entire year, while many counters and observers will be basing their observations on a snapshot in time (e.g., a given day, time of day, or week). Non-continuous counts and observations necessarily capture a snapshot of traffic that may not account for annual trends until seasonality, day-of-week, time-of-day, and other factors are accounted for. Therefore, if a person observes a higher or a lower volume of a roadway during the summer on a weekday, their observation will not reflect how traffic volumes might change on other days of the week or in other seasons. Single

observations could also be distorted by special events or occurrences that change traffic volumes away from what they would typically be at that time, on that day, or in that season. It is important to explain the difference between AADT estimates' purpose and more casual observations that may not account for factors that could cause variation (e.g., season, day, time of day, etc.) to avoid the appearance in inaccuracies in estimates where they are not truly present.

## Count Locations

Roads or areas with the least accurate estimates should be first to be added to ITD's count program to improve model accuracy and validation efforts. Additionally, the previous project analyzed the characteristics of off-system count locations and found the least representation in low-density rural areas below 100 people per square mile and for unpaved roads. If count locations can be added on more roads with these characteristics in the future, it will increase the amount of data available for geospatial interpolation and for validation, improving estimates' accuracy where it is likely to be needed most.

# 4. User Training Guide

This guide describes the process for operating the Off-System AADT Estimation toolbox. The Off-System AADT Estimation Toolbox estimates AADT values for off-system roads (where counts are not typically collected) using on-system traffic counts combined with demographic and geographic attributes. This toolbox collates the required data sources, applies a spatial interpolation algorithm to create AADT estimates, and balances these estimates against statewide annual Vehicle Miles Traveled (VMT). This toolbox is intended to be run annually to update off-system AADT estimates for federal reporting purposes.

As shown in Figure 1, data is collected from sources within the Idaho Transportation Department (ITD) as well as from the federal government and local agencies. This data forms the basis of the process shown in Figure 2 where the most recent AADT values are identified and rasters (an "image" that provides continuous values across a spatial extent) are created for the independent variables that will be used to estimate AADT at unmeasured locations. These are combined through a form of spatial interpolation called Kriging, specifically using Esri's Empirical Bayesian Kriging Regression, to model the spatial relationship between traffic volumes, the network, and community context. These initial estimates are scaled to equal the portion of the statewide annual VMT estimate that is not accounted for by the Federal-Aid System. As they are available to ITD, estimates are compared to AADT values based on counts on off-system roads by local agencies.

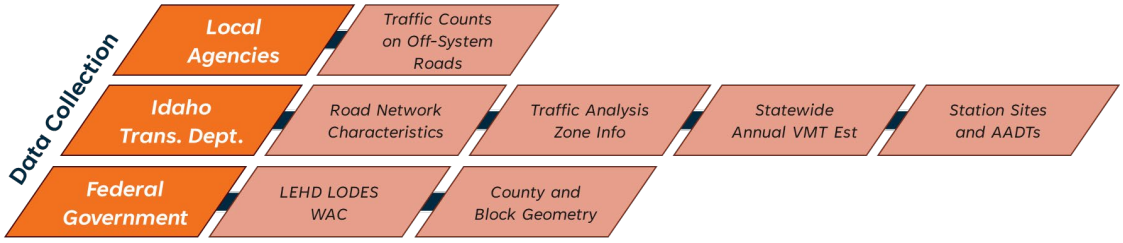


**Figure 4-4. Data collection groups by source**

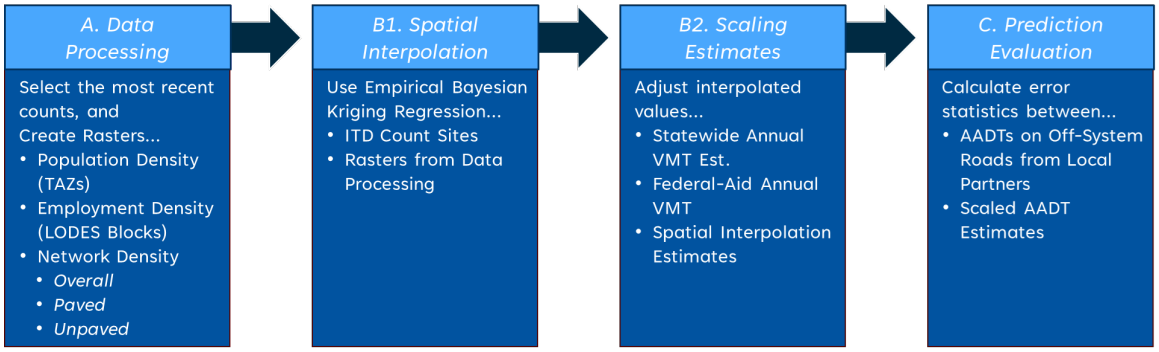


**Figure 4-5. Toolbox step outline**

This document describes the setup, development of initial data sources, step-by-step instructions for running the tools, and a conceptual framework for updating the tool for new or novel data sources should they become available through the efforts of ITD or its partners.

This guide is intended for tool users already familiar with navigating ArcGIS Pro. Data preparation and toolbox operation will require skills for managing tabular and spatial data files, generating intermediate data products, and using ITD's internal information systems.

The toolbox was developed in the ArcGIS Pro 3.3 environment. Computers that are able to run this software are expected to run the constituent parts of the toolbox in a matter of minutes, but it should be noted that some operations are computationally intensive as they utilize the entirety of the public road network in Idaho.

## Setting Up Your Project

Before collecting initial input data, complete the following steps.

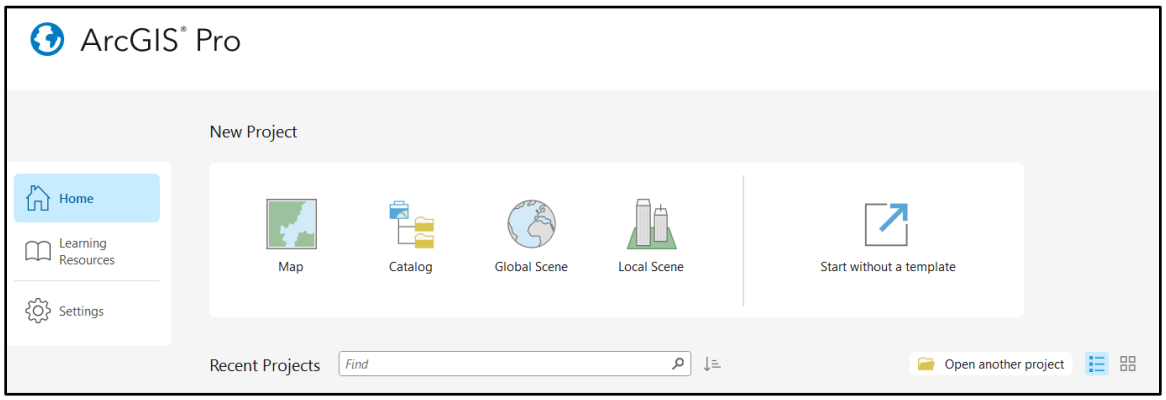


**Figure 4-6. Screenshot of the Esri ArcGIS Pro app when first opened**

1. Create a new ArcGIS Pro project by clicking the "Map" button when you open ArcGIS Pro.
2. Make sure your Location Referencing and Spatial Analyst licenses are active.
3. Clone your environment, Esri's documentation for more information to allow for installation of required Python packages in the next step. https://pro.arcgis.com/en/pro-app/3.3/arcpy/get-started/clone-an-environment.htm
    a. In the "Project" tab in the ArcGIS Pro project, click on the "Package Manager". From there, click the gear / settings icon next to "Active Environment", select "clone arcgispro-py3".
    b. In the destination box, click the folder button and navigate to your desired location, then click save.
    c. Viewing the destination box again, add a descriptive suffix (e.g., "\off_system_aadt_estimation") to name the environment.
    d. The new environment should appear in the Environment Manager.
    e. Activate the new environment by selecting it and clicking "OK".
4. Now in the dedicated environment, install the following Python packages through the ArcGIS Pro Package Manager from the "Project" tab.
    a. pandas
    b. geopandas

c. oracledb

    i. This is for connecting to ITD's internal systems. Other agencies may use a different method for connecting to agency databases.

    ii. From the C Drive, navigate to "ProgramFiles\ArcGIS\Pro\bin\Python\Scripts"

    iii. Then enter the command prompt.

    iv. Execute "activate ~" followed by the path to the newly created environment.

    v. Execute "conda install oracledb"

5. Create designated folders for data products within your ArcGIS Pro project's main folder. These folders should be where the user directs all tool outputs into. By default, select the processing folder. Only Tools B2 and C1 create outputs that should be considered outputs of the toolbox as a whole.

    a. Raw Inputs: Files downloaded or received from sources or other groups within ITD.

    b. Processing: Items generated by tools within this toolbox that are used in later steps.

    c. Outputs: Final products including the estimated AADTs and accuracy evaluations.

# Gathering Initial Data

Next, proceed to gather files as described for the following datasets: Traffic Analysis Zones (TAZs), counties, employment, and census blocks. For all shapefile inputs, include all the files (e.g., .cpg, .db, .prj, .shp, .shx).

It is important to note that temporality plays a key role in this analysis. Traffic counts and the status of the road network should reflect the conditions found in the desired year of analysis. For example, if the user is developing estimates for 2022, a roadway that was constructed in 2024 should not be included in the network. Due to the nature of date-time formatting in various data storage formats, it is the responsibility of the user to ensure that the correct features are provided to the tools as inputs.

This is typically done by using the "Select By Attribute" tool to select features with a "From Date" that is before the year of analysis and a "To Date" that is at the end of or after the year of analysis. "Not Applicable" or "null" values may also be used by organizations to represent that the feature has always been part of a dataset.

## Count Station Book

1. Add a new database connection to ITD's Roads and Highways (R&H) database where the StationBook feature class is maintained.

## Traffic Analysis Zones

1. Contact Data Analytics to request a copy of the TAZ shapefile with fields containing population and area, as well as any other fields of interest to the analyst (e.g., K12 enrollment, income group populations, etc.).
2. Put the shapefile and its supporting files in the processing folder.

## Counties

1. Go to https://www2.census.gov/geo/tiger/TIGER2023/COUNTY/, or the most recent version available from the TIGER repository.
2. Download "tl_2023_us_county.zip".
3. Extract the downloaded file to a folder.
4. Ensure that the data includes a field with the 5-digit State-County FIPS code.
5. Move the shapefile and its associated support files to the processing data folder.
6. Rename the shapefile and its associated support files to "tl_2023_us_county" followed by their respective extensions.

## Employment

Employment data at the census block level comes from the Longitudinal Employer-Household Dynamics (LEHD) Origin-Destination Employment Statistics (LODES) dataset.

Download the Workplace Area Characteristics (WAC) top-level file for Idaho from the latest version of LODES for the most recent year available. The filename will be formatted as "id_wac _S000_JT00_[YEAR].csv.gz" where the components denote state, job group ("S000" is all jobs), and year. The ".gz" file extension indicates that the file should be directly extracted to create a readable ".csv" file. Do ***not*** extract as a new folder.

For the LODES WAC file, …

1. Go to https://lehd.ces.census.gov/data/ (for all LODES datasets) or https://lehd.ces.census.gov/data/lodes/LODES8/id/wac/ (for the LODES8 Idaho Workplace Area Characteristics).
2. From the dropdown menus at https://lehd.ces.census.gov/data/ or the links at https://lehd.ces.census.gov/data/lodes/LODES8/id/wac/, select and download "id_wac_S000_JT00_2022.csv.gz".
    a. The filename is formatted as [State]_[rac/wac]_S000_JT00_[YYYY].csv.gz", indicating the state covered, whether it is a residence or workplace area characteristics, and the year.
    b. "S000" is the file for data covering all employment. Other subsets are available.
    c. Select the most recent option for the "wac_S000_JT00" for the state of interest.
3. Extract the file to the raw inputs data folder.
    a. ".gz" files should be extracted directly to a file, not to a folder.
4. Rename the file to "lehd_wac.csv".

## Census Blocks

For census block geometry, …

1. Go to https://www2.census.gov/geo/tiger/TIGER2023/TABBLOCK20/.
2. Select and download the zip file for ID "tl_2023_16_tabblock20.zip".
    a. Where the year matches the vintage of census blocks indicated in the technical documentation of the LODES version you are using for employment.
    b. The 2-digit code following the year, is the 2-digit FIPS code for the state covered.
3. Unzip the downloaded zip file.

4. Move the files to the raw inputs folder

5. Rename the shapefile and its support files to “tl_2023_16_tabblock20” followed by their extension.

    a. When LEHD’s LODES is next updated and uses new census blocks, the user can either continue to name the files with the name in the step above, or the user can set a new name with the appropriate year, and alter the code in “create_input_data.py” to replace “tl_2023_16_tabblock20.shp” with the new name.

## Statewide Annual VMT Estimate

Request the annual Statewide annual VMT estimate from the Roadway Data Program Manager for the year in which AADT values are being estimated. During the development of this tool, the value for 2023 was 19,700,000,000.

# Toolbox Operating Process

The Rural AADT Estimation Toolbox consists of eight tools, split into three sections: “Data Preparation”, “Spatial Interpolation”, and “Validation”. Data Preparation transforms the input data into a format that supports the underlying interpolation techniques and the calculation of user-inputs for the second section. The Spatial Interpolation tools first take in independent variables (e.g., population density and employment rate) to distribute AADT across off-system roads and then scales those outputs to fit the total expected Vehicle Miles Traveled for that system. Validation compares the AADT values based on actual counts at specific locations to the estimated AADT values on the nearest segment and then reports the difference between the predicted and actual values as well as the overall error statistics.

## Toolbox Installation

To add the toolbox, …

1. Open the “Insert” portion of the ArcGIS Pro ribbon,
2. Click “Toolbox” in the “Project” section,
3. Click “Add Toolbox”,
4. Navigate to the location of the “Off-System AADT Estimation.atbx” toolbox file,
5. Select it, and then
6. Click “Open”.

The toolbox will then be available through the “Catalog” pane under “Toolboxes”.

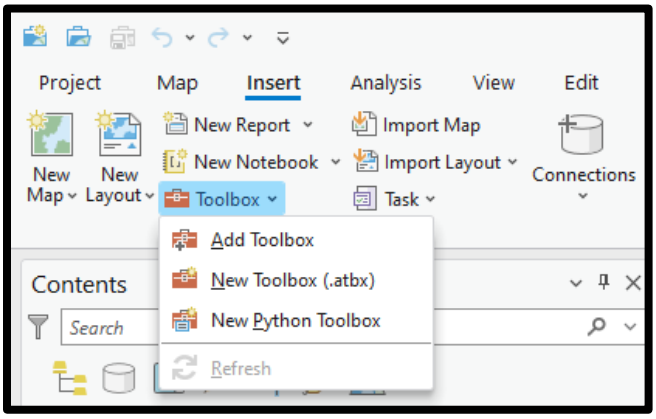


**Figure 4-7. Menu location to add a new toolbox in Esri ArcGIS Pro**

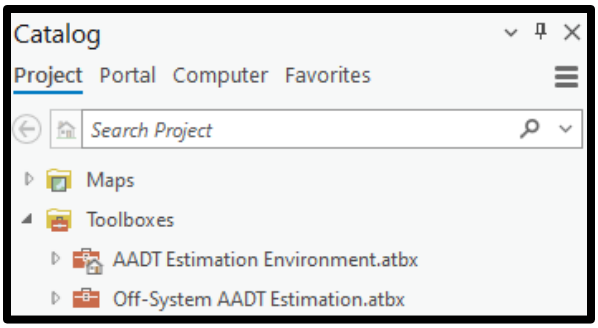


**Figure 4-8. Toolbox location in Esri ArcGIS Pro once added to the project file**

The tools in the toolbox should be run in sequence, starting with Toolbox A: Data Preparation, Tool A1. Create Data Inputs, A2. AADT Count Join, etc., using each tool in the sub-toolbox before moving on to the next sub-toolbox. Tools can be entered to run them by double clicking them in the Catalog pane.

## A. Data Preparation

A. Data Preparation contains five tools: A1. Create Data Inputs, A2. AADT Count Join, A3. Multiple Input Normalized Field Raster, A4. Network Density, and A5. Total VMT from AADT. These tools prepare initial and intermediate data products from some of the user inputs for the second set of tools for spatially interpolating AADT values on off-system roads.

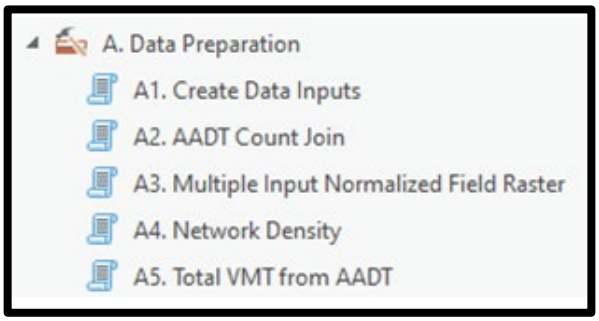


**Figure 4-9. Screenshot of the tools in Toolbox A. Data Preparation**

### A1. Create Data Inputs

Create Data Inputs pulls information from the data in the raw inputs folder as well as from ITD internal databases to create some of the intermediate data products needed for later tools.

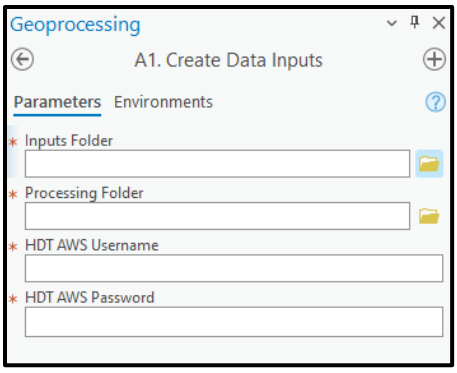


**Figure 4-10. Screenshot of Tool A1. Create Data Inputs parameters**

### Inputs:

- Locations for the "raw inputs" and "processing" folders (see Section 2. Setting Up Your Project)
- The user's HDT AWS Username and Password

### Outputs:

- Overall Road Network feature class titled "RoadNetwork" in the project geodatabase
  (Used in Tools A4. Network Density and B1)
- Federal-Aid System Network feature class titled "FederalAidSystem" in the project geodatabase
  (Used in Tool A5. VMT from AADT)
- Census Blocks shapefile joined with LEHD LODES data titled "lehd_wac_block.shp" in the processing folder
  (Used in Tool A3. Multiple Input Normalization Field Raster)
- A csv file containing AADT estimates from traffic counts titled "StationCounts.csv"
  (Used in Tool A2. AADT Count Join)

### Steps:

1. In the field labeled "Inputs Folder" and "Processing Folder", type in the names of the respective folders within the project geodatabase created as noted in Section 2. Developing Initial Data. This can also be selected by clicking the folder icon to the right of the field, navigating to the input data folder location, and double clicking the folder.
2. Enter your HDT AWS Username and Password in the associated fields.
    a. This portion of the tool will need to be updated both within the toolbox and in the underlying code if the location of the ITD GIS database is altered or the provider changes. See the "create_overaly_network()" function within "create_input_data.py".
3. Click "Run".

4. Once the tool has finished running, verify that there is data in the "lehd_wac_block.shp" file. If there are many missing data points, it is likely that the TIGERLine geodata TABBLOCK used was the incorrect vintage. Verify the correct version with the LEHD LODES Technical Documentation.

## A2. AADT Count Join

The AADT Count Join tool joins the most recent AADT value within the last five years and joins them to valid stations.

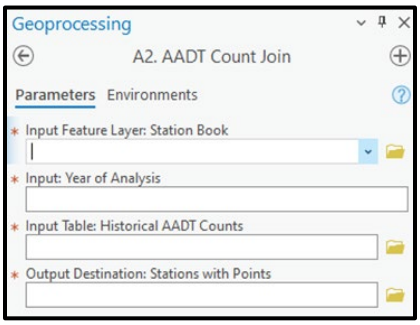


**Figure 4-11. Screenshot of Tool A2. AADT Count Join parameters**

### Inputs:

- The Station Book feature class from Roads and Highways
    - Important: The stations with valid FromDate and ToDate fields should be selected prior to running this tool. Do this by opening the Select By Attribute tool and selecting
        - FromDate that is on or before CURRENT_TIMESTAMP or FromDate is null
        - And
        - ToDate that is after CURRENT_TIMESTAMP or ToDate is null
- Definition Query Example: (FromDate IS NULL OR FromDate <= CURRENT_TIMESTAMP) AND (ToDate IS NULL OR ToDate > CURRENT_TIMESTAMP)The year (YYYY) for which the AADT estimation is being conducted
    - This year should also match the year for which the statewide annual VMT estimate was requested for.
- The historical set of AADT values from traffic counts
  (From "StationCounts.csv" produced in Tool A1. Create Input Data)
- The destination path and name for the output feature layer

### Outputs:

- A feature layer of count stations with the most recent traffic count that occurred in the last five years

(Used in Tool B1. Aggregated AADT Estimation and, if there are counts on off-system roads, Tool C1. AADT Comparison)

Steps:

1. Open the attribute table of the "Stations_With_Counts" feature class created by Tool A1. Create Input Data; it contains the ITD count station points with the ID field "StationID".
    a. In the attribute table, click "Select By Attribute" and use the resulting dialog box to select features based on the FromDate and ToDate fields to ensure that stations that were added after the year of analysis or that were removed before the year of analysis are not included.
        i. FromDate that is on or before CURRENT_TIMESTAMP or FromDate is null
        ii. And
        iii. ToDate that is after CURRENT_TIMESTAMP or ToDate is null
    b. Definition Query Example: (FromDate IS NULL OR FromDate <= CURRENT_TIMESTAMP) AND (ToDate IS NULL OR ToDate > CURRENT_TIMESTAMP)
2. In the field labeled "Input Feature Layer: Station Book", select the feature layer containing the selection from Step 1.
3. In the field labeled "Input: Year of Analysis", enter the year for which this AADT estimation is being done.
4. To the right of the field labeled "Input Table: Historical AADT Counts", click on the folder icon to open a file explorer window. Navigate to the most recent "StationCounts.csv" created by A1. Create Input Data in the processing folder. Select the file and click "OK".
5. In the field labeled "Output Destination: Stations with Points", type in a file path and name (e.g., "C:\Users\Public\Off-System AADT\AADT Estimation Environment.gdb\Stations_With_Counts") for the output location for the generated feature layer of count station points that contains the most recent AADT count as a field in its attribute table. This will be located in the project geodatabase.
6. Click "Run".

## A3. Multiple Input Normalization Field Raster

Multiple Input Normalization Field Raster creates a raster of values from a "primary field" that are normalized by another field from the feature layer. Multiple rasters can be generated from the same or different polygon feature layers in one running of the tool by clicking the "Add another" button below the input field boxes.

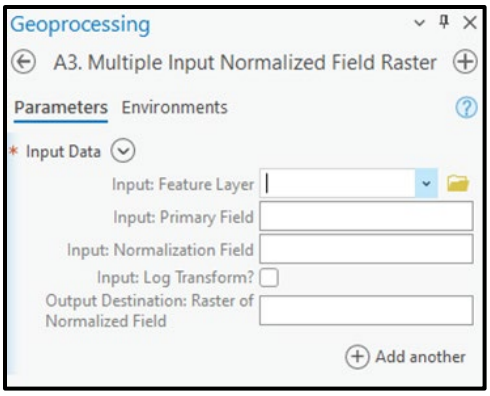


**Figure 4-12. Screenshot of Tool A3. Multiple Input Normalization Field Raster parameters**

From the development of the toolbox, it is recommended that the user develop two rasters for use in Tool B1, see the table after this tool's steps subsection:

- Logged TAZ population density
- Logged LODES employment density

### Inputs:

- A polygon feature layer containing the fields that will produce the output raster
  (TAZs from Section 3 and LODES census blocks from Tool A1)
- The field name of numerator of the output raster value
- The field name of the denominator of the output raster value
- Whether the output raster value should be log transformed
    - This is recommended for independent variables where applying such a transformation creates a distribution more like the normal distribution (population density is one example) but not for cases such as percentages (e.g., percent of jobs that have a monthly income less than $1,250) where a transformation has already been applied.
- The name for the output raster
    - Due to restrictions in file naming for rasters, this tool automatically places the raster inside the current project geodatabase. Therefore, only the name of the raster needs to be provided here, not the entire path.

### Outputs:

- A raster where the cell values are primary field divided by the normalization field, log-transformed if specified by the user
  (Used in Tool B1. Aggregated AADT Estimation)

Steps:

1. In the field labeled “Input: Feature Layer” select the polygon feature layer from your project that you want to create a raster from.

2. In the field labeled “Input: Primary Field”, select a field name from the dropdown that will be the primary value of the output (e.g., the numerator, for example population if you are wanting to create a raster of population density). This field auto-populates from the “Input: Feature Layer”.

3. In the field labeled “Input: Normalization Field”, select a field name from the dropdown that will be the normalizing value of the output (e.g., the denominator, for example geographical area if you are wanting to create a raster of population density). This field auto-populates from the “Input: Feature Layer”.

4. If you want to apply a log transformation to the output values, check the box labeled “Input: Log Transform?”.

5. In the field labeled “Output Name: Raster of Normalized Field”, type in a name (e.g., “TAZ_Population_Density”) for the output location for the generated raster layer that contains the normalized values from the input polygons.

6. If you wish to run create another normalized raster at the same time, click the “Add another” button. This will create a new set of blank fields that can be filled in similarly.

7. Click “Run”.

The current process expects the creation of two rasters via the inputs shown in the table below: population density and employment density. Additional rasters may be created for testing to see if new or updated data sources have a significant impact on model accuracy; however, it should be noted that simply adding more data will not necessarily improve performance.

**Table 4-6. Example of Normalized Raster Inputs**

| Independent Variable | Feature Layer | Primary Field | Normalization Field | Log Transform | Raster Destination |
|---|---|---|---|---|---|
| Population Density | TAZ | TOTPOP_T | TAZ_AREA | Yes | raster_population_density |
| Employment Density | LEHD LODES WAC Census Blocks | C000 (Total Employment field) | Area* | Yes | raster_employment_density |

** An area field may need to be calculated as it is not included in LODES 8.3. For more information, see Esri’s documentation on calculating geometry attributes in new or existing attribute fields. https://pro.arcgis.com/en/pro-app/latest/tool-reference/data-management/calculate-geometry-attributes.htm*

## A4. Network Density

Network Density calculates the total length of a linear feature layer within a user-provided search radius via a raster covering the extent of the feature layer.

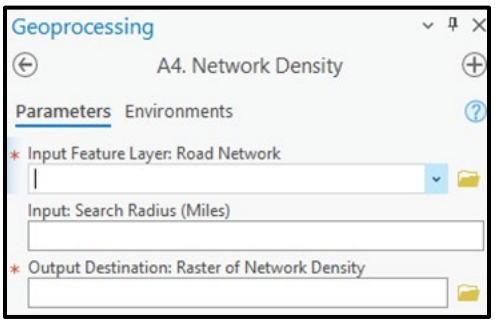


**Figure 4-13. Screenshot of Tool A4. Network Density parameters**

### Inputs:

- The entire road network that is valid for the year of analysis
  (From Tool A1. Create Input Data)
    - This tool should be run three times to generate centerline mile density for…
    - The entire network (features selected that represent the network for the year of analysis, as described in Step 1(a-c) below)
    - Paved roads (features selected that represent the network for the year of analysis, as described in step 1(a-c), below, AND Paved Off-System definition query statement, as shown in Step 2(b)(i) below)
    - Unpaved roads (features selected that represent the network for the year of analysis, as described in Step 1(a-c), below, AND Unpaved Off-System definition query statement, as shown in Step 2(b)(ii) below)
- A search radius in miles to describe the circle within which centerline miles are summed
    - Development of this toolbox used a default value of 5 miles
- The path and name where the output raster will be saved

### Outputs:

- A raster of centerline mile density
  (Used in Tool B1. Aggregated AADT Estimation)

### Steps:

1. Before opening the tool, use the Select By Attribute tool on the road network created in Tool A1. to ensure that the selected features represent the network for the year of analysis, select

a. FromDate that is on or before December 31st of the year of analysis or FromDate is null

b. And

c. ToDate that is after December 31st of the year of analysis or ToDate is null

2. In the field labeled “Input Feature Layer: Road Network”, add the road network you wish to compute the density for.

    a. If you want to create a density raster for a subset of the entire network, select the desired features in the feature layer before running the tool. The tool will only consider selected features.
    You can open the “Select Features by Attribute” tool from either the Geoprocessing pane or from the feature layer’s attribute table. Click the toggle under “SQL” to enter a specific query or use the dialog box features to build a custom query.

    b. SQL Query in “Select Features by Attribute” for

        i. Paved Off-System
        SurfaceType IN (3, 4) Or (SurfaceType IS NULL And (FunctionalClass IN (1, 2, 3, 4, 5, 6) OR FHWAUrbanCode < 99999 AND FHWAUrbanCode IS NOT NULL))

        ii. Unpaved Off-System
        SurfaceType IN (1, 2, 5, 6) Or (SurfaceType IS NULL And (FunctionalClass = 7 Or FunctionalClass IS NULL) And (FHWAUrbanCode = 99999 Or FHWAUrbanCode IS NULL))

    c. If you have an existing selection for the year of analysis, be sure to set the selection to “Refine Current Selection” before running the query.

3. In the field labeled “Input: Search Radius (Miles)”, enter a numeric value for the distance from each segment within which the resulting raster will sum centerline miles. (Testing during tool development used 5 as a default value.)

4. In the field labeled “Output Destination: Raster of Network Density”, type in a file path and name (e.g., “C:\Users\Public\Off-System AADT\AADT Estimation Environment.gdb\Network_Centerline_Mile_Density”) for the output location for the generated raster layer that contains the total centerline miles of the selected features within the search of each cell.

5. Click “Run”.

6. Add the output raster to the ArcGIS Pro project.

## A5. Total VMT from AADT

Total VMT from AADT calculates annual vehicle miles traveled (VMT) on the Federal-Aid System (AADT * Segment Length in Miles * 365) by county. During development, the total value was 16,964,592,140 VMT.

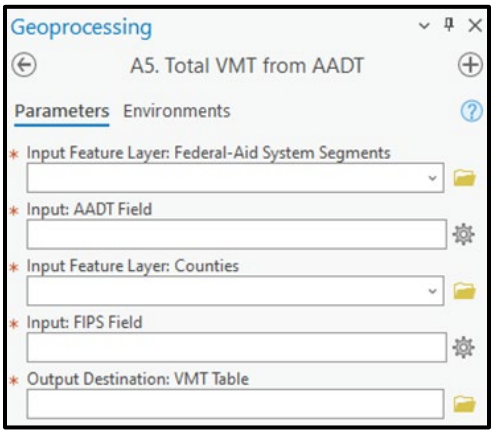


**Figure 4-14. Screenshot of Tool A5. Total VMT from AADT parameters**

### Inputs:

- The Federal-Aid System feature layer that is valid for the year of analysis
  (From Tool A1. Create Input Data)
- The field containing AADT in the Federal-Aid System feature layer
- The county shapefile located in the processing folder (e.g., tl_2023_us_county.shp)
  (From Tool A1. Create Input Data)
- The field containing the 5-digit Federal Information Processing Standards (FIPS) code in the county feature layer
- The path and name for the output table

### Outputs:

- In the "View Details" window of the tool, total annual VMT on the Federal-Aid System is printed as a message
  (Used to calculate an input to Tool B1. Aggregated AADT Estimation)
- A table containing total annual VMT by county

### Steps:

1. Before opening the tool, use the Select By Attribute tool on the Federal-Aid System layer created by Tool A1. Create Input Data. To ensure that the selected features represent the network for the year of analysis, select

a. FromDate that is on or after January 1st of the year of analysis, on or before December 31st of the year of analysis, or FromDate is null

b. And

c. ToDate that is after December 31st of the year of analysis or ToDate is null

2. In the field labeled “Input Feature Layer: Federal-Aid System Segments”, select the Federal-Aid System layer created by Tool A1. Create Data Inputs. It contains the current ITD AADT estimates on the Federal-Aid System.

3. In the field labeled “Input: AADT Field”, select the field name from the dropdown where AADT estimates are stored. The options in the dropdown auto-populate from the “Input Feature Layer: Federal-Aid System Segments” feature layer.

4. In the field labeled “Input Feature Layer: Counties”, select the county shapefile in the processing folder (e.g., tl_2023_us.county.shp).

5. In the field labeled “Input: FIPS Field”, select the field that contains the counties’ 5-digit FIPS codes (“16” for Idaho, “###” for the county). The options in the dropdown auto-populate from the “Input Feature Layer: Counties” feature layer.

6. In the field labeled “Output Destination: VMT Table”, type in a file path and name (e.g., “C:\Users\Public\Off-System AADT\AADT Estimation Environment.gdb\County_VMT_Table”) for the output location for the generated table that contains the total Vehicle Miles Traveled for each County.

7. Click “Run”.

8. Click “View Details”.

9. Record the total VMT value reported in the “Messages” tab. This will be subtracted from the Statewide annual VMT estimate to create an input for Tool B2.

## B. Spatial Interpolation

The Spatial Interpolation portion of the Off-System AADT Estimation toolbox contains the initial modeling and distribution of AADT across the subset of roadways of interest and then scales those initial estimates to a targeted annual VMT value. A subset of off-system roads can be selected before running these tools to produce a more focused estimation. For example, the user could select only the facilities owned and operated by the National Parks Service before running B1 and then provided a reduced amount of annual VMT for the target annual VMT based on prior years' ratio of modeled AADT distributions.

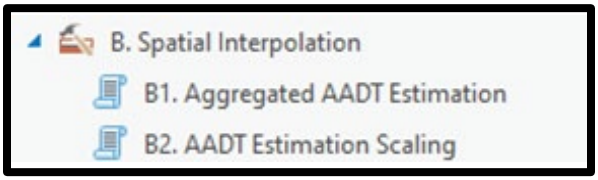


**Figure 4-15. Screenshot of tools in the Toolbox B. Spatial Interpolation**

### B1. Aggregated AADT Estimation

Aggregated AADT Estimation models the relationship between AADTs at Station Book sites with spatial distributions and independent variables to estimate AADT values for each segment.

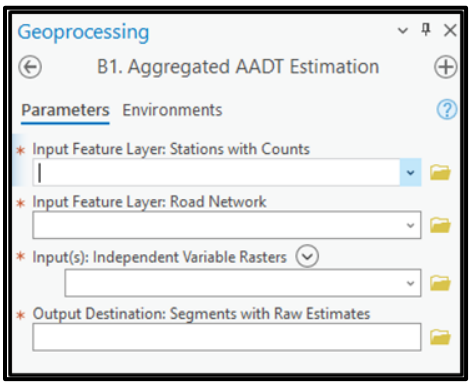


**Figure 4-16. Screenshot of Tool B1. Aggregated AADT Estimation parameters**

The tool uses Empirical Bayesian Kriging Regression (EBK); see Esri's documentation for more information on the functionality and on the principles of Kriging and particularly EBK.

https://pro.arcgis.com/en/pro-app/latest/help/analysis/geostatistical-analyst/what-is-ebk-regression-prediction-.htm

#### Inputs:

- The Station Book with AADT values from traffic counts
  (From Tool A2. AADT Count Join)

- The Road Network for the year of analysis that AADT estimates will be generated for (From Tool A1. Create Input Data)

- Any number of rasters of independent variables

    - The Normalized Field Rasters (logged TAZ population density and logged LODES census block employment density) from Tool A3 and the Network Density Rasters (overall network centerline miles, paved centerline miles, unpaved centerline miles) from Tool A4 were the inputs used during toolbox development.

    - Additional rasters of independent variable can be generated using Tool A3. Multiple Input Normalization Field Raster or Tool A4 Network Density. They can also be generated using the Esri "Polygon to Raster" tool. See Esri's documentation for more details.

- The path and name for the output feature layer that contains the initial AADT estimates

### Outputs:

- A feature layer of segments that includes an initial estimate of AADT (Used in Tool B2)

### Steps:

1. Before opening the tool, use the Select By Attribute tool on the road network created in Tool A1. Create Input Data. To ensure that the selected features represent the network for the year of analysis, select

    a. FromDate that is on or before December 31st of the year of analysis or FromDate is null

    b. And

    c. ToDate that is after December 31st of the year of analysis or ToDate is null

2. In the field labeled "Input Feature Layer: Stations with Counts", select the feature layer in your project from the dropdown that contains the stations with the most recent AADT counts, generated as the output from Tool A2. AADT Count Join.

3. In the field labeled "Input Feature Layer: Road Network", select the feature layer in your project from the dropdown containing the network of interest with fields for Route ID, Functional Class, Local Road Inventory category and Surface Type, and Urban Type, created in Tool A1. Create Input Data.

    a. If you want to create estimates for a subset of the network (e.g., federally owned roads), select that subset of the feature layer before running this tool.

4. In the field labeled "Input(s): Independent Variable Rasters", select a raster layer in your project from the dropdown containing one of the independent variables you wish to use to spatially interpolate AADT. When you add a raster here, ArcGIS Pro will add another field below where

you can add another raster of an independent variable. You can continue this process until you have entered all the rasters you wish to use for this run.

a. During initial testing of the toolbox, the following set of independent variables were identified as having highest predictive capacity that kept the number of independent variables to a minimum. Adding additional rasters increased data preparation and computation time without contributing significant gains in predictive accuracy or statistical relevancy.

    i. TAZ Population Density (Population / Area)

    ii. Census Block LEHD LODES Employment Density (Workers Employed Within the Block / Block Area)

    iii. Overall Network Centerline Mile Density within 5 Miles

    iv. Paved Network Centerline Mile Density within 5 Miles

    v. Unpaved Network Centerline Mile Density within 5 Miles

5. In the field labeled “Output Destination: Segments with Raw Estimates”, type in a file path and name (e.g., “C:\Users\Public\Off-System AADT\AADT Estimation Environment.gdb\Initial_AADT_Estimates”) for the output location for the generated feature layer.

    a. This feature class’ segments match those of the road network segments inputted in the field labeled “Input Feature Layer: Road Network” with estimated AADT values.

6. Click “Run”.

## B2. AADT Estimation Scaling

AADT Estimation Scaling takes the raw estimates produced in Tool B1 and scales the values to match a target total annual VMT. Values are then rounded and summarized in a separate table by county.

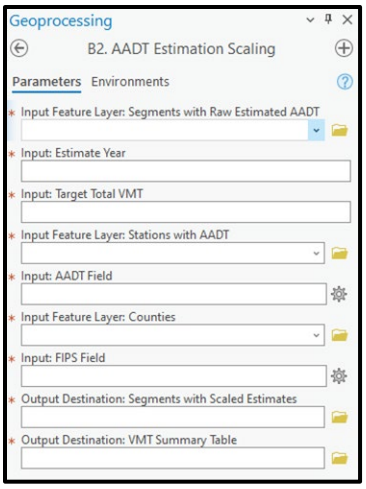


**Figure 4-17. Screenshot of Tool B2. AADT Estimation Scaling parameters**

### Inputs:

- The Initial AADT Estimates linear feature layer with initial AADT estimates
  (From Tool B1. Aggregated AADT Estimation)
- The year (YYYY) for which the AADT estimation is being conducted
    - This year should also match the year for which the statewide annual VMT estimate was requested for.
- The target total VMT for the linear feature layer
    - This is typically the annual Statewide VMT Estimate minus the Federal-Aid System annual VMT generated by Tool A5.
- The Stations With Counts feature layer containing values from traffic counts
  (From Tool A2. AADT Count Join)
- The field containing the stations' AADT values
  (From Tool A2. AADT Count Join)
- The county shapefile located in the processing folder (e.g., tl_2023_us_county.shp)
  (From Tool A1. Create Input Data)
- The field containing the 5-digit FIPS code in the county feature layer
- The path and name for the output feature layer with scaled and rounded AADT estimates
- The path and name for the output table summarizing VMT by county

### Outputs:

- A copy of the feature layer generated by Tool B1. Aggregated AADT Estimation that adds scaled AADT and rounded AADT

    - Unlike AADT values on the Federal-Aid System which have a minimum value of ten, rounding for these segments rounds to the nearest integer when values are below ten.

- A table of the scaled VMT by county where segments are assigned to counties by "largest overlap"

### Steps:

1. In the field labeled "Input Feature Layer: Segments with Raw Estimated AADT", select the Initial AADT Estimates feature layer in your project from the dropdown that contains the estimated AADT values generated in Tool B1. Aggregated AADT Estimation.
2. In the field labeled "Input: Estimate Year", enter the year (YYYY format) for which AADT estimates are being made.
3. In the field labeled "Input: Target Total VMT", enter a numeric value (without commas) for the expected annual Vehicle Miles Traveled on the subset of off-system roads used in B1. Aggregated AADT Estimation. (Testing during tool development used 2,335,407,860 as the VMT value for 2023.)
4. In the field labeled "Input Feature Layer: Counties", select the county shapefile in the processing folder (e.g., tl_2023_us_county.shp) that was prepared by Tool A1. Create Input Data.
5. In the field labeled "Input: FIPS Field", select the field that contains the counties' 5-digit FIPS codes ("16" for Idaho, "###" for the county). The dropdown for this field is auto-populated from the "Input Feature Layer: Counties" feature layer.
6. In the field labeled "Output Destination: Segments with Scaled Estimates", type in a file path and name (e.g., "C:\Users\Public\Off-System AADT\AADT Estimation Environment.gdb\Scaled_AADT_Estimates") for the output location for the generated segments that contain the scaled and rounded AADT values for the segments inputted into the field labeled "Input: Segments with Raw Estimated AADT".
7. In the field labeled "Output Destination: VMT Summary Table", type in a file path and name (e.g., "C:\Users\Public\Off-System AADT\AADT Estimation Environment.gdb\County_Estimated_VMT_Table") for the output location for the generated table that contains the total annual Vehicle Miles Traveled on the segments inputted into the field labeled "Input: Segments with Raw Estimated AADT" for each County.
8. Click "Run".

## C. Validation

The Validation Toolbox contains a single tool for comparing estimated AADT values to the values at count stations: “C1. AADT Comparison”.


**Figure 4-18. Screenshot of the tools in Toolbox C. Validation**

### C1. AADT Comparison

The AADT Comparison tool takes a point feature layer containing count stations with AADT values and compares those AADT values to AADT values on the nearest segment (within a user-defined search radius) that matches its Route ID. Error statistics (Root Mean Squared Error, Mean Absolute Error, and Mean Absolute Percent Error) are calculated for the differences between these values.

The purpose of this tool is to compare the outputs of various model inputs and configurations to determine which option provides the lowest error compared to the real-world counts of traffic volumes. However, at the time of development, ITD had not conducted its own counts on off-system roads.

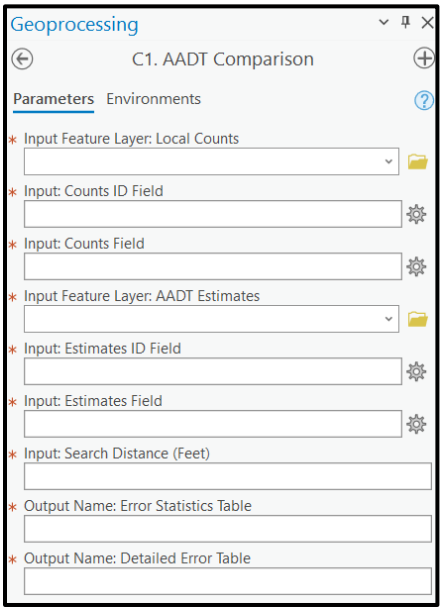


**Figure 4-19. Screenshot of Tool C1. AADT Comparison parameters**

#### Inputs:

- A point feature layer containing AADT values from counts on off-system roads
- The Route ID field from the local counts feature layer

- The field containing AADT values from the local counts feature layer
- The Scaled AADT Estimates linear feature layer with AADT estimates that are being evaluated (From Tool B2. AADT Estimation Scaling)
- The Route ID field from the Scaled AADT Estimates feature layer (From Tool B2. AADT Estimation Scaling)
- The field containing the estimated AADT values from the Scaled AADT Estimates feature layer (From Tool B2. AADT Estimation Scaling)
- A search distance to find the nearest route segment to the local counts feature layer
    - A default value of 5 feet was used during tool development
- The path and name for the table of error values

### Outputs:

- A table of error values that can be used to calculate error metrics
- Error metrics are also reported in the "View Details" window messages

### Steps:

1. In the field labeled "Input Feature Layer: Local Counts", add the feature layer from your project of count stations that contains a Route ID field and an AADT field.
2. In the field labeled "Input: Counts ID Field", select the field name from the dropdown that contains the count station's associated Route IDs. This field auto-populates from the layer selected in "Input Feature Layer: Local Counts".
3. In the field labeled "Input: Counts Field", select the field name from the dropdown that contains the count station most recent AADT value. This field auto-populates from the layer selected in "Input: Local Counts".
4. In the field labeled "Input Feature Layer: AADT Estimates", add the Scaled AADT Estimates feature layer from your project of segments with estimated AADTs that contains a Route ID field and an AADT field. This will typically be the same as the "Network of Interest" from the tools in the B. Spatial Interpolation toolbox.
5. In the field labeled "Input: Estimates ID Field", select the field name from the dropdown that contains the segment's associated Route IDs. This field auto-populates from the layer selected in "Input: AADT Estimates".
6. In the field labeled "Input: Estimates AADT Field", select the field name from the dropdown that contains the most recent AADT estimates (e.g., "Rounded_AADT" from the output of Tool B2. AADT Estimation Scaling). This field auto-populates from the layer selected in "Input: AADT Estimates".

7. In the field labeled “Input: Search Distance (Feet)”, enter a numeric value for the distance in feet from each count station within which the tool will look for a segment with a matching Route ID. (Testing during tool development used 5 as a default value.)
8. In the field labeled “Output Destination: Error Statistics Table”, type in a file path and name (e.g., “C:\Users\Public\Off-System AADT\AADT Estimation Environment.gdb\AADT_Error_Statistics”) for the output location for the generated table that contains the calculated error statistics for the segments inputted into the field labeled “Input: AADT Estimates”. (These values are also displayed in the “Messages” found in the tool details.)
9. Click “Run”.

## Excluding Existing Off-System AADT Values

There may already be traffic counts on off-system routes. If that is the case and the analyst wishes to exclude those facilities from the estimation process and to remove their associated annual VMT from the scaling process of other off-system roads, consider the following process and questions.

1. Estimate AADT for all off-system roads.
2. Using Tool C1, compare differences between those estimates and AADT values based on local counts on off-system roads.
    a. How many are significantly different?
    b. How big are these differences? Overall and by individual facility?
3. Calculate annual off-system VMT from roads with AADTs based on local counts
    a. What percentage of the annual Statewide VMT estimate is this value?
4. Subtract that total from the “target VMT” input used in Tool B1.
5. Select off-system roads that ***do not*** have an AADT value based on traffic counts.
6. Run Tool B1. Aggregated AADT Estimation on the selected facilities.
7. Run Tool B2. AADT Estimation Scaling with the new target value.

## File Metadata

- A1 Output (StationCounts.csv):
    - STATION_BOOK_ID: Identifier for unique count stations
    - COUNT_DATE: Date the count was taken
    - AADT_COUNT: AADT value associated the count's identifier and date
- A2 Output (Stations with Counts):
    - Adds "AADT_COUNT" and "AADT_LOG" fields to the supplied Station Book feature layer
    - AADT_Count: Most recent value within five years for the station
    - AADT_LOG: The natural log of AADT_Count
- A3 Output (Normalized Field Rasters):
    - E.g., the population density raster and employment density raster
    - Band1: The normalized field value calculated by the tool
- A4 Output (Network Density Rasters):
    - E.g., the network centerline mile density, paved centerline mile density, and unpaved centerline mile density
    - Band1: The total length in miles of selected features within the search radius
- A5 Output (County VMT Table):
    - Route ID: The route identifier of the segment
    - FromMeasure / ToMeasure: The starting and ending measure of the segment
    - [AADT_Field]: The user-specified field for AADT values
    - [GEOID_Field]: The user-specified field for county 5-digit FIPS codes
    - VMT: Annual Vehicle Miles Traveled on the segment in the county
- B1 and B2 Outputs
    - VMT Summary Table
        - FIPS Field: 5-digit FIPS code for the county, exact name will depend on the file inputted into Tool A1. Create Data Inputs. Examples include "ST_CNTY_CODE", "GEOID", GEOID20", etc.

- FREQUENCY: Number of segments within the county
- SUM_Rescaled_VMT: The total annual VMT within the county on the off-system roads that estimates were created for

- Segments (Initial_AADT_Estimates and Scaled_AADT_Estimates)
  - Route ID: The route identifier of the segment
  - FromMeasure / ToMeasure: The starting and ending measure of the segment
  - FacilityType: The operational characteristic of the roadway (segment)
  - FunctionalClass: Represents a specific classification of roadway (segment)
  - SurfaceType: The surface type of a given section of roadway (segment)
  - FHWAUrbanCode: The U.S. Census Urban Area Code
  - FIPS Field: 5-digit FIPS code for the county, exact name will depend on the file inputted into Tool A1. Create Data Inputs. Examples include “ST_CNTY_CODE”, “GEOID”, GEOID20”, etc.
  - RASTERVALU: Median value extracted from the raster of predictions generated by the EBK Regression process. Units are log-transformed as the AADT values supplied to the process are log-transformed.
  - EXP_AADT: The exponentialized RASTERVALU to return the units to actual AADT
  - Rescaled_AADT: EXP_AADTs that have been scaled to ensure they combine with segment lengths to produce annual Vehicle Miles Traveled
  - Rescaled_VMT: Rescaled_AADT multiplied by the segment length (in miles) multiplied by 365
  - Rounded_AADT: Rescaled_AADTs that have been rounded according to ITD’s traffic rounding process, with the exception that values less than 10 are rounded to the nearest integer.
  - Year: The estimation year inputted by the user

# Toolbox Update Framework

The toolbox can be updated in one of three ways: updated information within user inputs; new parameters or independent variables for the input development, data preparation, or spatial interpolation processes; or modifications to the underlying code processes.

## Updated User Inputs

Examples of user inputs that will be updated regularly include:

- New AADT counts
- Census bureau socioeconomic information
- TAZ socioeconomic information
- Roadway geometry and characteristics (e.g., Functional Class, Surface Type, Urban Type, etc.)

## New Parameters or Independent Variables

New parameters or independent variables can be provided by the user in the following locations

- A2: Selecting a source feature layer, what field will be rasterized, what field will be used for normalization, whether a log-transform will be applied
- A3: What subset of the network is selected to calculate density
- B1: What network of interest are estimates being created for and what independent variable rasters are provided for the EBK spatial interpolation process
- B2: What portion of the off-system annual VMT is used as the target VMT

## Underlying Code Modifications

The underlying Python code of the tools can be modified; however, it is recommended that this only be done by developers familiar with both Python and its integration into the Esri software suite. Direct modification of the code is required to change aspects such as:

- The dynamic cutoff date for “recent” AADT counts
- Rounding rules for scaled AADT estimates
- The default cell size for rasters
- The search radius for proximity filters

# 5. Project User Guide

This guide provides detailed documentation of the Off-System AADT Estimation Toolbox's underlying components and processes involved in generating, processing, and analyzing AADT (Annual Average Daily Traffic) estimates, as well as related metrics such as Vehicle Miles Traveled (VMT) and error evaluations. Unlike the separate user training guide—designed to walk end users through installation and operational procedures, this guide is aimed at ITD staff and future developers. It is intended to facilitate record keeping, maintenance, and modification of the Toolbox as project needs evolve.

The guide is organized into sections that correspond to each key script used in the Toolbox. Each section describes the inputs, functionality, processing logic, outputs, and considerations for maintenance and future modifications of a specific script. Topics covered include:

- Input Data Preparation: Creating and managing spatial and tabular datasets that serve as the foundation for traffic analysis (e.g., joining LEHD data to census blocks).
- AADT and Count Data Integration: Merging historical AADT counts with count station data and processing these for further analysis.
- Raster Processing and Normalization: Generating normalized raster datasets from polygon features based on field values, including optional logarithmic transformations.
- Road Network Analysis: Calculating road network density and estimating VMT on Federal-Aid roadways.
- AADT Estimation and Scaling: Estimating AADT on non-Federal-Aid roadways using Empirical Bayesian Kriging, and then scaling these estimates to match statewide VMT targets.
- Error Analysis: Comparing modeled AADT estimates against real-world counts and computing key error metrics (RMSE, MAE, MAPE) to assess model performance.

By documenting each component in detail, this guide ensures that ITD staff have a comprehensive understanding of the Toolbox's internal workings. This, in turn, supports efficient troubleshooting, system updates, and future enhancements to the analytical framework.

# A1. create_input_data.py

This script is responsible for preparing several input datasets needed in the AADT (Annual Average Daily Traffic) estimation process. It accesses ITD databases, user-provided files, and spatial datasets to generate processed outputs (spatial files, feature classes, and CSV reports) used by subsequent analysis tools.

## Inputs and Parameters

### External Dependencies

- **Python Libraries:**
  - `requests`
  - `geopandas`
  - `pandas`
  - `arcpy` (ArcGIS Python package)
  - `oracledb` (for Oracle database connectivity)
- **Other Modules:**
  - The script uses file system operations (e.g., `os.path.join`), so ensure the `os` module is imported or available in the environment.

### Runtime Parameters (supplied via ArcGIS)

When executed, the script retrieves five key parameters using `arcpy.GetParameterAsText`:

1. **input_dir:**
   - Directory containing user-provided datasets, e.g., `lehd_wac.csv` and the census block shapefile (`tl_2023_16_tabblock20.shp`).
2. **processing_dir:**
   - Working directory where processed outputs (e.g., intermediate shapefiles, CSV files) are written.
3. **username:**
   - Database username for connecting to the ITD Oracle database.
4. **password:**
   - Corresponding password for the ITD Oracle connection.

### External Data Sources & Database Connections

- **ITD GIS Databases:**
  The script contains hard-coded SDE connection strings for various feature layers (e.g., AADT counts, FunctionalClass, FacilityType, etc.). Future changes in the ITD GIS database locations will require updating these paths.
- **Oracle Database:**
  The script connects to the ITD database using the DSN `"HDTPRODAWS"` to query station count data.

## Detailed Functionality and Processes

### LEHD_WAC_Block Function

- **Inputs:**
  - LEHD LODES CSV file (`lehd_wac.csv`) from `input_dir`
  - Census block shapefile (`tl_2023_16_tabblock20.shp`) from `input_dir`
- **Processing Steps:**
  - Reads and prepares tabular data (sets `w_geocode` as string and index).
  - Reads the spatial census block data, converts the GEOID field to string, and sets it as index.
  - Joins the tabular LEHD data with the spatial census blocks.
  - Reprojects the joined data to the EPSG:8826 coordinate reference system.
  - Calculates an "Area_MI" attribute (area in square miles) from the geometry.
- **Outputs:**
  - Writes a new shapefile (`lehd_wac_block.shp`) to the `processing_dir`.

### create_overlay_network Function

- **Inputs:**
  - Uses ArcGIS project connection (`"CURRENT"`) and several ITD GIS feature layers via SDE connection strings.
- **Processing Steps:**
  - Creates individual feature layers for various road network components (AADT, FunctionalClass, FacilityType, UrbanLimits, ThroughLanes, LocalRoadInventory).
  - Creates a road network layer.
  - Uses `arcpy.locref.OverlayEvents` to overlay these feature layers into a single in-memory composite layer.
  - Performs field renaming for convenience after the overlay.
  - Exports two separate feature classes:
    - **RoadNetwork:** Filtered by an active records query based on date fields.
    - **FederalAidSystem:** Filtered using a query that selects appropriate FacilityType and FunctionalClass records.
- **Outputs:**
  - Two feature classes added to the ArcGIS project's default geodatabase:
    - `"RoadNetwork"`
    - `"FederalAidSystem"`

### create_counts_csv Function

- **Inputs:**
  - Connects to the ITD Oracle database using provided credentials.
  - Executes an SQL query on the `pdb_stat` and `pdb_site` tables to retrieve station count data (filtering records post-2015 and excluding placeholder values).
- **Processing Steps:**
  - Executes the SQL query and fetches all rows.

    - Converts the query results into a pandas DataFrame.
    - Assigns meaningful column names: `"STATION_BOOK_ID"`, `"COUNT_DATE"`, and `"AADT_COUNT"`.
- **Outputs:**
    - Writes a CSV file (`StationCounts.csv`) to the `processing_dir`.

### project_counties Function

- **Inputs:**
    - Reads a counties shapefile (`tl_2023_us_county.shp`) from the `processing_dir`.
- **Processing Steps:**
    - Reprojects the counties shapefile to the EPSG:8826 coordinate system.
- **Outputs:**
    - Overwrites the original counties shapefile in the `processing_dir` with the reprojected version.

### run Function and Execution Flow

- **Function Role:**
  The `run()` function orchestrates the overall execution by calling the above functions sequentially.
- **Execution Order:**
    1. **create_overlay_network()**
    2. **LEHD_WAC_Block()**
    3. **create_counts_csv()**
    4. **project_counties()**
- **Logging:**
  The script uses `arcpy.AddMessage` to log progress messages throughout execution, aiding in debugging and user feedback.

## Outputs Summary

- **Spatial Datasets:**
    - `lehd_wac_block.shp` (Joined LEHD and census block data)
    - Feature classes in the geodatabase:
        - `RoadNetwork`
        - `FederalAidSystem`
    - Reprojected counties shapefile (`tl_2023_us_county.shp`)
- **Tabular Data:**
    - `StationCounts.csv` (CSV file containing station counts and dates)

## Considerations for Maintenance and Future Modifications

- **Database and File Path Changes:**
  If the ITD GIS database locations or filenames change:
    - Update the SDE connection strings in `create_overlay_network()`.

    - Adjust the file paths (in `input_dir` and `processing_dir`) for the LEHD data, census blocks, and counties shapefiles.
- **Schema Changes:**
  Changes to field names or data structure in either the input datasets or database tables might necessitate modifications to:
    - The join operations in `LEHD_WAC_Block()`
    - The overlay and field renaming process in `create_overlay_network()`
    - The SQL query in `create_counts_csv()`
- **Error Handling & Logging:**
  Currently, error handling is minimal. Developers should consider implementing additional try/except blocks around critical operations (e.g., file I/O, database connections) and enhancing logging for better traceability.
- **Environment Dependencies:**
  This script is designed to run within an ArcGIS environment with access to `arcpy`. Ensure that all required Python libraries are installed and that the script is executed in an environment with proper GIS licensing and configurations.
- **Extensibility:**
  Future enhancements (e.g., additional data processing steps or new output formats) should follow the modular function structure. New functions can be added and then invoked in the `run()` function.

## Summary

This document section provides an overview of the script's components and processes, ensuring that ITD staff can understand how each function contributes to the overall AADT estimation process. It outlines the inputs, processing steps, and outputs, as well as considerations for modifications and maintenance. Future developers can use this guide to troubleshoot issues, update processing logic, or extend the functionality as required by evolving project needs.

# A2. aadt_count_join.py

This script is designed to join the most recent AADT (Annual Average Daily Traffic) count from historical records (within the last five years) to its corresponding count station. The joined dataset is then processed further by calculating the logarithm of the AADT count to support subsequent analyses.

## Inputs and Parameters

### External Dependencies

- **Python Modules:**
  - `arcpy` – ArcGIS Python package for spatial data management and processing.
  - `sys` (specifically `argv`) – for handling command line arguments (if needed).
- **Note:**
  The script relies on an ArcGIS environment and uses its geoprocessing tools.

### Runtime Parameters

When executed, the script accepts the following parameters using `arcpy.GetParameter` functions:

1. **Station_Book:**
   - A spatial dataset representing geolocated count stations.
2. **Year:**
   - A string or numeric value indicating the reference year. This parameter is used in filtering active events and setting a threshold date (December 31 of the specified year).
3. **AADT_Counts:**
   - A tabular dataset containing historical AADT counts. This data is assumed to have been generated by an earlier process (e.g., from the Create Input Data script).
4. **Stations_with_Counts (Out Points):**
   - The output destination where the joined spatial dataset will be written.

## Detailed Functionality and Processes

### Data Preparation

- **Filtering Active Stations:**
  The script starts by selecting active count stations from the Station_Book dataset. An SQL query filters stations based on their FromDate and ToDate fields using the provided Year as the reference date.
- **Creating a Working Copy:**
  A temporary copy of the filtered station data is created, which will later be updated with the latest AADT counts.
- **Setting Up In-Memory Tables:**
  An in-memory table named "Latest_AADT_Counts" is created with fields to store:
  - `STATION_BOOK_ID` (the station identifier)
  - `COUNT_DATE` (the date of the count)

- `AADT_COUNT` (the traffic count)

### Data Processing

- **Establishing a Time Threshold:**
  The script calculates the threshold year for the last five years. Using the user-defined year of analysis, it subtracts five years to define the cutoff for which counts are considered recent.
- **Filtering Historical AADT Counts:**
  A temporary table ("Filtered_AADT_Counts") is generated by copying rows from the input AADT_Counts table. An update cursor then deletes rows where the year of the count (from the "YEAR" field) is older than the computed five-year threshold or more recent than the user-defined year of analysis.
- **Extracting the Latest Count:**
  A dictionary is built to store the most recent AADT count per STATION_BOOK_ID. A search cursor iterates over the filtered table (ordered by station and count date in descending order) and captures only the first (latest) record for each station.
- **Populating the In-Memory Table:**
  The latest count for each station is inserted into the "Latest_AADT_Counts" table using an insert cursor.

### Joining and Post-Processing

- **Joining Datasets:**
  The script joins the latest counts from the in-memory table to the working copy of the count stations (Stations_with_Counts) using the matching fields (StationID in the stations and STATION_BOOK_ID in the counts table).
- **Data Cleanup:**
  After the join, any station that does not have an associated AADT_COUNT is removed from the output dataset to ensure data consistency.
- **Calculating Derived Field:**
  A new field, AADT_LOG, is added to the joined dataset. This field is populated by computing the natural logarithm of the AADT_COUNT values, which may be used in further statistical or spatial analysis.

## Outputs Summary

- **Joined Spatial Dataset:**
  The final output is a spatial dataset (written to the user-specified destination) that includes:
  - The original station attributes.
  - The most recent AADT count (filtered for counts within the last five years).
  - A derived field (`AADT_LOG`) containing the logarithm of the AADT count.

## Considerations for Maintenance and Future Modifications

- **Time-Based Filtering:**

    - The script currently calculates the five-year threshold based on the system's current date. If a different time window is required, adjust the calculation accordingly.
- **Field and Schema Updates:**
    - If there are changes in the schema of the input datasets (e.g., field names or data types in `AADT_Counts` or `Station_Book`), modifications in the cursor operations and join parameters may be needed.
- **Error Handling:**
    - Consider implementing additional error handling (e.g., try/except blocks) around critical operations such as database connections, file I/O, or arithmetic operations (like logarithm calculations) to improve robustness.
- **Performance Enhancements:**
    - As the dataset grows, review the efficiency of in-memory operations and cursor-based processing. Indexing fields or using more efficient SQL queries may improve performance.
- **Dependency Updates:**
    - Ensure that the script's dependencies (e.g., arcpy and its associated ArcGIS environment) are kept up to date and are compatible with any changes in the project infrastructure.

## Summary

This script automates the process of joining the most recent AADT counts to their corresponding count stations. By filtering historical data for counts within the last five years, creating an in-memory table for efficient processing, and performing a join operation with subsequent cleanup and calculation of a logarithmic field, the script ensures that downstream processes work with current and properly formatted data. Future developers and ITD staff can use this documentation to troubleshoot, maintain, or extend the functionality as needed.

# A3. Geo_field_norm_raster_multiple.py

This script generates raster datasets from polygon feature layers by normalizing values from specified fields. The normalization is performed by dividing a primary field (numerator) by a normalizing field (denominator). Optionally, a logarithmic transformation is applied to the normalized values prior to rasterization. The output raster is then exported to the project's geodatabase with a sanitized name that complies with geodatabase naming conventions.

## Inputs and Parameters

### External Dependencies

- **Python Modules:**
  - `arcpy` – For geospatial processing within an ArcGIS environment.
  - `re` – For sanitizing raster names.
- **Execution Environment:**
  The script is designed to run within an ArcGIS project, requiring access to an active ArcGIS session and its default geodatabase.

### Runtime Parameters

- **ValueTable Parameter:**
  The script retrieves a ValueTable that contains one or more rows of parameters. Each row must include:
  1. **Geos (Polygon Feature Layer):**
     - The source polygon dataset from which the raster is derived.
  2. **Primary_Field_Name:**
     - The field containing the primary data (e.g., population or another numerator value).
  3. **Norm_Field_Name:**
     - The field used for normalization (e.g., area).
  4. **Log_Transform:**
     - A boolean flag that indicates whether to apply a logarithmic transformation to the normalized values.
  5. **Raster_Geo_Field_Normalized:**
     - The desired output name for the raster dataset.

## Detailed Functionality and Processes

### Name Sanitization

- **Function:** `sanitize_name(name)`
- **Process:**
  - Replaces invalid characters in the raster name with underscores.
  - Ensures that the name does not begin with a number or an underscore.

- Truncates the name to a maximum of 50 characters to comply with geodatabase naming restrictions.

### Raster Creation and Normalization

- **Function:** GeoFieldNorm(Geos, Primary_Field_Name, Norm_Field_Name, Log_Transform, Raster_Geo_Field_Normalized)
- **Data Preparation:**
  - **Layer Copy:**
    The input polygon features are copied to an in-memory layer to preserve the original data.
  - **Field Addition and Calculation:**
    A new field, `density`, is added to calculate the ratio of the primary field to the normalization field.
- **Cell Size Determination:**
  - The script retrieves the spatial reference of the input layer to determine its linear unit.
  - A corresponding cell size is set based on the linear unit (e.g., 1000 meters for "Meter", or appropriate conversions for "Foot", "Kilometer", "Mile", or degrees).
  - A warning is issued and a default cell size is used if the unit is unrecognized.
- **Optional Logarithmic Transformation:**
  - If `Log_Transform` is enabled, a new field `density_log` is added.
  - The logarithm of the `density` field is calculated, substituting with the minimum positive density value when necessary to handle zero or negative values.
  - The log-transformed field is then used for rasterization.
- **Rasterization:**
  - The processed polygon layer is converted to a raster dataset using `arcpy.conversion.PolygonToRaster`.
  - The cell size is applied (multiplied by 5 to adjust resolution as needed).

### Raster Export

- **Export Process:**
  - The script retrieves the current ArcGIS project's default geodatabase.
  - The desired output raster name is sanitized using the `sanitize_name` function.
  - The in-memory raster is exported to the geodatabase using `arcpy.management.CopyRaster`.
  - A success message is logged with the export path.

### Iterating Over Multiple Inputs

- **Function:** `main()`
- **Process:**
  - Retrieves the ValueTable parameter containing multiple sets of inputs.
  - Iterates over each row, logging details such as the input polygon layer, fields for normalization, log transformation flag, and output raster name.

- Calls `GeoFieldNorm` for each dataset, processing and exporting each raster accordingly.
- Provides status messages for each iteration to indicate success or failure.

## Outputs Summary

- **Raster Datasets:**
  - For each row in the ValueTable, a raster dataset is created representing the normalized (and optionally log-transformed) field values.
  - The output rasters are stored in the project's default geodatabase with names sanitized to conform to geodatabase naming standards.
- **Logging Information:**
  - The script outputs detailed messages at various stages (e.g., feature copying, field calculations, rasterization, and export) to assist with debugging and process verification.

## Considerations for Maintenance and Future Modifications

- **Input Validation:**
  - Verify that the input polygon features contain the necessary fields (`Primary_Field_Name` and `Norm_Field_Name`) with valid numerical data.
  - Consider additional error handling for scenarios with zero or negative values that may affect the logarithmic transformation.
- **Cell Size and Resolution:**
  - The cell size is determined based on the spatial reference unit with a fixed multiplication factor. Adjust this factor as needed based on the specific analysis or dataset resolution requirements.
- **Name Sanitization Updates:**
  - Periodically review the `sanitize_name` function to ensure it remains effective with any changes to geodatabase naming restrictions.
- **Performance Considerations:**
  - For large datasets, evaluate the efficiency of in-memory processing and consider optimizations or batch processing to improve performance.
- **ArcGIS and Dependency Compatibility:**
  - Ensure that the script remains compatible with updates to ArcGIS and arcpy, especially changes in spatial reference handling and raster processing tools.

## Summary

This script automates the creation of normalized rasters from polygon datasets by computing field ratios and optionally applying a log transformation. Through its modular design—comprising input sanitization, normalization processing, and iterative raster creation, the script ensures that output rasters are consistent, properly named, and stored within the project's geodatabase. This documentation is intended to aid ITD staff and future developers in maintaining, troubleshooting, and extending the functionality of this component of the Toolbox.

# A4. network_density.py

This script calculates the density of a road network's centerline miles by creating a raster where each cell's value represents the cumulative centerline length (density) of the road features within a specified search radius. This tool is useful for analyzing spatial patterns in road networks.

## Inputs and Parameters

### External Dependencies

- **Python Modules:**
  - `arcpy` – Core module for geospatial processing.
  - `arcpy.sa` – Spatial Analyst extension used for raster calculations.
  - `sys` – For handling command-line arguments if needed.
- **ArcGIS Environment:**
  The script must run within an ArcGIS environment where the Spatial Analyst extension is available and licensed.

### Runtime Parameters

1. **Complete_Road_Network:**
   - A spatial dataset (feature class) representing the entire road network whose centerline density is to be calculated.
2. **Search_Radius_Raw:**
   - A user-specified search radius (in miles) used to determine the area over which road network lengths are aggregated.
   - If not provided, a default value of 5 miles is used.
3. **Density_Raster:**
   - The output destination (raster dataset) where the computed density raster will be saved.

## Detailed Functionality and Processes

### Environment Setup and Data Preparation

- **Extension Licensing:**
  The script begins by checking out the Spatial Analyst extension using `arcpy.CheckOutExtension("spatial")` to ensure that all necessary tools are available.
- **Copying the Input Data:**
  The complete road network is copied to an in-memory feature class (`Copied_Network`) to protect the original data and optimize processing.

### Cell Size and Search Radius Calculation

- **Determining Cell Size:**
  The script retrieves the spatial reference of the input road network and examines its linear unit. It then sets a cell size based on common units:
  - **Meter:** 1 mile = 1609.34 meters.
  - **Foot:** 1 mile = 5280 feet.
  - **Kilometer:** 1 mile ≈ 1.60934 kilometers.
  - **Mile:** 1 mile = 1.
  - **Degree/Decimal Degree:** Uses a conversion factor (0.0145) as an approximation.
  - **Default:** Issues a warning and falls back to 1609.34 meters if the unit is unrecognized.
- **Adjusting the Search Radius:**
  The provided `Search_Radius_Raw` is converted to an integer and multiplied by the calculated cell size to determine the actual search radius in the appropriate unit.

### Density Calculation and Raster Generation

- **Raster Density Calculation:**
  The script uses the `arcpy.sa.LineDensity` function to compute a density raster based on the copied road network. The function calculates the total centerline length per cell within the defined search radius.
  - The output density is expressed in "SQUARE_MILES."
- **Saving the Raster:**
  The resulting raster is saved to the user-specified output location (`Density_Raster`).

## Outputs Summary

- **Density Raster Dataset:**
  A raster dataset is generated where each cell value represents the density (centerline length per unit area) of the road network within the defined search radius. This raster is saved at the location specified by the user.

## Considerations for Maintenance and Future Modifications

- **Spatial Analyst Extension:**
  Ensure that the Spatial Analyst extension is available and properly licensed before executing the script.
- **Input Data and Units:**
  - Verify that the input road network has a correctly defined spatial reference with an identifiable linear unit.
  - Adjust the cell size logic if additional or custom linear units are introduced.
- **Search Radius Handling:**
  The script currently defaults to a 5-mile search radius if none is provided. Future changes to analysis requirements might necessitate a parameter for dynamic radius adjustments.

- **Performance:**
  Since the script uses in-memory processing, monitor resource usage with large datasets. Consider performance tuning or batch processing for extensive road networks.
- **Error Handling:**
  The current script has minimal error handling. Future enhancements might include more robust try/except blocks to capture and log potential issues during processing.

## Summary

This script calculates the density of a road network's centerline miles by leveraging the ArcGIS Spatial Analyst tools. It processes an input road network feature class, computes cell size based on the network's spatial reference, and aggregates the road lengths within a user-defined search radius to produce a density raster. This documentation serves as a reference for ITD staff and future developers to maintain, troubleshoot, or extend the functionality of the density calculation tool.

# A5. vmt_from_aadt.py

This script calculates the total annual Vehicle Miles Traveled (VMT) on the Federal-Aid System (FAS) by using existing AADT estimates on roadway segments. It computes the annual VMT per segment by multiplying the AADT value by the segment's length (in miles) and by the number of days in a year. The statewide annual VMT for Federal-Aid roadways is summarized and then joined to county-level data to produce a CSV table of annual VMT totals by county. This information is subsequently used to adjust AADT scaling on off-system facilities.

## Inputs and Parameters

### External Dependencies

- **Python Modules:**
  - `arcpy` – For geospatial processing.
  - `math` – For mathematical operations.
  - `arcpy.sa` – Spatial Analyst module for advanced raster operations.
  - `sys` – For command-line argument handling.
- **Execution Environment:**
  The script is intended to run within an ArcGIS environment with access to the necessary geoprocessing tools and licenses.

### Runtime Parameters

1. **FAS_Segments:**
   - A feature class representing Federal-Aid roadway segments with AADT estimates.
2. **AADT_Field:**
   - The field name in the FAS_Segments layer that contains the AADT values.
3. **Counties:**
   - A feature class containing Idaho TIGERLine county boundaries.
4. **GEOID_Field:**
   - The field in the Counties dataset that holds the 5-digit State-County FIPS codes.
5. **VMT_Table:**
   - The output destination (CSV file) where the aggregated annual VMT values by county will be written.

## Detailed Functionality and Processes

### Data Preparation and Layer Setup

- **Overwrite Management:**
  The script begins by setting `arcpy.env.overwriteOutput = True` to allow output replacement.
- **Temporary Layers:**
  Temporary layers are created for the AADT segments (`AADT_Layer`), Idaho counties

(`County_Layer`), and the joined dataset (`Joined_Layer`). Any existing layers or output files with these names are deleted to prevent conflicts.

- **County Filtering:**
  The Counties layer is filtered using a SQL where clause (`SUBSTRING({GEOID_Field}, 1, 2) = '16'`) to include only Idaho counties.

### VMT Calculation on Federal-Aid Segments

- **Segment Length Calculation:**
  The script calculates the length of each roadway segment in miles using geodesic measurements.
- **VMT Field Addition and Calculation:**
  - A new field, `VMT`, is added to store the VMT for each segment.
  - VMT is calculated as: VMT=(AADT value or 0)×(Segment Length or 0)×365.0\text{VMT} = (\text{AADT value} \text{ or } 0) \times (\text{Segment Length} \text{ or } 0) \times 365.0 This estimates annual VMT per segment.
- **Statewide VMT Summation:**
  A search cursor iterates over the segments to accumulate the total Federal-Aid VMT.
  - The script checks for valid numeric values and logs warnings for any invalid entries.
  - The statewide annual VMT is logged as a message for verification.

### Spatial Join and Data Consolidation

- **Spatial Join:**
  A spatial join is performed between the AADT layer and the filtered Counties layer. This assigns county FIPS codes (from the Counties dataset) to the roadway segments based on the largest area of overlap.
- **Field Management:**
  After the join, the script retains only key fields such as `OBJECTIVEID`, `RouteID`, `FromMeasure`, `ToMeasure`, the AADT field, the county FIPS field (`GEOID_Field`), and the calculated `VMT`. Unnecessary fields are deleted to streamline the output.

### Exporting the Summary Table

- **Table Conversion:**
  The joined feature class is converted to a table stored in memory.
- **CSV Export:**
  The in-memory table is then exported to a CSV file (specified by the `VMT_Table` parameter), summarizing the annual VMT totals by county FIPS code.

## Outputs Summary

- **CSV Table (VMT_Table):**
  A CSV file is generated that contains aggregated annual VMT totals for Federal-Aid roadways, organized by county (using the FIPS codes). This output provides a key input for subsequent AADT scaling on off-system facilities.

- **Log Messages:**
  The script logs progress messages, including the calculated statewide Federal-Aid annual VMT, which aids in verification and troubleshooting.

## Considerations for Maintenance and Future Modifications

- **Data Integrity:**
  - Ensure that the FAS_Segments layer contains valid AADT values and accurately measured segment lengths.
  - Confirm that the Counties dataset has correct and up-to-date FIPS codes.
- **Field and Schema Adjustments:**
  Changes in field names or data structure (in either the segments or counties datasets) will require updates to the SQL queries and field management sections.
- **Error Handling and Logging:**
  The current implementation includes basic warnings and messages. Consider enhancing error handling (e.g., try/except blocks) to capture and respond to exceptions more robustly.
- **Performance Considerations:**
  For larger datasets, review the performance of the spatial join and cursor-based annual VMT summation. Optimizations or batch processing may be warranted if processing times become excessive.
- **Licensing:**
  Verify that the Spatial Analyst extension is available and licensed, as it is essential for some of the processing steps.

## Summary

This script aggregates annual Vehicle Miles Traveled (VMT) on Federal-Aid roadways by leveraging AADT estimates and segment lengths. The resulting statewide annual VMT is computed and then spatially joined with county-level data to produce a summary CSV file. This document provides an overview of the input parameters, key processing steps, and outputs, ensuring that ITD staff and future developers have a clear reference for maintaining and modifying the annual VMT estimation process within the Toolbox.

# B1. aggregated_aadt_estimation.py

This script estimates Average Annual Daily Traffic (AADT) on rural, off-system roadways in Idaho by integrating existing count station data and independent variable rasters into an Empirical Bayesian Kriging (EBK) regression model. The resulting spatially interpolated AADT raster is sampled along the network of interest, and the median estimated values are computed and joined back to generate raw AADT estimates for each roadway segment. These estimates serve as a basis for further scaling adjustments.

## Inputs and Parameters

### External Dependencies

- **Python Modules:**
  - `arcpy` – For geospatial processing and data management.
  - `math` – For mathematical operations.
  - `pandas` – For data aggregation and manipulation.
  - `arcpy.sa` – For Spatial Analyst tools (e.g., ExtractValuesToPoints).
- **Toolboxes Imported:**
  - Data Management Tools
  - GeoAnalytics Desktop Tools

### Runtime Parameters

1. **Count_Stations:**
   - A feature class of count stations that serve as base points for spatial interpolation.
2. **Network_of_Interest:**
   - The roadway segments (off-system) for which AADT estimates will be produced.
3. **Independent_Variable_Rasters:**
   - A semicolon-delimited list of raster datasets that serve as explanatory variables in the EBK regression analysis.
4. **Out_Segments:**
   - The output destination for the roadway segments with raw, modeled AADT estimates.

## Detailed Functionality and Processes

### Data Preparation

- **Environment and Licensing:**
  - The script sets `arcpy.env.overwriteOutput` to `True` and checks out the necessary extensions (GeoStats and Spatial Analyst).
  - It imports required toolboxes to enable additional geoprocessing functions.
- **Temporary Layers Setup:**
  - Several in-memory feature layers and tables are created (e.g., for count stations, the network of interest, EBK regression outputs, sample points, and summarized results).

    - Any existing temporary outputs with the same names are deleted to avoid conflicts.
- **Filtering the Network:**
    - The input network is copied to a temporary layer and then filtered using a SQL expression (`expression_off_system`) to select only segments that represent off-system roadways.
- **Count Station Proximity:**
    - Count stations are copied to an in-memory layer.
    - A spatial selection is performed to retain only those stations within 5 feet of the filtered network segments (NOI).

### Spatial Interpolation Using EBK Regression

- **EBK Regression Analysis:**
    - The Empirical Bayesian Kriging (EBK) regression model is run on the count stations near the network.
    - The dependent variable (`AADT_LOG`) and the provided independent variable rasters are used to generate a continuous AADT estimate raster.
    - The output raster is converted to a format suitable for subsequent value extraction.

### Value Extraction and Aggregation

- **Generating Sample Points:**
    - Points are generated along the selected network segments at 1-mile intervals (with endpoints included) using `GeneratePointsAlongLines`.
- **Extracting Raster Values:**
    - AADT values from the EBK raster are extracted to the generated sample points using `ExtractValuesToPoints`, storing the results in a temporary points layer.
- **Composite Key Creation and Aggregation:**
    - A composite key is calculated for each sample point by concatenating key attribute values (e.g., RouteID, FromMeasure, ToMeasure, FacilityType, SurfaceType, FHWAUrbanCode, FunctionalClass). This key is used to group points that belong to the same segment.
    - Using pandas, the sample point data are aggregated by composite key to compute the median estimated AADT (RASTERVALU) for each group.
    - The aggregated results are converted back into an in-memory table.

### Value Transformation and Joining

- **Exponentiation:**
    - Since the regression model operates on log-transformed values, the median AADT estimates are exponentiated (with error handling to avoid underflow) to revert to the original scale.
- **Joining Back to the Network:**
    - A composite key is generated for the network segments.
    - The summarized median and exponentiated AADT values are joined to the network of interest based on this composite key.

- Non-essential fields are deleted to streamline the output.

### Exporting the Final Output

- **Output Export:**
  - The network segments, now appended with raw modeled AADT estimates, are exported to the user-specified output destination as a feature layer.

## Outputs Summary

- **Modeled AADT Network Segments:**
  - A feature layer is created containing the roadway segments of the network of interest, each with raw AADT estimates derived from the EBK regression analysis and subsequent median value extraction.
- **Intermediate Data Products:**
  - In-memory layers and tables (e.g., the AADT estimate raster, sample points, and summarized table) are used during processing and serve as key steps in the estimation workflow.

## Considerations for Maintenance and Future Modifications

- **Toolbox Paths and Dependencies:**
  - Verify that the paths to the Data Management and GeoAnalytics Desktop toolboxes remain valid, especially if the ArcGIS installation path changes.
- **Input Data Validity:**
  - Ensure that the input count station and network datasets contain the necessary fields for constructing composite keys and for spatial operations.
  - Review the SQL expression used to filter the network segments to ensure it aligns with evolving definitions of off-system roadways.
- **Spatial Interpolation Parameters:**
  - Adjust parameters for the EBK regression model (e.g., dependent variable, explanatory rasters) as needed based on model performance or data availability.
- **Performance Optimization:**
  - Monitor performance, particularly in the value extraction and pandas-based aggregation steps, and consider optimizing these sections for larger datasets.
- **Error Handling:**
  - Expand error handling (e.g., try/except blocks) around critical operations (such as raster processing, field calculations, and joins) to improve robustness.

## Summary

This script provides an automated workflow to estimate AADT on off-system roadways by integrating spatial interpolation via EBK regression with a series of extraction and aggregation steps. The process involves filtering the network and count stations, modeling AADT values, extracting and summarizing these estimates along roadway segments, and finally joining the results back to the network for export.

This documentation offers a comprehensive reference for ITD staff and future developers to understand, maintain, and extend this component of the Toolbox.

# B2. aadt_estimation_scaling.py

This script rescales raw AADT estimates (generated from previous modeling steps) on off-system roadways so that the total calculated annual Vehicle Miles Traveled (VMT) matches a specified statewide target. The script adjusts the AADT estimates iteratively—first by reducing high values and then by scaling overall totals—to ensure that:

- The maximum AADT does not exceed the 3rd quartile plus the interquartile range (Q3 + IQR) of observed AADT values.
- The overall estimated annual VMT (computed as AADT × segment length × 365) is within a defined tolerance of the target total annual VMT.

## Inputs and Dependencies

### External Dependencies

- **Python Modules:**
  - `arcpy` – For geospatial processing and field calculations.
  - `math` – For mathematical operations.
  - `numpy` – For statistical computations (percentiles).
  - `arcpy.sa` – Spatial Analyst module (if needed in other parts of the workflow).
- **Execution Environment:**
  This script is designed to run within an ArcGIS Pro environment, where the necessary licenses and dependencies are available.

### Runtime Parameters

1. **Estimate_Segments:**
   - A feature layer containing raw AADT estimates on network segments (from prior modeling).
2. **Est_Year:**
   - A numeric or string value representing the estimation year.
3. **Target_VMT:**
   - The target total annual VMT that the scaled AADT values, when multiplied by segment lengths and 365, should sum to. This value is typically derived by subtracting Federal-Aid annual VMT from the statewide annual VMT estimate.
4. **Stations_With_AADT:**
   - A feature layer with actual AADT values from count stations, used for deriving observed AADT statistics.
5. **AADT_Field:**
   - The field in the stations layer that contains the actual AADT values.
6. **Counties:**
   - A feature class of Idaho counties (e.g., TIGERLine counties).
7. **GEOID_Field:**

    - The field in the Counties dataset that holds the 5-digit State-County FIPS codes.
8. **Scaled_Segments:**
    - The output destination (feature layer) for the segments with scaled and rounded AADT values.
9. **VMT_Summary_Table:**
    - The output destination for a summary table that aggregates the rescaled annual VMT by county.

## Detailed Functionality and Processes

### Rescaling Routine

**Function:** `rescale_field_to_target`

- **Purpose:**
  This function rescales a numeric field (raw AADT estimates) to ensure that:
    - The maximum rescaled AADT does not exceed the observed 3rd quartile plus the interquartile range (Q3 + IQR) from count station data.
    - The computed total annual VMT (derived as AADT × segment length × 365) is within an acceptable tolerance of the target value.
- **Key Steps:**
    1. **Extract Initial Values:**
        - Retrieve raw estimated AADT values from the feature layer and actual AADT values from the stations layer.
    2. **Statistical Computations:**
        - Calculate the minimum and raw maximum of the estimated values.
        - Compute the 25th (Q1) and 75th (Q3) percentiles and the IQR for the actual AADT data.
        - Define an upper bound for estimates as Q3 plus the IQR.
    3. **Initial Scaling:**
        - Scale the raw AADT values such that the maximum does not exceed the lower of the raw maximum or Q3 + IQR.
        - Calculate a new annual VMT field using the scaled AADT and segment length.
    4. **Iterative Adjustment:**
        - If the computed total annual VMT is outside the tolerance, iteratively adjust the rescaled AADT:
            - **If Total annual VMT is Too High:** Scale down the rescaled values proportionally.
            - **If Total annual VMT is Too Low:** Scale up either the maximum or minimum values depending on whether the maximum is below the Q3 + IQR threshold.
        - Recalculate annual VMT after each iteration until the total annual VMT is within tolerance or the maximum number of iterations is reached.

**Helper Function:** `round_aadt`

- **Purpose:**
  Implements ITD's AADT rounding rules:
  - For very low values, simple rounding is applied.
  - For larger values, rounding is done to the nearest 10, 100, or 500 as appropriate.
  - Values below 0.5 are rounded to zero, accommodating the possibility of extremely low traffic on remote segments.

### Overall Scaling Workflow

**Function:** `AggregatedAADTScaling`

- **Purpose:**
  Orchestrates the scaling process to adjust raw AADT estimates on the network segments so that their combined annual VMT approximates the target value.
- **Key Processes:**
  1. **Data Preparation:**
     - Create a copy of the input estimate segments as the working network layer.
     - Filter and prepare the counties layer (using the GEOID field) to include only Idaho counties.
     - Perform a spatial join between the network layer and the counties layer to append county FIPS codes to each segment.
  2. **Modeled VMT Calculation:**
     - Calculate the length of each segment (using geodesic measurements in miles).
  3. **Rescaling and Rounding:**
     - Call the `rescale_field_to_target` function to iteratively adjust the raw AADT values.
     - Use the `round_aadt` helper to round the rescaled AADT values according to ITD standards.
     - Remove extraneous fields to retain only necessary attributes.
     - Append the estimation year to the final output.
  4. **Exporting Outputs:**
     - Export the final feature layer containing scaled and rounded AADT estimates to the specified output location.
     - Generate a summary table that aggregates the new annual VMT by county using the county FIPS codes.

## Outputs Summary

- **Scaled Segments Feature Layer:**
  - A feature class containing roadway segments with the rescaled and rounded AADT estimates. These values have been adjusted so that the total estimated annual VMT (when combined with segment lengths and 365 days) is within a specified tolerance of the target statewide annual VMT.
- **VMT Summary Table:**
  - A CSV or table file summarizing the total rescaled annual VMT aggregated by county (identified by their 5-digit FIPS codes).

## Considerations for Maintenance and Future Modifications

- **Parameter Adjustments:**
  - The tolerance for annual VMT matching and the maximum number of iterations for rescaling can be tuned based on future requirements or changes in statewide traffic estimates.
- **Field and Schema Updates:**
  - Monitor any changes in field names or schema in the input datasets. Adjust the SQL expressions, field names, and composite key calculations accordingly.
- **Rounding Methodology:**
  - The `round_aadt` function applies ITD's current rounding rules. If these rules are updated, the function should be revised to reflect the new standards.
- **Performance Optimization:**
  - The iterative scaling loop may need optimization for larger datasets. Consider adding logging or error handling to capture issues during iterations.
- **Error Handling:**
  - While the script provides basic warnings for invalid values, more robust try/except blocks could be added to ensure smoother operation in the event of data anomalies.

## Summary

This script scales and rounds raw AADT estimates for off-system roadways so that the overall estimated annual VMT aligns with a specified statewide target. By adjusting the AADT values iteratively and then applying standardized rounding rules, the script ensures that the network segments' traffic estimates are both realistic and consistent with statewide metrics. This documentation provides a comprehensive overview for ITD staff and future developers to understand, troubleshoot, and modify the scaling process as needed.

# C1. error_evaluation.py

This script compares modeled AADT estimates (provided as line segments) to observed AADT values (recorded at count stations represented as points). It calculates key error metrics—Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percent Error (MAPE)—to evaluate the performance of the AADT estimation model. In addition, the script generates a detailed output table that records the comparison for each point, allowing for further analysis of localized errors.

## Inputs and External Dependencies

### External Dependencies

- **Python Modules:**
  - `arcpy` – For geospatial data processing, spatial queries, and table management.
  - `math` – For mathematical computations (e.g., square root for RMSE).
- **Environment:**
  The script is designed to run within an ArcGIS Pro environment with proper licensing and access to spatial processing tools.

### Runtime Parameters

1. **in_points:**
   - Point feature layer containing actual AADT count stations.
2. **pt_id_field:**
   - Field from the point layer that uniquely identifies each count station (e.g., a Route ID).
3. **pt_value_field:**
   - Field in the point layer containing observed AADT values.
4. **in_lines:**
   - Line feature layer containing modeled or estimated AADT values along roadway segments.
5. **line_id_field:**
   - Field from the line layer that corresponds to the Route ID.
6. **line_value_field:**
   - Field in the line layer with the modeled AADT estimates.
7. **search_distance:**
   - A numeric value (in feet) defining the maximum distance within which count stations are associated with nearby segments.
8. **out_table:**
   - Output name for a summary table of error metrics (RMSE, MAE, MAPE).
9. **detailed_out_table:**
   - Output name for a detailed table that logs the error (difference) for each count station compared with its nearest line segment.

## Detailed Functionality and Processes

### Name Sanitization

- **Function:** `sanitize_name(name)`
- **Process:**
    - Replaces invalid characters in the raster name with underscores.
    - Ensures that the name does not begin with a number or an underscore.
    - Truncates the name to a maximum of 50 characters to comply with geodatabase naming restrictions.

### Spatial Association and Data Preparation

- **Temporary Layer Creation:**
  A temporary feature layer is created from the input lines to ensure that the original dataset remains unmodified.
- **Selecting Relevant Points:**
  Points (count stations) are spatially selected if they fall within the specified search distance (in feet) of the line segments. This spatial query limits the analysis to count stations that are in close proximity to the modeled segments.

### Iterative Comparison Process

- **Processing Each Count Station:**
  For each count station:
    - A buffer (using the provided search distance) is generated to identify nearby line segments.
    - A subsequent attribute query ensures that only line segments with a matching Route ID (or equivalent identifier) are considered.
- **Calculating Errors:**
    - If multiple segments are found, their AADT estimates are averaged.
    - The error is computed as the difference between the observed AADT (point) and the averaged modeled AADT (line).
    - The error is stored in a list for global metric calculation.
    - Denominator values (absolute modeled AADT) are recorded for use in MAPE calculation, with points having a zero-value skipped for MAPE.

### Calculation of Error Metrics

- **Error Metrics Computation:**
  Once all points have been processed:
    - **RMSE** is computed as the square root of the mean of squared errors.
    - **MAE** is computed as the mean of the absolute errors.
    - **MAPE** is computed as the average of the absolute percentage errors (only including points with nonzero modeled AADT), multiplied by 100 to yield a percentage.

- **Progress and Warnings:**
  The script uses progressors and logs messages to provide status updates during processing, and it issues warnings if any count station has no matching segment or if a segment's modeled value is zero (which would affect MAPE).

### Output Table Creation

- **Summary Table:**
  An output table is created (or updated) to record the overall error metrics (RMSE, MAE, MAPE).
- **Detailed Comparison Table:**
  A separate detailed output table is generated that logs, for each count station, the following:
  - Count station ID
  - Observed AADT value at the point
  - Associated segment IDs (if more than one, combined as a comma-separated string)
  - Averaged modeled AADT value from the segment(s)
  - Calculated error (difference)
- Both tables are written to user-specified destinations.

## Outputs Summary

- **Error Metrics Table (out_table):**
  - Contains a single row with the computed RMSE, MAE, and MAPE values.
- **Detailed Comparison Table (detailed_out_table):**
  - Contains individual comparison records for each count station, including the observed and modeled AADT values and their differences.

## Considerations for Maintenance and Future Modifications

- **Spatial Query Parameters:**
  - The search distance is a critical parameter. Adjustments may be needed based on changes in the spatial distribution of count stations or segmentation logic.
- **Field Name Consistency:**
  - Ensure that the field names for unique identifiers and AADT values in both the point and line layers remain consistent. Any schema changes in the input datasets should be reflected in the script parameters.
- **Error Handling Enhancements:**
  - While the script currently logs warnings for points with no matching lines or zero modeled values, additional error handling (e.g., try/except blocks) could further improve robustness.
- **Performance Considerations:**
  - For larger datasets, consider optimizing the use of cursors and spatial queries to reduce processing time.
- **Output Table Management:**
  - The script creates temporary and output tables in a scratch workspace if needed. Validate that the workspace paths remain valid in your environment.

## Summary

This script is a vital component of the Toolbox that quantitatively evaluates the performance of the AADT estimation model. By comparing modeled values against real-world counts at nearby locations, it provides key error metrics (RMSE, MAE, MAPE) and detailed comparison data. This documentation enables ITD staff and future developers to understand the underlying logic, maintain the codebase, troubleshoot issues, and modify the process as data or requirements evolve.

# 6. Works Cited


AASHTO. 2009. "AASHTO Guidelines for Traffic Data Programs."

Ada County Highway District. 2022. *Traffic Counts.* Accessed September 28, 2022. http://www.achdidaho.org/Departments/Engineering/Traffic/trafficCounts.aspx.

Anderson, Michael D, Khalid Sharfi, and Sampson E Gholston. 2006. "Direct demand forecasting model for small urban communities using multiple linear regression." *Transportation Research Record* 1981 (1): 114-117. Accessed June 8, 2022. http://dx.doi.org/10.3141/1981-18.

Apronti, Dick, Khaled Ksaibati, Kenneth Gerow, and Jamie Jo Hepner. 2016. "Estimating traffic volume on Wyoming low volume roads using linear and logistic regression methods." *Journal of Traffic and Transportation Engineering* 493-506. Accessed June 2, 2022. doi:http://dx.doi.org/10.1016/j.jtte.2016.02.004.

Baffoe-Twum, Edmund, Eric Asa, and Bright Awuku. 2023. "Estimating annual average daily traffic (AADT) data on low-volume roads with the cokriging technique and census/population data." *Emerald Open Research.* doi:https://doi.org/10.35241/emeraldopenres.14632.2.

Ben-Gurion University of the Negev. 2022. *Chapter 12: Spatial Interpolation of Point Data.* February 21. Accessed November 4, 2022. http://132.72.155.230:3838/r/spatial-interpolation-of-point-data.html.

Bivand, Roger. 2009. "Applying Measures of Spatial Autocorrelation: Computation and Simulation." *Geographical Analysis* 41 (4): 375-384. doi:10.1111/j.1538-4632.2009.00764.x.

Calderon, Vicky, interview by Peter Hylton and Bryce Miller. 2022. *Email* (Deember 20).

Calderon, Vicky, and Amanda Laib, interview by Peter Hylton and Bryce Miller. 2022. *ITD Off-System Roads AADT Estimation Study - Progress Meeting #9* (September 27).

Castro-Neto, Manoel, Youngseon Jeong, Myong K. Jeon, and Lee D. Han. 2009. "AADT prediction using support vector regression with data-dependent aprameters." *Expert Systems with Applications* 2979-2986. Accessed May 27, 2022. doi:10.1016/j.eswa.2008.01.073.

Coladner, David, interview by Peter Hylton. 2022. (October 20).

Coladner, David, Amanda Laib, and Vicky Calderon, interview by Peter Hylton and Bryce Miller. 2022. *Modernization Processing Tools* (June 23).

Das, Subasish, and Ioannis Tsapakis. 2020. "Interpretable machine learning approach in estimating traffic volume on low-volume roadways." *International Journal of Transportation Science and Technology* 76-88. Accessed June 2, 2022. doi:https://rosap.ntl.bts.gov/view/dot/29681.

datagy. 2022. *How to Calculate MAPE in Python.* Feburary 11. Accessed January 13, 2023. https://datagy.io/mape-python/.

Datarade. 2022. *INRIX Real-time Traffic Flow from GPS data (for Europe and North America).* Accessed October 7, 2022. https://datarade.ai/data-products/inrix-real-time-traffic-flow.

DeVine, Ryan Andrew. 2020. "Use of Geospatial Technique to Estimate Traffic Volume on South Carolina Roads (Doctoral Dissertation)." Masters Thesis. Accessed June 2, 2022. https://scholarcommons.sc.edu/etd/6125 .

Dixon, Michael. 2004. "The effects of errors in annual average daily traffic forecasting: study of highways in Rural Idaho." Accessed June 8, 2022. https://apps.itd.idaho.gov/apps/research/Completed/RP167.pdf.

Eom, Jin Ki, Man Sik Park, Tae-Young Heo, and Leta F Huntsinger. 2006. "Improving the prediction of annual average daily traffic for nonfreeway facilities by applying a spatial statistical method." *Transportation Research Record* 1968 (1): 20-29. Accessed June 8, 2022. https://doi.org/10.1177%2F0361198106196800103.

Esri. n.d. *FAQ: What version of Python is used in ArcGIS?* Accessed January 13, 2023. https://support.esri.com/en/technical-article/000013224.

—. 2021. *How IDW works.* Accessed January 13, 2023. https://desktop.arcgis.com/en/arcmap/latest/tools/3d-analyst-toolbox/how-idw-works.htm#:~:text=IDW%20relies%20mainly%20on%20the%20inverse%20of%20the,real%20number%2C%20and%20its%20default%20value%20is%202.

Federal Highway Administration. 2016. "Appendix C. Vehicle Types." In *Traffic Monitoring Guide*. Accessed January 17, 2023. https://www.fhwa.dot.gov/policyinformation/tmguide/tmg_2013/vehicle-types.cfm.

Federal Highway Administration. 2018. "Highway Performance Monitoring System Field Manual." Accessed September 23, 2022. https://www.fhwa.dot.gov/policyinformation/hpms/fieldmanual/page00.cfm.

Federal Highway Administration. 2018. "Traffic Data Computation Method." Pocket Guide. https://www.fhwa.dot.gov/policyinformation/pubs/pl18027_traffic_data_pocket_guide.pdf.

Federal Highway Administration. 2016. "Traffic Monitoring Guide." https://www.fhwa.dot.gov/policyinformation/tmguide/.

Federal Highway Administration. 2022. "Traffic Monitoring Guide." Accessed January 23, 2023. https://www.fhwa.dot.gov/policyinformation/tmguide/.

—. 2022. *Traffic Volume Trends.* Accessed July 18, 2022. https://www.fhwa.dot.gov/policyinformation/travel_monitoring/tvt.cfm.

Fortin, Marie-Josée, Pierre Drapeau, and Pierre Legendre. 2012. "Spatial autocorrelation and sampling design in plant ecology." In *Progress in theoretical vegetation science*, by G. Grabherr, L. Mucina, M. B. Dale and C. J. F. Braak, 209–222. doi:10.1007/978-94-009-1934-1_18.

Fu, Miao, J. Andrew Kelly, and J. Peter Clinch. 2017. "Estimating annual average daily traffic and transport emissions for a national road network: A bottom-up methodology for both nationally-aggregated and spatially-disaggregated results." *Journal of Transport Geography* 186-195. Accessed June 2, 2022. http://dx.doi.org/10.1016/j.jtrangeo.2016.12.002.

Gadda, Shashank, Atul Magoon, and Kara M. Kockelman. 2007. "Estimates of AADT: Quantifying the Uncertainty." *86th Annual Meeting of the Transportatin Research Board.* Washington, DC.

Golbeck, Jennifer. 2015. "Analyzing Networks." In *Introduction to Social Media Investigation*, by Jennifer Golbeck. Accessed September 23, 2022. https://www.sciencedirect.com/topics/computer-science/degree-centrality.

Golbeck, Jennifer. 2013. "Network Structure and Measures." In *Analyzing the Social Web*, by Jennifer Golbeck. Accessed September 23, 2022. doi:https://doi.org/10.1016/C2012-0-00171-8.

Hansen, Derek L., Ben Shneiderman, Smith Marc A., and Itai Himelboim. 2020. "Social network analysis: Measuring, mapping, and modeling collections of connections." In *Analyzing Social Media Networks with NodeXL*, by Derek L. Hansen, Ben Shneiderman, Smith Marc A. and Itai Himelboim, edited by Second. Accessed September 23, 2022. doi:https://doi.org/10.1016/C2018-0-01348-1.

Hanson, Nicole, and Vicky Calderon, interview by Peter Hylton and Bryce Miller. 2022. *GIS and AADT Temporality* (June 23).

Holik, William, Ioannis Tsapakis, Anita Vandervalk, Shawn Turner, and John Habermann. 2017. *Innovative Traffic Data QAQC Procedures and Automating AADT Estimation.* Federal Highway Adminstration Office of Safety. Accessed June 16, 2022. https://safety.fhwa.dot.gov/rsdp/downloads/fhwasa17035.pdf.

Huynh, Nathan, Jing Wang, Ryan DeVine, Mashurur "Ronnie" Chowdhury, Weimin Jin, Pronab Biswas, and Gurcan Comert. 2021. *Estimating AADT on Non-coverage Roads.* South Carolina Department of Transportation. Accessed June 16, 2022. https://scdot.scltap.org/wp-content/uploads/2022/01/SPR-749-Final-Report-2.pdf.

Idaho Association of Highway Districts. 2022. *Highway District Members.* Accessed September 28, 2022. https://iahd.com/highway-district-members/ .

Idaho Geospatial Office. 2022. *Land Use & Land Cover TWG.* Accessed September 23, 2022. https://gis.idaho.gov/land-use-land-cover-twg.

Idaho Transportation Department . 2022. *Road Data.* Accessed September 28, 2022. https://itd.idaho.gov/road-data/.

Idaho Transportation Department. 2022. "Historical Count History: 1960-2021." Accessed October 25, 2022.

Idaho Transportation Department. 2016. "Idaho Transportation Department Systems Procedures." 22. https://apps.itd.idaho.gov/apps/plan/ITDSystemsProcedures.pdf.

—. 2022. *ITD Open Data.* Accessed August 25, 2022. https://data-iplan.opendata.arcgis.com/.

Idaho Transportation Department. 2020. "Traffic Smoothing Documentation Guide."

INRIX. 2014. "I-95 Vehicle Probe Project II." https://tetcoalition.org/wp-content/uploads/2015/02/I-95-VPP-II-INRIX-Inteface-Guide-v1-Dec-2014.pdf.

—. 2017. *INRIX Selected by The U.S. Federal Highway Administration for National Traffic Data Set.* June 5. Accessed October 7, 2022. https://inrix.com/press-releases/npmrds/.

—. 2022. *INRIX Volume Profile.* Accessed October 7, 2022. https://inrix.com/products/volume/.

INSIDE Idaho. 2020. "Attributes for Idaho Transportation." https://insideidaho.org/data/ago/uofi/library/roads/roadTransportationAttributesDescription.pdf.

Islam, Sababa. 2016. *Estimation of Annual Average Daily Traffic (AADT) and Missing Hourly Volume Using Artificial Intelligence.* Masters Thesis, Clemson University. Accessed June 2, 2022. https://tigerprints.clemson.edu/all_theses/2562.

Jayasinghe, Amila, Kazushi Sano, C. Chethika Abenayake, and P.K.S. Mahanama. 2019. "A novel approach to model traffic on road segments of large-scale urban road networks." *MethodsX* 1147-1163. Accessed June 2, 2022. https://doi.org/10.1016/j.mex.2019.04.024.

Jiang, Zhuojun, Mark R McCord, and Prem K Goel. 2006. "Improved AADT estimation by combining information in image-and ground-based traffic data." *Journal of Transportation Engineering* (American Society of Civil Engineers) 132 (7): 523-530. Accessed June 8, 2022. https://doi.org/10.1061/(ASCE)0733-947X(2006)132:7(523).

Keehan, McKenzie. 2017. "Annual Average Daily Traffic (AADT) Estimation with Regression Using Centrality and Roadway Characteristic Variables." Thesis, Clemson University. Accessed September 26, 2022. https://tigerprints.clemson.edu/all_theses/2644.

Kehan, McKenzie. 2017. "Annual Average Daily Traffic (AADT) Estimation with Regression Using Centrality and Roadway Characteristic Variables." *All Theses.* Clemson University. Accessed June 2, 2022. https://tigerprints.clemson.edu/all_theses/2644.

Kumar, Amrit, Rajeev Kumar Mishra, and Kiranmay Sarma. 2020. "Mapping spatial distribution of traffic induced criteria pollutants and associated health risks using kriging interpolation tool in Delhi." *Journal of Transport & Health* 18. doi:https://doi.org/10.1016/j.jth.2020.100879.

Liu, Jinhao, Jinming Liu, Zhongwei Li, Guoliang Dai, and Xiaoyu Hou. 2021. "Estimating CPT Parameters at Unsampled Locations Based on Kriging Interpolation Method." *Applied Sciences* 11. doi:https://doi.org/10.3390/app112311264.

Lowry, Michael. 2014. "Spatial interpolation of traffic counts based on origin--destination centrality." *Journal of Transport Geography* 36: 98-105. Accessed June 8, 2022. https://doi.org/10.1016/j.jtrangeo.2014.03.007.

Lowry, Michael, and Michael Dixon. 2012. *GIS Tools to Estimate Average Annual Daily Traffic.* Final Report, US Department of Transportation Research and Special Programs Administration. Accessed May 27, 2022. https://rosap.ntl.bts.gov/view/dot/24942.

Marques, Samuel de Franca, and Cira Souza Pitombo. 2021. "Ridership Estimation Along Bus Transit Lines Based on Kriging: Comparative Analysis Between NEtwork and Euclidean Distances." *Journal of Geovisualization and Spatial Analysis* 5. doi:https://doi.org/10.1007/s41651-021-00075-w.

Marsden, Peter V. 2005. "Network Analysis." In *Encyclopedia of Social Measurement*, by Kimberly Kempf-Leonard. Elsevier. Accessed September 23, 2022. https://www.sciencedirect.com/topics/computer-science/centrality-measure.

Mathew, Sonu. 2020. *Modeling Annual Average Daily Traffic for Local Functionally Classified Roads.* PhD Dissertation, University of North Carolina at Charlotte. Accessed June 2, 2022. https://repository.charlotte.edu/islandora/object/etd%3A2066.

McCord, Mark R., and Prem Goel. 2009. *Estimating AADT from combined air photos and ground based data: System design, prototyping, and testing.* NEXTRANS Project, USDOT Region V Regional Univeristy Transportation Center. Accessed June 2, 2022. https://rosap.ntl.bts.gov/view/dot/5707.

Mohamad, Dadang, Kumares C Sinha, Thomas Kuczek, and Charles F Scholer. 1998. "Annual average daily traffic prediction model for county roads." *Transportation Research Record* 1617 (1): 69-77. Accessed June 8, 2022. https://doi.org/10.3141%2F1617-10.

Moody, James. 2019. *What does RMSE really mean?* September 5. Accessed November 1, 2022. https://towardsdatascience.com/what-does-rmse-really-mean-806b65f2e48e.

Morley, Steve Karl. 2016. *Alternatives to accuracy and bias metrics based on percentage errors for radiation belt modeling applications.* Los Alamos National Laboratory. https://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-16-24592.

Owaniyi, Kunle Meshach. 2019. *Geostatistical Interpolation and Analyses of Washington State AADT Data from 2009-2016.* Master's Thesis, Construction Management and Engineering, Fargo: North Dakota State University of Agriculture and Applied Science.

Pan, Tao. 2008. "Assignment of Estimated Average Annual Daily." Thesis, University of South Florida. https://digitalcommons.usf.edu/cgi/viewcontent.cgi?httpsredir=1&article=1441&context=etd#:~:text=Assignment%20of%20Estimated%20Average%20Annual%20Daily%20Traffic%20Volumes, fulfillment%20of%20the%20requirements%20for%20the%20degree%20of.

Pan, Tao. 2008. *Assignment of estimated average annual daily traffic volumes on all roads in Florida.* Master's Thesis, University of South Florida. Accessed June 8, 2022. https://digitalcommons.usf.edu/etd/442/.

Park, Nokil. 2004. *Estimation of Average Annual Daily Traffic (AADT) Using Geographically Weighted Regression (GWR) Method and Geographic Information System (GIS).* PhD Dissertation, Florida Internaional University. Accessed June 2, 2022. https://digitalcommons.fiu.edu/dissertations/AAI3128612/.

Pridmore, Margaret. 2022. *Email.* August 30.

Pridmore, Margaret, and Vicky Calderon, interview by Peter Hylton and Bryce Miller. 2022. *Historical Method/Modernization* (June 21).

Pulugurtha, Srinivas S, and Prasanna R Kusam. 2012. "Modeling annual average daily traffic with integrated spatial data from multiple network buffer bandwidths." *Transportation Research Record* 2291 (1): 53-60. Accessed June 8, 2022. https://doi.org/10.3141%2F2291-07.

Pulugurtha, Srinivas S., and Sonu Mathew. 2020. *Develop Local Functional Classification VMT and AADT Estimation Method.* Final Report, North Carolina Department of Transportation Research and Analysis Group. Accessed June 2, 2022. https://connect.ncdot.gov/projects/research/RNAProjDocs/Pulugurtha_NCDOT-RP2019-12_FinalReport.pdf.

Raja, Prithiviraj, Mehrnaz Doustmohammadi, and Michael D. Anderson. 2018. "Estimation of Average Daily Traffic on Low Volume Roads in Alabama." *International Journal of Traffic and Transportation Engineering* 1-6. Accessed June 2, 2022. doi:10.5923/j.ijtte.20180701.01.

Replica. 2022. *Solutions.* Accessed October 7, 2022. https://replicahq.com/solutions/.

Robinson, T. P., and Graciela Metternicht. 2006. "Testing the performance of spatial interpolation techniques for mapping soil properties." *Computers and electronics in agriculture* 50 (2): 97-108. 10.1016/j.compag.2005.07.003.

Seaver, William L, Arun Chatterjee, and Mark L Seaver. 2000. "Estimation of traffic volume on rural local roads." *Transportation Research Record* 1719 (1): 121-128. Accessed June 8, 2022. https://doi.org/10.3141%2F1719-15.

Selby, Brent, and Kara M Kockelman. 2013. "Spatial prediction of traffic levels in unmeasured locations: applications of universal kriging and geographically weighted regression." *Journal of Transport Geography* 29: 24-32. Accessed June 8, 2022. https://doi.org/10.1016/j.jtrangeo.2012.12.009.

Selby, Brent, and Kara M. Kockelman. 2013. "Spatial prediction of traffic levels in unmeasured locations: applications of universal kriging and geographically weighted regression." *Journal of Transport Geography* 29: 24-32.

Sfyridis, Alexandros, and Paolo Agnolucci. 2020. "Annual average daily traffic estimation in England and Wales: An application of clustering and regression modelling." *Journal of Transport Geography* 83: 102658. Accessed June 8, 2022. https://doi.org/10.1016/j.jtrangeo.2020.102658.

Shamo, Benedict, Eric Asa, and Joseph Membah. 2015. "Linear spatial interpolation and analysis of annual average daily traffic data." *Journal of Computing in Civil Engineering* 29 (1). Accessed June 8, 2022. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000281.

Sharma, Satish C, Pawan Lingras, Fei Xu, and Guo X Liu. 1999. "Neural networks as alternative to traditional factor approach of annual average daily traffic estimation from traffic counts." *Transportation Research Record* 1660 (1): 24-31. Accessed June 8, 2022. https://doi.org/10.3141%2F1660-04.

Sharma, Satish, Pawan Lingras, Fei Xu, and Peter Kilburn. 2001. "Application of neural networks to estimate AADT on low-volume roads." *Journal of Transportation Engineering* 127 (5): 426-432. Accessed June 8, 2022. https://doi.org/10.1061/(ASCE)0733-947X(2001)127:5(426).

Silverberg, Jonathan and Wichman, Chris, interview by Peter and Miller, Bryce Hylton. 2022. (September 15).

Staats, William Nicholas. 2016. *Estimation of Annual Average Daily Traffic on Local Roads in Kentucky.* Master's Thesis, University of Kentucky. Accessed May 27, 2022. https://uknowledge.uky.edu/ce_etds/36.

Statology. 2020. *How to Calculate MAPE in Python.* July 7. https://www.statology.org/mape-python/.

—. 2020. *How to Calculate RMSE in Python.* September 3. Accessed January 13, 2023. https://www.statology.org/rmse-python/.

StreetLight Data. 2022. *Get more data, at finer resolution, for 1/3 the cost.* Accessed September 25, 2022. https://www.streetlightdata.com/streetlight-cost-analysis/.

—. 2022. *StreetLight AADT.* Accessed September 26, 2022. https://www.streetlightdata.com/aadt-average-annual-daily-traffic-count/.

Sun, Xiaoduan, and Subasish Das. 2015. "Developing a Method for Estimating AADT on All Louisiana Roads." Final Report, Louisiana Department of Transportation and Devleopment. Accessed May 27, 2022. https://rosap.ntl.bts.gov/view/dot/29681.

Tobler, Waldo R. 1970. "A Computer Movie Simulating Urban Growth in the Detroit Region." *Economic Geography* 234-240.

Tobler, Waldo R. 1970. "A Computer Movie Simulating Urban Growth in the Detroit Region." *Economic Geography* (Taylor & Francis, Ltd.) 46: 234-240. doi:https://doi.org/10.2307/143141.

Tsapakis, Ioannis, William Holik, Subasish Das, Edgar Kraus, and Paul Anderson. n.d. *Data Collection and Annual Average Daily Traffic (AADT) Estimation for Non-Federal Aid System (NFAS) Roads.* FHWA-SA-20-064, Washington, DC: Federal Highway Administration. https://safety.fhwa.dot.gov/rsdp/downloads/fhwasa20064.pdf.

Tsapakis, Ioannis, William Holik, Subasish Das, Edgar Kraus, and Paul Anderson. 2020. "Informational Guide on Data Collection and Annual Average Daily Traffic (AADT) Estimation for Non-Federal Aid-System (NFAS) Roads." Accessed June 2, 2022. https://safety.fhwa.dot.gov/rsdp/downloads/fhwasa20064.pdf.

U.S. Census Bureau. 2022. *Cartographic Boundary Files.* Accessed September 23, 2022. https://www.census.gov/geographies/mapping-files/time-series/geo/cartographic-boundary.html#:~:text=Cartographic%20Boundary%20Files%20The%20cartographic%20boundary%20files%20are,are%20specifically%20designed%20for%20small%20scale%20thematic%20mapping.

U.S. Census Bureau. 2022. "Housing Units." Accessed February 23, 2023. https://www.census.gov/quickfacts/fact/note/US/HSG010221.

—. 2022. *Land Area and Persons Per Square Mile.* Accessed September 23, 2022. https://www.census.gov/quickfacts/fact/note/US/LND110210.

U.S. Census Bureau. n.d. "LEHD Origin-Destination Employment Statistics (LODES) Dataset Structure, Fromat Version 7.5." Accessed September 23, 2022. https://lehd.ces.census.gov/data/lodes/LODES7/LODESTechDoc7.5.pdf.

U.S. Geological Survey. 2018. *National Land Cover Data Base.* September 11. Accessed September 23, 2022. https://www.usgs.gov/centers/eros/science/national-land-cover-database#overview.

Unnikrishnan, A., M. Figliozzi, M.K. Moughari, and S. Urbina. 2018. *A Method to Estimate Annual Average Daily Traffic for Minor Facilities for MAP-21 Reporting and Statewide Safety Analysis.* Final Report, Oregon Department of Transportation. Accessed June 16, 2022. https://www.oregon.gov/ODOT/Programs/ResearchDocuments/SPR_804_Final_Report.pdf.

Wang, Tao. 2012. *Improve Annual Average Daily Traffic (AADT) Estimation for Local Roads Using Parcel-Level Travel Demand Modeling.* PhD Dissertation, Florida International University. Accessed May 27, 2022. https://digitalcommons.fiu.edu/dissertations/AAI3517050/.

Wang, Tao. 2012. *Improved Annual Average Daily Traffic (AADT) Estimation for Local Roads using Parcel-Level Travel Demand Modeling.* Dissertation, Florida International University. Accessed 25 October, 2022. https://digitalcommons.fiu.edu/dissertations/AAI3517050/.

Wang, Tao, Albert Gan, and Priyanka Alluri. 2013. "Estimating annual average daily traffic for local roads for highway safety analysis." *Transportation Research Record* 2398 (1): 60-66. Accessed June 8, 2022. https://doi.org/10.3141%2F2398-07.

Wang, Xiaokun, and Kara M. Kockelman. 2009. "Forecasting network data: Spatial interpolation of traffic counts from texas data." *Transportation Research Record* 2105 (1): 100-108.

Wang, Xiaokun, and Kara M. Kockelman. 2009. "Forecasting Network Data: Spatial Interpolation of Traffic Counts Using Texas Data." *Transportation Research Rcord* 100-108. Accessed May 27, 2022. https://www.caee.utexas.edu/prof/kockelman/public_html/trb09aadtkriging.pdf.

Xia, Qing, Fang Zhao, Zhenmin Chen, L David Shen, and Diana Ospina. 1999. "Estimation of annual average daily traffic for nonstate roads in a Florida county." *Transportation Research Record* 1660 (1): 32-40. Accessed June 8, 2022. https://doi.org/10.3141%2F1660-05.

Yang, Bingduo, Sheng-Guo Wang, and Yuanlu Bao. 2014. "New Efficient Regression Method for Local AADT Estimation via SCAD Variable Selection." *IEEE Transactions on Intelligent Transportation Systems* (IEEE) 15 (6): 2726-2731. Accessed June 8, 2022. http://dx.doi.org/10.1109/TITS.2014.2318039.

Zhao, Fang, and Nokil Park. 2004. "Using geographically weighted regression models to estimate annual average daily traffic." *Transportation Research Record* 1879 (1): 99-107. Accessed June 8, 2022. https://doi.org/10.3141%2F1879-12.

Zhao, Fang, and Soon Chung. 2001. "Contributing factors of annual average daily traffic in a Florida county: exploration with geographic information system and regression models." *Transportation Research Record* 1769 (1): 113-122. Accessed June 8, 2022. https://doi.org/10.3141%2F1769-14.

Zhong, Ming, and Brody L Hanson. 2009. "GIS-based travel demand modeling for estimating traffic on low-class roads." *Transportation Planning and Technology* 32 (5): 423-439. Accessed June 8, 2022. https://doi.org/10.1080/03081060903257053.

# 7. Appendix A: Variables for Potential Use in Geospatial Interpolation Model

Table 7-1 summarizes the variables that were reviewed for used in the geospatial interpolation model. Not all variables were used in the final model. The final variables that were found to have the greatest overall predictive power and that were used in final tool development are designated with an asterisk (*).

**Table 7-1. Variables and Data Sets**

| Variable Category | Variable | Comments | Data Name | Data Source |
|---|---|---|---|---|
| Traffic Volume* | Average Annual Daily Traffic (AADT) | AADT values derived from as far back as 10 years ago can be used as traffic patterns in rural areas are less likely to change quickly. | historical_count_history_1960-2021 | ITD |
| Roadway | Number of through lanes | Data not available for all facilities. Assume two lanes when unknown. | rhgdb.ITDRH.LRSE_HPMS_ThroughLanes | ITD, HPMS Coordinator |
| Roadway* | Surface type | Only distinguishes paved and unpaved roads on Federal-Aid System roads. Assume unpaved in rural areas and paved in urban areas when information not available from Local Road Inventory. | rhgdb.ITDRH.LRSE_HPMS_SurfaceType<br>• Use where data is available.<br>rhgdb.ITDRH.LRSE_LocalRoadInventory<br>• Use to supplement HPMS surface type data. | ITD, HPMS Coordinator |
| Roadway* | Functional class | Classes must be treated in geospatial interpolation models as categorical data rather than ordinal or interval. | rhgdb.ITDRH.LRSE_FunctionalClass | ITD, HQ Planning |
| Network | Degree centrality | Indicates the number of connected facilities. Calculate using centrality tools in GIS applications. | rhgdb.ITDRH.LRSE_RoadNetwork | ITD, HQ GIS |
| Network | Closeness centrality | Indicates proximity to the rest of the network. Calculate using centrality tools in GIS applications. | rhgdb.ITDRH.LRSE_RoadNetwork | ITD, HQ GIS |
| Network | Betweenness centrality | Indicates how often a segment or intersection is part of a shortest route. Calculate using centrality tools in GIS applications. | rhgdb.ITDRH.LRSE_RoadNetwork | ITD, HQ GIS |
| Network | Distance to intersection | Distance from each road segment to the nearest intersection. | rhgdb.ITDRH.LRSE_RoadNetwork | ITD, HQ GIS |
| Network | Intersection density | Intersections per square mile. Calculated for a given radius or as a moving average because administrative boundaries such as counties or Census tracts vary in size. | rhgdb.ITDRH.LRSE_RoadNetwork | ITD, HQ GIS |

| Variable Category | Variable | Comments | Data Name | Data Source |
|---|---|---|---|---|
| Network | Accessibility (to Primary or Secondary Roads) | Network distance to the nearest facility of functional class 1-2 (Primary) or 3-5 (Secondary). | rhgdb.ITDRH.LRSE_RoadNetwork | ITD, HQ GIS |
| Network* | Road mileage density | Miles of road per square mile. Calculated for a given radius or as a moving average because administrative boundaries such as counties or Census tracts vary in size. | rhgdb.ITDRH.LRSE_RoadNetwork | ITD, HQ GIS |
| Network* | Urban / rural designation | Determined by ITD. | rhgdb.ITDRH.LRSE_UrbanLimits | ITD |
| Economic* | Employment density | Use residence area characteristics. Aggregate at the Census tract level. Use land area (ALAND) from Tiger/LINE shapefiles to calculate density. | LEHD LODES<br>• Used to calculate total employment.<br>Tiger/LINE<br>• Use for land area (ALAND is the attribute for land area in square meters) | U.S. Census Bureau |
| Economic | Employment by Industry | Use two-digit NAICS codes to define industries. Use workplace area characteristics. Aggregate at the Census tract level. | LEHD LODES | U.S. Census Bureau |
| Economic | Median income | Aggregate at the Census tract level. | American Community Survey | U.S. Census Bureau |
| Economic | Poverty rates | Can be calculated as a rate (x per 1000 residents) or as a percentage of residents. Aggregate at the Census tract or county levels. | American Community Survey | U.S. Census Bureau |
| Economic | Nearby population | Although necessary for other calculations, this factor is likely to have high correlation to population density and to be less informative. If used, add an approximately ¼-mile buffer around each road segment and calculate the population density within that buffer. | Decennial Census, P1 \| Race<br>• Use for population. | U.S. Census Bureau |
| Demographic* | Population density | Calculate at the Census tract level using population data from the Decennial Census and land area (ALAND) from Tiger/LINE shapefiles. | Decennial Census, P1 \| Race<br>• Use for population.<br>Tiger/LINE<br>• Use for land area (ALAND is the attribute for land area in square meters) | U.S. Census Bureau |

| Variable Category | Variable | Comments | Data Name | Data Source |
|---|---|---|---|---|
| Demographic | Dwelling units | Dwelling units are likely correlated with population variables but may be informative relative to the ratio of vehicles to population. | DEC Redistricting Data, H1 \| Occupancy status | U.S. Census Bureau |
| Demographic | School enrollment | Can be calculated as a rate (x per 1000 residents) or as a percentage of residents. This should be calculated at the Census tract level and assigned to roads within the Census tract. | ACS, S1401 \| School enrollment | U.S. Census Bureau |
| Demographic | Vehicle registration | Vehicle registrations are only available down to the county level and may therefore be less useful than more granular data. If used, this variable should capture the number of vehicles registered in the county. | DMV Data, "Driver Licenses" tab | ITD |

# 8. Appendix B: Tool Python Code

What follows is the text for the scripts that underlay the Off-System AADT Estimation toolbox. To run the tool, copy the text for each script into separate Python files (.py), and then direct the toolbox tools to the file from their “Properties” > “Execution” tab.

## create_input_data.py

```
"""

Creates input datasets for the AADT estimation process by accessing ITD databases and user-provided data.



Author: Idaho Transportation department and High Street Consulting Group

Date: February 2025

"""

import requests



import geopandas as gpd

import pandas as pd



def LEHD_WAC_Block():

    """

    Joins the tabular LEHD LODES WAC data to the spatial census block data.



    Returns:

        Does not return a value. Writes the joined data to the output directory.

    """
```

```
    # Path to LEHD LODES data and prepare data
    lehd_data = os.path.join(input_dir, "lehd_wac.csv")
    df = pd.read_csv(lehd_data)
    df.w_geocode = df.w_geocode.astype(str)
    df = df.set_index("w_geocode")

    # Path to census block data and prepare data
    blocks = gpd.read_file(os.path.join(input_dir, "tl_2023_16_tabblock20.shp") )
    blocks.GEOID20 = blocks.GEOID20.astype(str)
    blocks = blocks.set_index("GEOID20")

    # Join spatial and tabular data
    data = blocks.join(df)
    data = data.to_crs("EPSG:8826")
    data["Area_MI"] = (data.geometry.area /  2.59e+6)

    # Export spatial LODES data
    out_path = os.path.join(processing_dir, "lehd_wac_block.shp")
    data.to_file(out_path)



def create_overlay_network():
    """
    Creates the primary input network containing information from both the Federal-Aid System and the
off-system public roads.
```

```
    Returns:

        Does not return a value. Adds two feature layers to the project geodatabase. One for the whole
network, and another for the FederalAid System.

    """


    project = arcpy.mp.ArcGISProject("CURRENT")


    # Select the feature layers with the needed characteristics for subsequent analyses

    # These addresses will need to be updated if and when the location of ITD's GIS database changes or
the provide changes to ensure a proper connection.

    features = dict()

# Insert your local geodatabase connection to the following lines.

    features["AADT"] =

    features["FunctionalClass"] =

    features["FacilityType"] =

    features["UrbanLimits"] =

    features["ThroughLanes"] =

    features['LRI'] =



    roadNetwork =


    for name, dbLink in features.items():

        arcpy.AddMessage("Creating layer: {0}".format(name))

        arcpy.management.MakeFeatureLayer(dbLink, name)
```

```
    arcpy.AddMessage("Creating layer: roadNetwork")

    arcpy.management.MakeFeatureLayer(roadNetwork, "roadNetwork")



    arcpy.AddMessage("Overlaying layers.")

    overlayedLayers = 'Overlay'

    arcpy.locref.OverlayEvents('roadNetwork', list(features.keys()), overlayedLayers,
'INCLUDE_GEOMETRY')



    # Previously needed to drop fields with multiple instances, now ArcPro3.3 overlays fields with
identical elements

    #   (such as from_date field from each overlayed table, if identical, now one field output instead of 5)

    #   This update removed the need to drop excess fields and do mass renaming. The following
renaming is for convenience.

    arcpy.management.AlterField(overlayedLayers, "route_id", "RouteID")

    arcpy.management.AlterField(overlayedLayers, "from_measure", "FromMeasure")

    arcpy.management.AlterField(overlayedLayers, "to_measure", "ToMeasure")

    arcpy.management.AlterField(overlayedLayers, "from_date", "FromDate")

    arcpy.management.AlterField(overlayedLayers, "to_date", "ToDate")



    arcpy.AddMessage("Creating {0} feature class.".format("RoadNetwork") )


    arcpy.conversion.ExportFeatures(overlayedLayers,
```

```
                    os.path.join(project.defaultGeodatabase, "RoadNetwork"))


    arcpy.AddMessage("Creating {0} feature class.".format("FederalAidSystem") )
    federalAidSystemQuery = """(FacilityType <= 4 OR FacilityType IS NULL) AND
        (FunctionalClass < 6 OR (FunctionalClass = 6 AND (FHWAUrbanCode IS NOT NULL AND
FHWAUrbanCode < 99999)))"""

    arcpy.conversion.ExportFeatures(os.path.join(project.defaultGeodatabase, "RoadNetwork"),
                    os.path.join(project.defaultGeodatabase, "FederalAidSystem"),
                    federalAidSystemQuery)



def create_counts_csv():
    """
    Creates a csv file of counts with the station book id and the count date.

    Returns:
      Does not return a value. Writes data to the output directory.
    """

    import oracledb
    import pandas as pd
```

#2015 is chosen arbitrarily, the Off-System AADT Estimation Toolbox will filter out to the most recent 5 years.

```
query = """SELECT
s.roadnamestr3,
EXTRACT(YEAR FROM p.datadttm),
ROUND(p.data, 0)
FROM pdb_stat p
JOIN pdb_site s
ON p.siteid=s.siteid
WHERE p.usageid=10941
AND p.summing=3
AND p.stat=103
AND EXTRACT(YEAR FROM p.datadttm) > 2015
AND s.roadnamestr3 != '-----'
AND s.roadnamestr3 != '----------------------------------'
"""
# Connect to ITD database
connection = oracledb.connect(user=username, password=password, dsn="HDTPRODAWS")
cursor = connection.cursor()
cursor.prepare(query)
rows = cursor.execute(query).fetchall()

# Extract data
df = pd.DataFrame(rows)
df.columns = ["STATION_BOOK_ID", "COUNT_DATE", "AADT_COUNT"]
```

```
    # Write data to output directory
    df.to_csv(os.path.join(processing_dir, "StationCounts.csv"), index= False)



def project_counties():
    """
    Reprojects tiger line counties into the 8826 EPSG coordinate reference system

    Returns:
        Does not return a value. Writes projected counties to the output directory.
    """
    id_epsg = 8826
    counties_shp = os.path.join(processing_dir, "tl_2023_us_county.shp")

    counties = gpd.read_file(counties_shp)
    counties = counties.to_crs(id_epsg)
    counties.to_file(counties_shp)



def run():
    """
    Runs the other functions in the script.

    Returns:
```

```
    Does not return a value. Writes data to the output directory.
    """

    arcpy.AddMessage("Creating Road Network feature class and the Federal-Aid System feature class in the project geodatabase.")
    create_overlay_network()

    arcpy.AddMessage("Creating LEHD WAC by Block in the processing folder.")
    LEHD_WAC_Block()

    arcpy.AddMessage("Creating Counts CSV in the processing folder.")
    create_counts_csv()

    arcpy.AddMessage("Reprojecting Counties feature class to EPSG:8826.")
    project_counties()



if __name__ == "__main__":

    input_dir = arcpy.GetParameterAsText(0)
    processing_dir = arcpy.GetParameterAsText(1)
    username = arcpy.GetParameterAsText(2)
    password = arcpy.GetParameterAsText(3)
```

```
    run()
```

## aadt_count_join.py

```
"""
Tool to join latest AADT counts to count stations.

Author: High Street Consulting Group
Date: February 2025
"""
import arcpy
from sys import argv
from datetime import datetime, timedelta




def AADTCountJoin(Station_Book, Year, AADT_Counts, Stations_with_Counts):
    """
    Joins the most recent count within five years to its respective count station.

    Args:
        Station_Book: Spatial data for the count stations.
        AADT_Counts: Tabular data with historical counts, generated by A1. Create Input Data / create_input_data.py
```

```
    Stations_with_Counts: Output destination for the joined data.

  Returns:

    Does not return a value. Writes the joined data to the user-specified location.

  """

  # Data Preparation
##############################################################################

  # Copy the input count stations to a temporary feature class.

  arcpy.management.CopyFeatures(in_features=Station_Book,

                                out_feature_class=Stations_with_Counts)

  # Create a new in-memory table for the latest AADT counts.

  latest_counts_table = "memory\\Latest_AADT_Counts"

  arcpy.management.CreateTable("memory", "Latest_AADT_Counts")

  arcpy.management.AddField(latest_counts_table, "STATION_BOOK_ID", "TEXT")

  arcpy.management.AddField(latest_counts_table, "COUNT_DATE", "DATE")

  arcpy.management.AddField(latest_counts_table, "AADT_COUNT", "DOUBLE")

  # Data Processing
##############################################################################

  # Calculate date threshold for the last five years

  five_years_ago = int(year) - 5
```

```
# Create a filtered table for recent AADT counts within the last five years
temp_table = "memory\\Filtered_AADT_Counts"
arcpy.management.CopyRows(
   in_rows=AADT_Counts,
   out_table = temp_table
)

arcpy.AddMessage("Filtering counts by date...")
# Remove rows with COUNT_DATE older than five years
with arcpy.da.UpdateCursor(temp_table, ["COUNT_DATE"]) as cursor:
   for row in cursor:
      if row[0] is None or row[0] < five_years_ago or row[0] > int(year):
         cursor.deleteRow()



# Create a dictionary to store the latest AADT count for each STATION_BOOK_ID.
most_recent_counts = {}
with arcpy.da.SearchCursor(temp_table, ["STATION_BOOK_ID", "COUNT_DATE", "AADT_COUNT"],
                           sql_clause=(None, 'ORDER BY STATION_BOOK_ID, COUNT_DATE DESC')) as cursor:
   for station_id, count_date, aadt_count in cursor:
      if station_id not in most_recent_counts:
         most_recent_counts[station_id] = (count_date, aadt_count)
```

```
    # Insert the latest counts into the new table.
    with arcpy.da.InsertCursor(latest_counts_table, ["STATION_BOOK_ID", "COUNT_DATE",
"AADT_COUNT"]) as insert_cursor:
        for station_id, (count_date, aadt_count) in most_recent_counts.items():
            insert_cursor.insertRow((station_id, count_date, aadt_count))



    # Joining
##############################################################################
    arcpy.AddMessage("Joining counts to stations....")


    # Join the filtered latest counts to the station_book dataset
    arcpy.management.JoinField(Stations_with_Counts, "StationID", latest_counts_table,
"STATION_BOOK_ID", ["AADT_COUNT"], )


    # Remove rows from Stations_with_Counts where AADT_COUNT is null.
    with arcpy.da.UpdateCursor(Stations_with_Counts, ["AADT_COUNT"]) as cursor:
        for row in cursor:
            if row[0] is None or row[0] == 0:
                cursor.deleteRow()


    arcpy.management.AddField(Stations_with_Counts, "AADT_LOG", "FLOAT")
    arcpy.management.CalculateField(in_table=Stations_with_Counts,
                                    field="AADT_LOG",
                                    expression=f"math.log(!AADT_COUNT!)",
                                    code_block="import math")
```

```
if __name__ == '__main__':


    # Geolocated count stations

    station_book = arcpy.GetParameter(0)

    # Year being considered / maximum year

    year = arcpy.GetParameter(1)

    # Table of historical AADT counts

    aadt_counts = arcpy.GetParameter(2)

    # User-specified destination for the joined points

    out_points = arcpy.GetParameterAsText(3)


    # Join most recent AADT value to stations

    AADTCountJoin(station_book, year, aadt_counts, out_points)
```

## geo_field_norm_raster_multiple.py

```
"""

Tool to create rasters from polygon where values are pulled from and normalized by feature fields.


Author: High Street Consulting Group

Date: February 2025

"""

import arcpy

import re  # For sanitizing raster names
```

```
def sanitize_name(name):
    """
    Cleans user-provided names for rasters to remove invalid characters for geodatabase.

    Args:
        name: string to evaluate

    Returns:
        Returns a valid name for the raster.
    """

    # Replace invalid characters with underscores
    name = re.sub(r"[^\w]", "_", name)
    # Ensure the name doesn't start with a number or underscore
    if name[0].isdigit() or name[0] == "_":
        name = f"r{name}"
    # Trim to 50 characters (max length for geodatabase names)
    return name[:50]



def GeoFieldNorm(Geos, Primary_Field_Name, Norm_Field_Name, Log_Transform,
Raster_Geo_Field_Normalized):
```

```
    """
    Creates a raster of normalized values from a polygon feature layer.

    Args:
        Geos: Polygons containing the primary and normalizing fields.
        Primary_Field_Name: The field name from Geos for the primary field of the raster, e.g., the numerator.
        Norm_Field_Name: The field name from Geos for the normalizing field of the raster, e.g., the denominator.
        Log_Transform: A boolean indicator of whether to apply a log transform before creating the raster.
        Raster_Geo_Field_Normalized: Output destination for the raster.

    Returns:
        Returns a boolean for the loop to verify completion. Writes a raster to the output location.
    """

    # Data Preparation
################################################################################

    # Set up temporary layers
    arcpy.AddMessage("Copying feature layer...")
    Output_Data = "memory\\output_data"
    arcpy.management.CopyFeatures(
        in_features=Geos,
        out_feature_class=Output_Data
    )
```

```
arcpy.AddMessage("Adding normalized field...")
arcpy.management.AddField(
    in_table=Output_Data,
    field_name="density",
    field_type="FLOAT"
)

arcpy.AddMessage("Calculating normalized field...")
arcpy.management.CalculateField(
    in_table=Output_Data,
    field="density",
    expression=f"!{Primary_Field_Name}! / !{Norm_Field_Name}!",
    code_block="import math"
)



# Calculate cell size based on underlying layer's units.
# Define cell size based on TAZs linear units
spatial_ref = arcpy.Describe(Geos).spatialReference
linear_unit = spatial_ref.linearUnitName if spatial_ref.linearUnitName else "Unknown"

arcpy.AddMessage('Linear Spatial Reference Unit is...')
arcpy.AddMessage(spatial_ref.linearUnitName)
```

```
    # Define cell size based on unit
    if linear_unit == "Meter":
        cell_size = 1000  # 1 km in meters
    elif linear_unit == "Foot":
        cell_size = 3280.84  # Approx. 1 km in feet
    elif linear_unit == "Kilometer":
        cell_size = 1  # 1 km
    elif linear_unit == "Mile":
        cell_size = 0.621371  # Approx. 1 km in miles
    elif linear_unit in ["Degree", "Decimal Degree"]:
        cell_size = 0.01  # Example cell size in degrees
    else:
        arcpy.AddWarning("Unrecognized linear unit; using default cell size of 1000 meters.")
        cell_size = 1000  # Default fallback cell size in meters




    # Transformation
################################################################################


    # Create and rasterize the transformed field
    # Name: Output Raster
    output_raster = "memory\\output_raster"
```

```
    if Log_Transform is True:

        # Add Field for Log Transform
        arcpy.AddMessage("Adding log field...")
        arcpy.management.AddField(
            in_table=Output_Data,
            field_name="density_log",
            field_type="FLOAT"
        )

        # Calculate Field for Log Transform
        arcpy.AddMessage("Calculating log field...")
        arcpy.management.CalculateField(
            in_table=Output_Data,
            field="density_log",
            expression="math.log(!density!) if !Popden! > 0 else math.log({min_greater_than_zero})",
            code_block="import math",
            field_type="FLOAT"
        )

        # Polygon to Raster: Log Density
        arcpy.AddMessage("Rasterizing polygon...")
        min_greater_than_zero = min(row[0] for row in arcpy.da.SearchCursor(Output_Data, ["density"]) if
row[0] > 0)
        arcpy.conversion.PolygonToRaster(
```

```
            in_features=Output_Data,
            value_field="density_log",
            out_rasterdataset=output_raster,
            cellsize = cell_size * 5
        )



    else:
        # Polygon to Raster: Density
        arcpy.AddMessage("Rasterizing polygon...")
        arcpy.conversion.PolygonToRaster(
            in_features=Output_Data,
            value_field="density",
            out_rasterdataset=output_raster,
            cellsize = cell_size * 5
        )



    # Exporting rasters
################################################################################


    # Export the raster to the project's geodatabase.
    arcpy.AddMessage("Attempting to export raster...")


    # Get the current project
```

```
    aprx = arcpy.mp.ArcGISProject("CURRENT")
    project_gdb = aprx.defaultGeodatabase  # Path to the current project's geodatabase

    # Construct the full output path
    sanitized_name = sanitize_name(Raster_Geo_Field_Normalized)
    output_raster_path = f"{project_gdb}\\{sanitized_name}"

    arcpy.management.CopyRaster(
        in_raster=output_raster,
        out_rasterdataset=output_raster_path
    )

    arcpy.AddMessage(f"Raster exported successfully to: {output_raster_path}")

    return True



def main():
    """
    Iterates through each value table to create the normalized rasters.

    Returns:
        Does not return a value. Writes rasters to user-specified output locations.
```

```
    """

    # Get the ValueTable parameter
    value_table = arcpy.GetParameter(0)

    # Get the number of rows in the ValueTable
    row_count = value_table.rowCount

    if row_count == 0:
        arcpy.AddError("No parameter sets found in the value table")
        return

    # Iterate through the provided feature layers to create the rasters.
    for i in range(row_count):
        # Get values from the current row
        in_geo = value_table.getValue(i, 0)        # First column; polygon geometry to be rasterized
        primary_field = value_table.getValue(i, 1)  # Second column; field of interest for the final raster
        norm_field = value_table.getValue(i, 2)     # Third column; normalization field, e.g., polygon area for population density
        log_trans = value_table.getValue(i, 3)      # Fourth column; a boolean for whether values should be log transformed
        out_raster = value_table.getValue(i, 4)     # Fifth column; destination for the raster

        # remove special characters from the output raster name
        out_raster = out_raster.replace(" ", "_").replace("-", "_").replace(".", "_").replace(",", "_").replace("(", "_").replace(")", "_")
```

```
        arcpy.AddMessage(f"\nProcessing set {i + 1} of {row_count}:")
        arcpy.AddMessage(f"Input TAZ: {in_geo}")
        arcpy.AddMessage(f"Population Field: {primary_field}")
        arcpy.AddMessage(f"Area Field: {norm_field}")
        arcpy.AddMessage(f"Output Raster: {out_raster}")

        success = GeoFieldNorm(
            in_geo,
            primary_field,
            norm_field,
            log_trans,
            out_raster
        )

        if success:
            arcpy.AddMessage(f"Successfully processed data set {i + 1}")
        else:
            arcpy.AddMessage(f"Failed to process data set {i + 1}")



if __name__ == '__main__':
    main()
```

## network_density.py

"""

Tool to calculate the density of a road network's centerline miles.

Author: High Street Consulting Group

Date: February 2025

"""

```
import arcpy
from arcpy.sa import *
from sys import argv



def NetworkDensity(Complete_Road_Network, Search_Radius_Raw, Density_Raster):  # Network Density
    """
    Creates a raster where cell values represent the centerline length density of the selected features within a search radius.

    Args:
        Complete_Road_Network: The selected features to calculate the local density of.
        Search_Radius_Raw: The search radius in miles for the density calculation.
        Density_Raster: Output destination for the resulting raster.

    Returns:
```

```
    Does not return a value. Writes the raster to the output destination.
"""

# Set up the environment and temporary layers.
# Check out any necessary licenses.
arcpy.CheckOutExtension("spatial")

# Temporary layer for the copied network
Copied_Network = "memory\\Copied_Network"
arcpy.management.CopyFeatures(in_features=Complete_Road_Network,
                              out_feature_class=Copied_Network)

# Temporary raster for the network density
Line_Density = "memory\\raster_network_density"

# Set: Search Radius
if Search_Radius_Raw == None:
    Search_Radius_Raw = 5


# Calculate cell size based on geometry's linear units.
spatial_ref = arcpy.Describe(Complete_Road_Network).spatialReference
linear_unit = spatial_ref.linearUnitName if spatial_ref.linearUnitName else "Unknown"

# Define cell size based on unit
```

```
    if linear_unit == "Meter":
        cell_size = 1609.34  # 1 mi in meters
    elif linear_unit == "Foot":
        cell_size = 5280  # Approx. 1 mi in feet
    elif linear_unit == "Kilometer":
        cell_size = 1.60934  # 1 mi
    elif linear_unit == "Mile":
        cell_size = 1
    elif linear_unit in ["Degree", "Decimal Degree"]:
        cell_size = 0.0145  # Example cell size in degrees (1 degree latitude ~=~ 69 miles)
    else:
        arcpy.AddWarning("Unrecognized linear unit; using default cell size of 1 mi")
        cell_size = 1609.34  # Default fallback cell size in meters



    # Create and save the raster of network density given the search radius.
    Search_Radius = int(Search_Radius_Raw) * cell_size


    # Calculate and save the density raster.
    Line_Density = arcpy.sa.LineDensity(Copied_Network, "NONE", cell_size, Search_Radius,
"SQUARE_MILES")


    Line_Density.save(Density_Raster)
```

```
if __name__ == '__main__':
    # The entire road network geometry, no additional fields required.
    complete_road_network = arcpy.GetParameter(0)
    # The search radius for the density calculation in kilometers.
    searchRadius = arcpy.GetParameterAsText(1)
    # The user-specific output location for the density raster.
    outRaster = arcpy.GetParameterAsText(2)

    # Calculate the centerline density of the selected network within the search radius.
    NetworkDensity(complete_road_network, searchRadius, outRaster)
```

## vmt_from_aadt.py

```
"""
Tool to estimate annual VMT Federal-Aid roadways in Idaho.
The resulting total annual vehicle miles traveled (VMT) is subtracted from the Statewide annual VMT Estimate.
The difference is then used as an input for AADT scaling on off-system facilities.

Author: High Street Consulting Group
Date: February 2025
"""

import arcpy
import math
from arcpy.sa import *
```

```
from sys import argv



def VMTAggregation(FAS_Segments, AADT_Field, Counties, GEOID_Field, VMT_Table):
    """
    Summarize annual Vehicle Miles Traveled (VMT) on the Federal-Aid System (FAS)

    Args:
        FAS_Segments: Federal-Aid System Segments with Average Annual Daily Traffic (AADT) volume values.
        AADT_Field: The field name in the FAS_Segments feature layer that contains AADT values.
        Counties: The most recent Idaho TIGERLine counties.
        GEOID_Field: The field name in the Counties feature layer that contains the 5-digit FIPS code.
        VMT_Table: The output destination for the annual VMT summed by county.

    Returns:
        Does not return a value. Include a message with total Federal-Aid System annual VMT and writes a file to the output destination including totals by County FIPS code.
    """

    # Data Preparation
################################################################################
    arcpy.env.overwriteOutput = True

    # Creating Temporary Layers
```

```
# Layer for AADT so the original is not modified
aadt_layer = "AADT_Layer"
# Layer for counties so the original is not modified
county_layer = "County_Layer"
# Layer for AADT segments with County FIPS information
joined_layer = "Joined_Layer"

# Delete existing layers if they exist
for layer in [aadt_layer, county_layer, joined_layer, VMT_Table]:
    if arcpy.Exists(layer):
        arcpy.AddMessage(f"Deleting existing output: {layer}")
        arcpy.management.Delete(layer)

# Copying the AADT features to the temporary layer
arcpy.management.CopyFeatures(in_features = FAS_Segments,
                              out_feature_class = aadt_layer)
# Copying Idaho county features to the temporary layer
arcpy.management.MakeFeatureLayer(in_features = Counties,
                              out_layer = county_layer,
                              where_clause = f"SUBSTRING({GEOID_Field}, 1, 2) = '16'")



# Calculate annual Vehicle Miles Traveled on Federal-Aid facilities (AADT * segment length in miles).
arcpy.AddMessage("Calculating segment length...")
```

```
    arcpy.management.CalculateGeometryAttributes(aadt_layer,
                                    geometry_property = [["Segment_Length", "LENGTH_GEODESIC"]],
                                    length_unit = "MILES_US")[0]


    arcpy.AddMessage("Calculating segment VMT...")
    arcpy.management.AddField(
        in_table = aadt_layer,
        field_name = "VMT",
        field_type = "FLOAT"
    )
    arcpy.management.CalculateField(
        in_table = aadt_layer,
        field = "VMT",
        expression = f"(!{AADT_Field}! or 0) * (!Segment_Length! or 0) * 365.0",
        expression_type = "PYTHON3"
    )




    arcpy.AddMessage("Calculating estimated Statewide Federal-Aid System VMT...")


    # Calculate Statewide FAS annual VMT
########################################################################
    # Initialize VMT accumulator
    federal_aid_vmt = 0.0
```

```
    # Safely sum the annual VMT field, handling None and invalid values
    with arcpy.da.SearchCursor(aadt_layer, ["VMT"]) as cursor:
        for row in cursor:
            if row[0] is not None and isinstance(row[0], (int, float)):  # Check for valid numeric values
                federal_aid_vmt += row[0]
            else:
                arcpy.AddWarning(f"Invalid VMT value encountered: {row[0]}")



    arcpy.AddMessage("Federal-AidStatewide Annual VMT: " + str(round(federal_aid_vmt, 0)))



    # Joining Data
################################################################################


    # Perform spatial join to add county GEOID to AADT layer
    arcpy.AddMessage("Joining AADT layer with county layer...")
    arcpy.analysis.SpatialJoin(
        target_features=aadt_layer,
        join_features=county_layer,
        out_feature_class=joined_layer,
        join_type="KEEP_ALL",
        match_option="LARGEST_OVERLAP"
    )
```

```
    # Keep only specified fields
    arcpy.AddMessage("Keeping specified fields...")
    fields_to_keep = ["OBJECTIVEID", "RouteID", "FromMeasure", "ToMeasure", AADT_Field,
GEOID_Field, "VMT"]
    fields_to_delete = [f.name for f in arcpy.ListFields(joined_layer) if f.name not in fields_to_keep and
not f.required]
    arcpy.management.DeleteField(joined_layer, fields_to_delete)



    # Writing out the final file
########################################################################
    arcpy.AddMessage("Preparing summary table...")
    # Check if the output already exists and delete if necessary
    if arcpy.Exists(VMT_Table):
        arcpy.AddMessage(f"Deleting existing output: {VMT_Table}")
        arcpy.management.Delete(VMT_Table)



    # Convert the feature class to a table
    arcpy.AddMessage("Converting feature class to a table...")
    table_output = "memory\\AADT_Table"
    arcpy.conversion.TableToTable(
        in_rows=joined_layer,
        out_path="memory",
```

```
        out_name="AADT_Table"
    )

    # Export the table to CSV
    arcpy.AddMessage("Exporting table...")
    arcpy.conversion.ExportTable(
        in_table = table_output,
        out_table = VMT_Table
    )

    arcpy.AddMessage(f"File saved to: {VMT_Table}")




if __name__ == '__main__':
    # Segments with existing AADT estimates on Federal-Aid facilities
    fas_segments = arcpy.GetParameter(0)
    # Field containing the AADT estimate from the segments layer
    aadt_field = arcpy.GetParameterAsText(1)
    # Idaho Counties
    counties = arcpy.GetParameterAsText(2)
    # Field containing the 5-digit State-County FIPS code
    geoid_field = arcpy.GetParameterAsText(3)
    # Destination for the County-Level VMT Table
```

```
vmt_table = arcpy.GetParameterAsText(4)



# Sum annual VMT on the Federal-Aid System
VMTAggregation(fas_segments,
               aadt_field,
               counties,
               geoid_field,
               vmt_table)
```

## aggregated_aadt_estimation.py

```
"""
Tool to estimate AADT on off-system roadways in Idaho.

Author: High Street Consulting Group
Date: February 2025
"""
import arcpy
import math
import pandas as pd
from arcpy.sa import *
from sys import argv
```

```
def AggregatedAADTEstimation(Count_Stations, Network_of_Interest,
                             Independent_Variable_Rasters,
                             Out_Segments):
    """
    Creates Average Annual Daily Traffic (AADT) volume estimates along a network of interest given existing counts and a set of independent
    variable raster that are inputted into an Empirical Bayesian Kriging regression analysis.
    The resulting raster of spatially interpolated AADT is sampled along each segment, and the median value is rejoined to a copy of the network of interest.

    Args:
        Count_Stations: Count stations that will be filtered to those near the network of interest as the base points for Kriging interpolation
        Network_of_Interest: Segments that AADT estimates will be created for
        Independent_Variable_Raster: A list of rasters that will be primary inputs for the EBK regression analysis
        Out_Segments: The output destination for the segments with raw AADT estimates (e.g., not adjusted to match an expected total value)

    Returns:
        Does not return a value. Writes a feature layer that contains segments of the network of interest with raw AADT estimates.
    """

    # Set environment settings
    arcpy.env.overwriteOutput = True
```

```
    # To allow overwriting outputs change overwriteOutput option to True.
    arcpy.env.overwriteOutput = True

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

    arcpy.ImportToolbox(r"c:\program files\arcgis\pro\Resources\ArcToolbox\toolboxes\Data
Management Tools.tbx")
    arcpy.ImportToolbox(r"c:\program files\arcgis\pro\Resources\ArcToolbox\toolboxes\GeoAnalytics
Desktop Tools.tbx")

    # Data Preparation
################################################################################

    ## Create feature layers from input datasets
    AADT_Estimate_Raster = "memory\\AADT_Estimate_Raster"
    count_stations_layer = "memory\\Count_Stations_Layer"
    NOI_EBKRegressionPrediction = "NOI_EBKRegressionPrediction"
    network_layer = "memory\\Network_of_Interest_Layer"
    NOI_Sample_Points = "memory\\PointsAlongLines"
```

```
summarized_table = "memory\\summarized_table"
temporary_points = "memory\\EstimatePoints"

for item_name in [AADT_Estimate_Raster, count_stations_layer, NOI_EBKRegressionPrediction,
                  network_layer, NOI_Sample_Points, summarized_table, temporary_points]:
    if arcpy.Exists(item_name):
        # arcpy.AddMessage(f"Deleting existing output: {item_name}")
        arcpy.management.Delete(item_name)



## Expressions
# expression_federal_aid_system = "(FacilityType <= 4 OR FacilityType IS NULL) AND (FunctionalClass < 6 OR (FunctionalClass = 6 AND (FHWAUrbanCode < 99999 AND FHWAUrbanCode IS NOT NULL)))"
expression_off_system = "(FacilityType IS NULL Or (FacilityType IN (1, 2) And (FunctionalClass IS NULL Or (FunctionalClass = 6 And (FHWAUrbanCode = 99999 Or FHWAUrbanCode IS NULL)))))"
# expression_paved = "SurfaceType IN (3, 4) Or (SurfaceType IS NULL And (FunctionalClass IN (1, 2, 3, 4, 5, 6) OR FHWAUrbanCode < 99999 AND FHWAUrbanCode IS NOT NULL))"
# expression_unpaved = "SurfaceType IN (1, 2, 5, 6) Or (SurfaceType IS NULL And (FunctionalClass = 7 Or FunctionalClass IS NULL) And (FHWAUrbanCode = 99999 Or FHWAUrbanCode IS NULL))"



## Base Network
arcpy.management.CopyFeatures(in_features = Network_of_Interest,
                              out_feature_class = network_layer)
arcpy.management.DeleteField(in_table = network_layer,
                             drop_field = ["RouteID", "FromMeasure", "ToMeasure",
```

```
                    "FacilityType", "FunctionalClass", "SurfaceType",
                    "FHWAUrbanCode"],
                method = "KEEP_FIELDS")


    # Selects features in a layer based on a SQL query that evaluates whether the segment is ...
    # - A valid public segment (Facility Type 1, 2, or NULL)
    # - Has a Functional Class of 7 or NULL (NULL assumed to be a Local Road, e.g., Functional Class 7)
    # - Has a Functional Class of 6 in a non-urban area (FHWAUrbanCode = 99999 or FHWAUrbanCode is NULL)
    NOI = arcpy.management.SelectLayerByAttribute(in_layer_or_view=network_layer,
                                    selection_type = "NEW_SELECTION",
                                    where_clause=expression_off_system)


    ## Count Stations
    arcpy.management.CopyFeatures(in_features = Count_Stations,
                        out_feature_class = count_stations_layer)


    # Select count station locations that are within 5 Feet of the NOI segments identified above.
    NOI_Stations = arcpy.management.SelectLayerByLocation(in_layer=count_stations_layer,
                                    overlap_type="WITHIN_A_DISTANCE",
                                    select_features=NOI,
                                    search_distance="5 Feet")
```

```
    # Spatial Interpolation
##########################################################################

    # Run the Empirical Bayesian Kriging (EBK) regression model to estimate AADT values statewide for
    # the NOI segments identified above.

    arcpy.AddMessage("Using EBK Regression to model AADT values...")
    Output_diagnostic_feature_class = ""
    arcpy.ga.EBKRegressionPrediction(in_features=NOI_Stations,
                                     dependent_field="AADT_LOG",
                                     in_explanatory_rasters=Independent_Variable_Rasters,
                                     out_ga_layer=NOI_EBKRegressionPrediction,
                                     out_raster=AADT_Estimate_Raster,
                                     out_diagnostic_feature_class=Output_diagnostic_feature_class)

    # Ensure raster is in the correct format for extraction.
    AADT_Estimate_Raster = arcpy.Raster(AADT_Estimate_Raster)
```

```
   # Extract Values
################################################################################


   # Generate points along the selected segments to be used in extracting values from the AADT raster.

   # The "1 Miles" distance parameter is a hard-coded value that can be adjusted by the analyst to refine
performance.

   # If a segment is shorter than the distance specified it, it will only generate points at the segment's
endpoints.

   arcpy.management.GeneratePointsAlongLines(Input_Features=NOI,

                                             Output_Feature_Class=NOI_Sample_Points,

                                             Point_Placement="DISTANCE",

                                             Distance="1 Miles",

                                             Include_End_Points = "END_POINTS")




   # Extract AADT values to the points generated above.

   arcpy.AddMessage("Extracting values...")

   arcpy.sa.ExtractValuesToPoints(NOI_Sample_Points, AADT_Estimate_Raster,

                                  temporary_points,

                                  "NONE", "VALUE_ONLY")
```

```
    # Summarize Values
################################################################################


    # Create a composite key using the Route ID, From/To Measure, FacilityType, SurfaceType,
FHWAUrbanCode, and FunctionalClass fields.

    # This key will be used to summarize the extracted AADT values and later join them back to the
network.


    arcpy.AddMessage("Creating composite key...")

    arcpy.management.AddField(temporary_points, "CompositeKey", "TEXT")


    # Check that all the required fields are present before creating the composite key.

    required_fields = ["RouteID", "FromMeasure", "ToMeasure", "FacilityType", "SurfaceType",
"FHWAUrbanCode", "FunctionalClass"]

    missing_fields = [field for field in required_fields if field not in [f.name for f in
arcpy.ListFields(temporary_points)]]


    if missing_fields:

        arcpy.AddError(f"The following required fields are missing in 'temporary_points': {',
'.join(missing_fields)}")

    else:

        # Calculate CompositeKey with a fallback for missing or invalid field values

        arcpy.management.CalculateField(

            temporary_points,

            "CompositeKey",

            """'_'.join([str(!RouteID! or '').strip(), str(!FromMeasure! or '').strip(),

                    str(!ToMeasure! or '').strip(),  str(!FacilityType! or '').strip(),
```

```
                str(!SurfaceType! or '').strip(), str(!FHWAUrbanCode! or '').strip(),
                str(!FunctionalClass! or '').strip()])""",
        "PYTHON3"
    )



# Determines the median estimated AADT value for each composite key, converting back and forth to NumPy to do so.
arcpy.AddMessage("Aggregating values...")

df = pd.DataFrame(arcpy.da.FeatureClassToNumPyArray(
    in_table = temporary_points,
    field_names = ("CompositeKey","RASTERVALU")
))

df_grouped = df.groupby("CompositeKey", as_index=False).agg({"RASTERVALU": 'median'})

structured_array = df_grouped.to_records(
    index = False,
    column_dtypes = {'CompositeKey': "<U255", "RASTERVALU": "f8"}
)

arcpy.da.NumPyArrayToTable(structured_array,
                summarized_table)
```

```
    # Transform Values
################################################################################


    # Exponentiate values (reverse natural log) and replace nulls with 0.

    # Uses a helper function to ensure the exponentiation process doesn't return an error.


    arcpy.AddMessage("Exponentiating values...")


    arcpy.management.AddField(summarized_table, "EXP_AADT", "FLOAT")


    arcpy.management.CalculateField(
        in_table=summarized_table,
        field="EXP_AADT",
        expression="safe_exponentiate(!RASTERVALU!)",
        code_block="""def safe_exponentiate(value):
        if value is None or value <= -700:  # Avoid underflow
            return 0
        try:
            return math.exp(value)
        except OverflowError:
            return float('inf')"""
```

```
    )


    # Join Values
################################################################################################


    # Join the summarized AADT values back to the original non-urban network of interest.

    # First creates the same composite key as developed above, and then joins the raw outputs of the EBK regression model

    # and the exponentiated values to the NOI segments.


    arcpy.AddMessage("Joining values...")


    arcpy.management.AddField(NOI, "CompositeKey", "TEXT")


    arcpy.management.CalculateField(
        NOI,
        "CompositeKey",
        """'_'.join([str(!RouteID! or '').strip(), str(!FromMeasure! or '').strip(),
                    str(!ToMeasure! or '').strip(), str(!FacilityType! or '').strip(),
                    str(!SurfaceType! or '').strip(), str(!FHWAUrbanCode! or '').strip(),
                    str(!FunctionalClass! or '').strip()])""",
        "PYTHON3"
    )
```

```
    arcpy.management.JoinField(
        in_data=NOI,
        in_field="CompositeKey",
        join_table=summarized_table,
        join_field="CompositeKey",
        fields=["RASTERVALU","EXP_AADT"]
    )


    arcpy.management.DeleteField(in_table = NOI,
                        drop_field = ["RouteID", "FromMeasure", "ToMeasure",
                                "FacilityType", "FunctionalClass", "SurfaceType",
                                "FHWAUrbanCode", "RASTERVALU", "EXP_AADT"],
                        method = "KEEP_FIELDS")




    # Export Data
################################################################################


    # Exporting segments with modeled AADT values to the user-specified location.
    arcpy.AddMessage("Writing segment data...")


    if arcpy.Exists(Out_Segments):
        #arcpy.AddMessage(f"Deleting existing output: {Out_Segments}")
```

```
        arcpy.management.Delete(Out_Segments)


    arcpy.conversion.ExportFeatures(in_features = NOI,
                                    out_features = Out_Segments)




if __name__ == '__main__':
    # Network Aspects
    count_stations = arcpy.GetParameter(0)
    network_of_interest = arcpy.GetParameter(1)
    # Independent Variables (Rasters) to interpolate AADT
    independent_variable_rasters = arcpy.GetParameterAsText(2).split(';')
    # Output Destinationf for Segments with raw modeled AADT estimates
    out_segments = arcpy.GetParameterAsText(3)


    # Create AADT estimates from Kriging-based spatial interpolation on the network of interest
    AggregatedAADTEstimation(Count_Stations = count_stations,
                             Network_of_Interest = network_of_interest,
                             Independent_Variable_Rasters = independent_variable_rasters,
                             Out_Segments = out_segments)
```

## aadt_estimation_scaling.py

"""

Tool to scale estimated AADT on off-system roadways in Idaho to the Statewide annual VMT estimate.

Author: High Street Consulting Group

Date: February 2025

"""

```
import arcpy
import math
import numpy as np
from arcpy.sa import *
from sys import argv



def rescale_field_to_target(feature_layer, input_field, rescaled_field, new_vmt_field, length_field,
                            target_total_vmt, stations_with_aadt, aadt_field, tolerance=1000, max_iterations=10):
    """
    Rescales a field in an ArcGIS Pro feature layer to ensure that
    - maximum AADT is less than the 3rd quartile (Q3) of the observed AADT values plus the Interquartile Range (IQR) and
    - total estimated annual VMT is within the provided tolerance of the targeted value.
```

It first adjusts the maximum estimate, then scales the overall total, and (if necessary) adjusts the minimum value.

Args:

- feature_layer: Name of the feature layer.

- input_field: The field containing original values.

- rescaled_field: The field to store rescaled values.

- new_vmt_field: The field to store the rescaled values associated annual VMT.

- length_field: The field containing segment length.

- target_total_vmt: Target sum of rescaled annual VMTs.

- stations_with_aadt: Feature layer with AADT values based on actual counts.

- aadt_field: The field with AADTs based on actual counts from the stations_with_aadt layer.

- tolerance: Acceptable deviation from target_total_vmt.

- max_iterations: Maximum number of iterations to adjust new_min.

Returns:

Does not return a value but modifies a field of a designated feature layer.

"""

# Extract values from the field

est_values = [row[0] for row in arcpy.da.SearchCursor(feature_layer, [input_field]) if row[0] is not None]

actual_values = [row[0] for row in arcpy.da.SearchCursor(stations_with_aadt, [aadt_field]) if row[0] is not None]

```
    # Compute starting values

    # Estimated Values

    est_min_val = min(est_values)

    est_max_val_raw = max(est_values)


    # Actual Values

    actual_q1 = np.percentile(actual_values, 25)  # 25th percentile

    actual_q3 = np.percentile(actual_values, 75)  # 75th percentile

    actual_iqr = actual_q3 - actual_q1 # interquartile range

    actual_max_val_iqr = actual_q3 + actual_iqr # find a maximum for estimates that ignores outliers


    # Select the lower of the two maximum values

    max_val = min(est_max_val_raw, actual_max_val_iqr)


    # Report information

    arcpy.AddMessage(f"Stats for estimates in {input_field}: Min={est_min_val},
Max={est_max_val_raw}")

    arcpy.AddMessage(f"Stats for actual values in {aadt_field}: Q1={actual_q1}, Q3={actual_q3},
IQR={actual_max_val_iqr}")




    # Initializing new AADT field
#########################################################################

    arcpy.management.AddField(
```

```
        in_table = feature_layer,
        field_name = rescaled_field,
        field_type = "FLOAT"
    )
    # Reduce maximum estimate if over Q3 + IQR of observed AADT
    arcpy.management.CalculateField(feature_layer, rescaled_field, f"!{input_field}! * {max_val} /
{est_max_val_raw}", "PYTHON3")

    # Calculate new annual VMT field
    arcpy.AddMessage(f"Calculating {new_vmt_field}...")
    arcpy.management.AddField(
        in_table = feature_layer,
        field_name = new_vmt_field,
        field_type = "FLOAT"
    )
    arcpy.management.CalculateField(feature_layer, new_vmt_field,
                                    f"float(!{rescaled_field}! or 0.0) * float(!{length_field}! or 0.0) * 365.0",
                                    "PYTHON3")

    # Compute the new total sum
    arcpy.AddMessage("Summing new annual VMT...")
    new_total = 0.0

    # Safely sum the annual VMT field, handling None and invalid values.
    with arcpy.da.SearchCursor(feature_layer, [new_vmt_field]) as cursor:
```

```
        for row in cursor:
            if row[0] is not None and isinstance(row[0], (int, float)):  # Check for valid numeric values
                new_total += row[0]
            else:
                arcpy.AddWarning(f"Invalid annual VMT value encountered: {row[0]}")


    # Check if current annual VMT total is within tolerance
    arcpy.AddMessage(f"Initial annual VMT of {new_total} - {target_total_vmt} = {new_total - target_total_vmt} is less than tolerance of {tolerance}: {abs(new_total - target_total_vmt) <= tolerance}")


    # If the estimate is already within tolerance, move on
    if abs(new_total - target_total_vmt) <= tolerance:
        return




    # Scale AADT to target annual VMT
########################################################################
    arcpy.AddMessage("Iterating to adjust total annual estimated VMT...")
    iteration = 0
    while iteration < max_iterations:
        rescale_values = [row[0] for row in arcpy.da.SearchCursor(feature_layer, [rescaled_field]) if row[0] is not None]
```

```
    rescale_min_val = min(rescale_values)
    rescale_max_val = max(rescale_values)
    arcpy.AddMessage(f"For rescaled values: Min={rescale_min_val}, Max={rescale_max_val}")

    if new_total > target_total_vmt:
        arcpy.AddMessage("New total larger than target value, scaling down...")
        arcpy.management.CalculateField(feature_layer, rescaled_field,
                                        f"!{rescaled_field}! * {target_total_vmt} / {new_total}",
                                        "PYTHON3")

    elif new_total < target_total_vmt and rescale_max_val < actual_max_val_iqr:
        arcpy.AddMessage("New total less than target value and maximum is less than Q3 + IQR, scaling
up maximum...")
        arcpy.management.CalculateField(feature_layer, rescaled_field,
                                        f"!{rescaled_field}! * {actual_max_val_iqr} / {rescale_max_val}",
                                        "PYTHON3")

    elif new_total < target_total_vmt and rescale_max_val >= actual_max_val_iqr:
        arcpy.AddMessage("New total less than target value but maximum is at Q3 + IQR, scaling up
minimum...")
        if rescale_min_val <= 0:
            min_val = 0.1
        else:
            min_val = rescale_min_val * 1.1

        arcpy.management.CalculateField(feature_layer, rescaled_field,
```

```
                        f"((float(!{rescaled_field}! or 0) - {min_val}) / ({rescale_max_val} - {min_val})) *
({rescale_max_val} - {min_val}) + {min_val}",
                        "PYTHON3")

    else:
        arcpy.AddMessage('Total annual VMT equals target value.')



    # Recalculate annual VMT
    arcpy.management.CalculateField(feature_layer, new_vmt_field,
                        f"float(!{rescaled_field}! or 0.0) * float(!{length_field}! or 0.0) * 365.0",
                        "PYTHON3")

    # Compute the new total sum
    arcpy.AddMessage("Summing new annual VMT...")
    new_total = 0.0

    # Safely sum the annual VMT field, handling None and invalid values.
    with arcpy.da.SearchCursor(feature_layer, [new_vmt_field]) as cursor:
        for row in cursor:
            if row[0] is not None and isinstance(row[0], (int, float)):  # Check for valid numeric values
                new_total += row[0]
            else:
                arcpy.AddWarning(f"Invalid annual VMT value encountered: {row[0]}")
```

```
        # Check if current annual VMT total is within tolerance

        arcpy.AddMessage(f"Initial annual VMT of {new_total} - {target_total_vmt} = {new_total - target_total_vmt} is less than tolerance of {tolerance}: {abs(new_total - target_total_vmt) <= tolerance}")


        # If the estimate is already within tolerance, move on

        if abs(new_total - target_total_vmt) <= tolerance:

            return


        iteration += 1


    arcpy.AddMessage(f"Reached max iterations ({max_iterations}). Final sum: {new_total}, target: {target_total_vmt}.")









def round_aadt(aadt):

    """

    Rounds numeric values according to ITD's existing process for rounding AADT.

    However, the current process has a minimum value of zero as it is applied to the Federal-Aid System.

    For off-system roads, it is possible (and even likely in especially remote areas) that a

    vehicle traverses the segment less than once every other day. Values less than 0.5 are rounded to zero.
```

```
Args:
    aadt: A numeric value for AADT that is to be rounded.

Returns:
    Returns a rounded value.
"""

# Check if the value is null or NA
if aadt is None or (isinstance(aadt, float) and math.isnan(aadt)):
    return 0
# Apply rounding thresholds
if aadt < 10:
    return round(aadt)
elif aadt < 1000:
    return round(aadt, -1)
elif aadt < 10000:
    return round(aadt, -2)
else:
    return round(aadt / 500) * 500
```

```
def AggregatedAADTScaling(Estimate_Segments, Est_Year, Target_VMT,
                          Stations_With_AADT, AADT_Field,
                           Counties, GEOID_Field,
                           Scaled_Segments, VMT_Summary_Table):
  """
  Scales and rounds raw Average Annual Daily Traffic (AADT) volumes according to a target total annual Vehicle Miles Traveled (VMT)
  and ITD's existing rounding rules.

  Args:
    Estimate_Segments: A feature layer of segments with raw AADT estimates generated from B1. Aggregated AADT Estimation / aggregated_aadt_estimation.py
    Target_VMT: A numeric value of the total annual VMT that the scaled AADT values should produce when multiplied by their segment length and 365.0 and then summed
    Counties: The most recent Idaho TIGERLine counties.
    GEOID_Field: The field name in the Counties feature layer that contains the 5-digit FIPS code.
    Scaled_Segments: The output destination for the segments from Estimate_Segments that contains scaled and rounded AADT values
    VMT_Summary_Table: The output destination for the Scaled_Segments' annual VMT summed by county.

  Returns:
    Does not return a value. Writes a feature layer containing scaled and rounded AADT estimates and a table summing those estimates by county.
  """
```

```
    # Set workspace options
    arcpy.env.overwriteOutput = True

    # Data Preparation
################################################################################

    # Setting up layers
    network_layer = "Network_of_Interest_Layer"
    county_layer = "County_Layer"
    joined_layer = "Joined_Layer"

    for layer in [network_layer, county_layer, joined_layer]:
        if arcpy.Exists(layer):
            #arcpy.AddMessage(f"Deleting existing output: {layer}")
            arcpy.management.Delete(layer)

    arcpy.management.CopyFeatures(in_features = Estimate_Segments,
                                  out_feature_class = network_layer)


    # Filters counties to only include Idaho.
    arcpy.management.MakeFeatureLayer(in_features = Counties,
                                      out_layer = county_layer,
                                      where_clause = f"SUBSTRING({GEOID_Field}, 1, 2) = '16'")
    num_counties = arcpy.management.GetCount(in_rows = county_layer)[0]
```

```
    # Joining counties to the AADT estimates to summarize annual VMT later.
    arcpy.AddMessage("Joining AADT layer with county layer...")
    arcpy.analysis.SpatialJoin(
        target_features=network_layer,
        join_features=county_layer,
        out_feature_class=joined_layer,
        join_type="KEEP_ALL",
        match_option="LARGEST_OVERLAP"
    )



    # Rescaling AADT ##############################################################################


    # Calculates new AADT through an iterative adjustment process contained in the
rescale_field_to_target function.
    arcpy.AddMessage("Calculating segment length...")
    arcpy.management.CalculateGeometryAttributes(joined_layer,
                                    geometry_property = [["Segment_Length", "LENGTH_GEODESIC"]],
                                    length_unit = "MILES_US")[0]



    arcpy.AddMessage("Rescaling AADT values....")
    rescale_field_to_target(feature_layer = joined_layer, input_field = "Exp_AADT",
                        rescaled_field = "Rescaled_AADT", new_vmt_field = "Rescaled_VMT", length_field =
"Segment_Length",
```

```
                target_total_vmt = float(Target_VMT), stations_with_aadt = Stations_With_AADT,
aadt_field = AADT_Field,
                tolerance=10000, max_iterations=10)



  # Rounding AADT
################################################################################
  arcpy.AddMessage("Rounding AADT values...")

  # Rounds AADT based on the thresholds in the round_aadt helper function.
  arcpy.management.AddField(joined_layer, "Rounded_AADT", "LONG")
  with arcpy.da.UpdateCursor(joined_layer, ["Rescaled_AADT", "Rounded_AADT"]) as cursor:
    for row in cursor:
      # Apply the round_aadt function directly
      row[1] = round_aadt(row[0])
      cursor.updateRow(row)

  arcpy.management.DeleteField(in_table = joined_layer,
                drop_field = ["RouteID", "FromMeasure", "ToMeasure",
                      "FacilityType", "FunctionalClass", "SurfaceType",
                      "FHWAUrbanCode", f"{GEOID_Field}", "Segment_Length",
                      "RASTERVALU", "EXP_AADT", "Rescaled_AADT", "Rescaled_VMT",
"Rounded_AADT"],
                method = "KEEP_FIELDS")

  arcpy.management.AddField(joined_layer, "YEAR", "LONG")
```

```
    arcpy.management.CalculateField(joined_layer, "YEAR", f"{Est_Year}")



    # Export Data
################################################################################################
    arcpy.AddMessage("Writing segments with scaled AADT values...")

    # Check if the output feature class already exists and delete if necessary.
    if arcpy.Exists(Scaled_Segments):
        #arcpy.AddMessage(f"Deleting existing output: {Scaled_Segments}")
        arcpy.management.Delete(Scaled_Segments)

    arcpy.conversion.ExportFeatures(in_features = joined_layer,
                                    out_features = Scaled_Segments)



    # Summarize annual VMT to the county level.
    arcpy.AddMessage("Creating a summary table of total VMT by county...")
    summary_fields = [["Rescaled_VMT", "SUM"]]
    arcpy.analysis.Statistics(
        in_table=joined_layer,
        out_table=VMT_Summary_Table,
        statistics_fields=summary_fields,
        case_field=GEOID_Field
    )
```

```
if __name__ == '__main__':
    # Segments with Raw Estimates from the "B1 Aggregated AADT EStimation" tool.
    estimate_segments = arcpy.GetParameter(0)
    # Year for which estimates are being made
    est_year = arcpy.GetParameter(1)
    # Target Total annual Vehicle Miles Traveled to scale the final output to.
    # Should equal the statewide estimate minus the annual VMT on the Federal-Aid System.
    target_vmt = arcpy.GetParameterAsText(2)
    # Stations with AADT values from the "A2. AADT Count Join" tool with identified AADT field.
    stations_with_aadt = arcpy.GetParameterAsText(3)
    aadt_field = arcpy.GetParameterAsText(4)
    # Idaho Counties with a field containing the 5-digit FIPS code
    counties = arcpy.GetParameterAsText(5)
    geoid_field = arcpy.GetParameterAsText(6)
    # Output Locations for the scaled segments and a summary table of total annual VMT by county.
    scaled_segments = arcpy.GetParameterAsText(7)
    vmt_summary_table = arcpy.GetParameterAsText(8)

    # Scale AADT Values to a provided target total annual VMT
    AggregatedAADTScaling(estimate_segments,
                          est_year,
                            target_vmt,
```

```
                stations_with_aadt,
                aadt_field,
                counties,
                geoid_field,
                scaled_segments,
                vmt_summary_table)
```

## error_evaluation.py

```
"""
Tool to compare AADT estimates to real-world counts and calculate the associate error metrics for the
model's estimates.

Author: High Street Consulting Group
Date: February 2025
"""

import arcpy
import math
import re  # For sanitizing raster names

def sanitize_name(name):
    """
    Cleans user-provided names for rasters to remove invalid characters for geodatabase.

    Args:
```

```
    name: string to evaluate

  Returns:

    Returns a valid name for the raster.

  """

  # Replace invalid characters with underscores

  name = re.sub(r"[^\w]", "_", name)

  # Ensure the name doesn't start with a number or underscore

  if name[0].isdigit() or name[0] == "_":

    name = f"r{name}"

  # Trim to 50 characters (max length for geodatabase names)

  return name[:50]




def AADTComparison(in_points, pt_id_field, pt_value_field,

                   in_lines, line_id_field, line_value_field,

                   search_distance, out_table, detailed_out_table):

  """

  Compares AADT values between local counts at points (e.g., count stations) and modeled estimates
along segments.

  Returns Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percent
Error (MAPE) via a table.


  Args:
```

in_points: Point feature layer containing sites with AADT values.

pt_id_field: Route ID field for the point feature layer.

pt_value_field: The field containing the AADT values at the count sites.

in_lines: Segment feature layer containing modeled / estimated AADT values.

line_id_field: Route ID field for the segment feature layer.

line_value_field: The field containing the modeled / estimates AADT values.

search_distance: A numeric value specifying the search distance in feet between count stations and the nearest segment with a matching route ID

out_table: The output destination for the table of error metrics.

detailed_out_table: the output destination for the table of error for each local count location.

Returns:

Does not return a value. Writes out a table of error metrics and includes those metrics in its details messages.

"""

```
# Create a temporary layer for the lines.
lines_layer = "temp_lines_layer"
if arcpy.Exists(lines_layer):
    arcpy.Delete_management(lines_layer)
arcpy.management.MakeFeatureLayer(in_lines, lines_layer)

points_layer = "temp_points_layer"
if arcpy.Exists(points_layer):
    arcpy.Delete_management(points_layer)
arcpy.management.MakeFeatureLayer(in_points, points_layer)
```

```
    # Select points within the search distance of the lines.
    arcpy.AddMessage("Selecting points within the search distance (" + search_distance + " Feet) of the lines...")
    in_points_selection = arcpy.management.SelectLayerByLocation(points_layer, "WITHIN_A_DISTANCE", lines_layer, str(search_distance) + " FeetInt", "NEW_SELECTION")

    if not arcpy.Exists(in_points_selection):
        raise RuntimeError("SelectLayerByLocation did not create a valid layer.")

    # Get the number of points for the progressor
    count_result = arcpy.management.GetCount(in_points_selection)

    arcpy.AddMessage(f"Feature count: {count_result.getOutput(0)}")

    point_count = int(count_result.getOutput(0))
    arcpy.SetProgressor("step", "Processing points...", 0, point_count, 1)

    # Prepare to iterate through points.
    errors = []        # List of errors (point value minus joined line value)
    denom_errors = []   # List of denominator values for MAPE (the joined line value; skip if 0)
    detailed_data = []  # List of all information by pairing with local counts

    # Get the number of points for the progressor
    count_result = arcpy.management.GetCount(in_points_selection)
```

```
point_count = int(count_result.getOutput(0))
arcpy.SetProgressor("step", "Processing points...", 0, point_count, 1)

arcpy.AddMessage("Comparing AADT values between points and lines...")
# Use a SearchCursor to iterate over each point.
point_fields = [ "SHAPE@", pt_value_field, pt_id_field ]
point_index = 0
with arcpy.da.SearchCursor(in_points_selection, point_fields) as pt_cursor:
    for pt_row in pt_cursor:
        pt_geom = pt_row[0]
        pt_value = pt_row[1]
        pt_id = pt_row[2]

        # arcpy.AddMessage("Processing point with ID {}...".format(pt_id))

        # Create a temporary feature (in-memory geometry) for this point.
        # Use a buffer around the point using the search distance.
        buffer_geom = "memory\\temp_buffer"
        arcpy.analysis.Buffer(in_features = pt_geom,
                              out_feature_class = buffer_geom,
                              buffer_distance_or_field = search_distance + " Feet")



        # Select lines that intersect the buffer.
```

```
        arcpy.SelectLayerByLocation_management(lines_layer, "INTERSECT", buffer_geom,
                                               search_distance, "NEW_SELECTION")

        # Now apply an attribute query so that only lines with matching identifier are considered.
        where_clause = "{} = '{}'".format(arcpy.AddFieldDelimiters(lines_layer, line_id_field), pt_id)
        arcpy.AddMessage(where_clause)
        lines_layer_selection = arcpy.SelectLayerByAttribute_management(lines_layer,
"SUBSET_SELECTION", where_clause)

        # Get the joined line values from the selected lines.
        line_values = []
        line_ids = []
        with arcpy.da.SearchCursor(lines_layer_selection, [line_id_field, line_value_field]) as line_cursor:
            for line_row in line_cursor:
                line_ids.append(line_row[0])
                line_values.append(line_row[1])

        # Clear selection for the next iteration.
        arcpy.SelectLayerByAttribute_management(lines_layer_selection, "CLEAR_SELECTION")

        if line_values:
            # If there are multiple lines, we take the average value.
            avg_line_value = sum(line_values) / float(len(line_values))
            segment_id = ", ".join(map(str, line_ids))
            error = pt_value - avg_line_value
```

```
            errors.append(error)
            if avg_line_value != 0:
                denom_errors.append(abs(avg_line_value))
            else:
                # For MAPE, if the joined line value is 0, we skip this point.
                arcpy.AddWarning(f"For point {pt_id}, the average line value is 0; skipping in MAPE
calculation.")
            detailed_data.append((pt_id, pt_value, segment_id, avg_line_value, error))
        else:
            # If no matching line is found, optionally warn and skip the point.
            arcpy.AddWarning(f"No matching line found for point {pt_id}. Point skipped.")
            continue

        # Update the progressor.
        point_index += 1
        arcpy.SetProgressorPosition(point_index)

    # Reset the progressor.
    arcpy.ResetProgressor()

    n = len(errors)
    if n == 0:
        raise arcpy.ExecuteError("No points had matching lines. Cannot compute error metrics.")

    arcpy.AddMessage("Comparison complete. Computing error metrics...")
```

```
# Compute error metrics.
# RMSE: sqrt(mean(square(error)))
rmse = math.sqrt(sum([e**2 for e in errors]) / n)
# MAE: mean(abs(error))
mae = sum([abs(e) for e in errors]) / n

# MAPE: mean( absolute error / |joined line value| ) * 100
# We only include points where the joined line value was nonzero.
if len(denom_errors) > 0:
    mape = (sum([abs(e)/d for e, d in zip(errors, denom_errors)]) / len(denom_errors)) * 100.0
else:
    mape = None
    arcpy.AddWarning("No valid denominator for MAPE (all joined line values were 0).")

arcpy.AddMessage("RMSE: {}".format(rmse))
arcpy.AddMessage("MAE: {}".format(mae))
if mape is not None:
    arcpy.AddMessage("MAPE: {}%".format(mape))
else:
    arcpy.AddMessage("MAPE: Not computed (no valid joined line values).")

# Create the output table and insert the metrics.
arcpy.AddMessage("Creating output table with error metrics...")
aprx = arcpy.mp.ArcGISProject("CURRENT")
project_gdb = aprx.defaultGeodatabase  # Path to the current project's geodatabase
```

```
    arcpy.AddMessage("Adding error metric fields to the output table...")
    sanitized_out_table = sanitize_name(out_table)
    error_statistic_table = arcpy.management.CreateTable(project_gdb,
sanitized_out_table).getOutput(0)
    # Add fields for the metrics
    arcpy.management.AddField(error_statistic_table, "RMSE", "DOUBLE")
    arcpy.management.AddField(error_statistic_table, "MAE", "DOUBLE")
    arcpy.management.AddField(error_statistic_table, "MAPE", "DOUBLE")

    # Insert one row with the error metrics
    with arcpy.da.InsertCursor(error_statistic_table, ["RMSE", "MAE", "MAPE"]) as icursor:
        icursor.insertRow((rmse, mae, mape if mape is not None else -1))

    arcpy.AddMessage(f"Output table with error metrics created: {error_statistic_table}")

    arcpy.AddMessage("Creating a table with the error for each count-estimate pair...")
    sanitized_detailed_table = sanitize_name(detailed_out_table)
    error_details_table = arcpy.management.CreateTable(project_gdb,
sanitized_detailed_table).getOutput(0)
    arcpy.management.AddField(error_details_table, "Point_ID", "TEXT")
    arcpy.management.AddField(error_details_table, "Point_AADT", "DOUBLE")
    arcpy.management.AddField(error_details_table, "Segment_ID", "TEXT")
    arcpy.management.AddField(error_details_table, "Segment_AADT", "DOUBLE")
    arcpy.management.AddField(error_details_table, "Error", "DOUBLE")
```

```
    with arcpy.da.InsertCursor(error_details_table, ["Point_ID", "Point_AADT", "Segment_ID",
"Segment_AADT", "Error"]) as icursor:
        for row in detailed_data:
            icursor.insertRow(row)

    arcpy.AddMessage(f"Output detailed comparison table created: {error_details_table}")




if __name__ == '__main__':
    # Points with local AADT counts
    local_counts = arcpy.GetParameter(0)
    # Points ID Field
    counts_id = arcpy.GetParameterAsText(1)
    # Points AADT Field
    counts_field = arcpy.GetParameterAsText(2)
    # Feature layer segments with AADT estimates
    aadt_estimates = arcpy.GetParameter(3)
    # Segment ID Field
    estimates_id = arcpy.GetParameterAsText(4)
    # Segment AADT Estimate Field
    estimates_field = arcpy.GetParameterAsText(5)
    # Search Distance
    search_distance = arcpy.GetParameterAsText(6)
```

```
    # Destination for the Estimate Error Statistics Table

    error_table = arcpy.GetParameter(7)

    # Destination for the Detailed Output Table

    detailed_out_table = arcpy.GetParameter(8)


    # Compare AADT values between a set of points (e.g., actual counts at stations) and segments (e.g., estimates for facilities)

    AADTComparison(in_points = local_counts, pt_id_field = counts_id, pt_value_field = counts_field,

                   in_lines = aadt_estimates, line_id_field = estimates_id, line_value_field = estimates_field,

                   search_distance = search_distance, out_table = error_table, detailed_out_table = detailed_out_table)
```