Estimating Active Transportation Annual Average Daily Traffic

Jacob Spertus, Philip Stark

December 18th, 2024

1 Background

1.1 What would we like to know?

Transportation agencies could use spatiotemporal information on bicycle and pedestrian traffic for a variety of purposes, including improving safety, planning infrastructure improvements, measuring the impact of infrastructure changes, and measuring and increasing mode shift from motor vehicles to active transportation (Bhomwick et al, 2024; NCHRP, 2024).

In an ideal world, such information might take the form of a map of the average density of different modes of transportation at every point in a managed network over any time period of interest. Unfortunately, it is impossible to conduct a genuine census of such information. Agencies instead rely on point data from count stations and, in some recent cases, spatio-temporal tracking data from location-based services (LBSs) like Strava Metro (Gupta et al, 2024; Ferster et al, 2021; Selala and Musakwa, 2016). Direct GPS tracing has also been used in some instances to augment motor-vehicle traffic data (e.g., McNally et al (2003)), but the privacy issues are substantial.

Here we consider the design and analysis of point data measured at a mix of continuous count station (CCSs) and short-term count stations (SCSs). CCSs are in place for a year or more and (disregarding measurement error and missing data) completely enumerate AT traffic at a location. SCSs are in place temporarily—typically hours to weeks. SCS counts are widely used because they are much cheaper to obtain than CCS counts.

This document focuses on estimating temporal averages of bicycle and pedestrian traffic at particular locations. In particular, we focus on estimating annual average daily traffic (AADT): the average number of cyclists and pedestrians who pass a particular point per day in a given year. If there is a CCS at a location where the agency wants to know the AADT, AADT can be *calculated* exactly (up to instrumental measurement error). When AADT is desired at a location with only an SCS or no station at all, it must instead be *estimated* using appropriate statistical methods. Along the way, we also discuss methods for estimating the (spatial) average of the AADTs (A-AADT) across some set of locations. For instance, the A-AADT could be the average AADT on all Class I bikeways in District 12. An A-AADT is proportional to the total volume of AT traffic across the set of locations that define it, and estimates of several A-AADTs may be useful in planning infrastructure or allocating budgets, among other tasks.

1.2 Data sources

Data on which to base estimates of AADT may come from fixed SCS or CCS counters (with various technologies including inductive loops, pneumatic tubes, infrared sensors, video processing, LIDAR, etc; see Nordback et al (2016)) or from LBSs (e.g., Selala and Musakwa (2016)).

1.2.1 A note on LBS Data

In principle, LBS data are richer than fixed counts because they track individuals through time and space, rather than simply recording the number of times a pedestrian or cyclist passes a point. Sufficiently accurate and complete LBS data could be used to estimate or infer the distance or duration of a trip, the purpose of a trip, the number of distinct riders who use a particular piece of infrastructure, the usage patterns of individuals over time or space, and the infrastructure utilized on a given trip.

In practice, LBS data have large biases resulting from who chooses to use a particular app and to share their location data (e.g., Venter et al (2023)). Moreover, the quantity and quality of LBS data have decreased in the last few years, in part because default privacy settings on smartphones have changed. In particular, Streetlight's estimates are far lower quality since January 2022, when new privacy defaults took effect.

1.2.2 CCS and SCS Data

CCS data can have substantial uncertainties and different CCS technologies are subject to different systematic and random errors. CCS data often have temporal gaps due to equipment failures of various kinds (Glazer et al, 2024). While SCSs are generally assumed to be accurate, the seasonal, daily, and hourly variability of AT traffic are often large, so the naive estimate of AADT from SCS—the average daily traffic during the observation period—is likely to have substantial bias, bias that it might be possible to model and thereby adjust for.

It is expensive to install and maintain CCSs. Moreover, no matter how many CCSs are deployed, there will still be a need to estimate AADT in other places. Thus, we need methods to estimate AADT using CCS data from elsewhere. The accuracy of such estimates depends on the predictability of traffic at a location from CCSs measurements and observable characteristics of that site (*covariates*), such as whether the site is in a central business district, whether it is a dedicated bike lane, or the weather during the short-term count (if a short-term count from an SCS is available). Other covariates that might be useful to improve estimates are mentioned below. Combining SCS data with CCS data raises important *design* questions—e.g., how to install CCSs to maximize the accuracy of AADT estimates at locations where an SCS will be installed or where an AADT estimate will be made with no count whatsoever. Using a given set of CCS and SCS data to estimate AADT at various locations is a matter of *analysis*.

This document is primarily concerned with generating an accurate set of AADT estimates across the infrastructure network of interest. To that end, we first outline how to design a statistically representative network of count stations, and then explore a method for using CCS and SCS data to estimate AADTs where there is no CCS (and possibly no SCS either).

1.3 Previous work on AADT

AADT estimation is well established for motor vehicle estimates, with a history spanning more than 100 years. In the 1920s, government agencies recognized the need for accurate, standardized traffic estimates in response to expanding motor vehicle ownership and infrastructure requirements. The Federal-Aid Highway Act of 1944 tied traffic estimates to federal funding, while the use of SCS counts in conjunction with statistical methods became increasingly common between 1940 and and the mid-1950s (Albright, 1991). The 1966 Drusch report codified the factor group method as a standard AADT estimation technique for motor vehicle traffic (NCHRP, 2024). Factor groups

remain the standard for estimating AADTs from SCS data, as evidenced by the recommendations in the FHWA's 2018 Traffic Data Computation Method Pocket Guide and 2024 *Traffic Monitoring Guide* (TMG). A comprehensive review of AADT estimation via the factor group is available in NCHRP (2024) (pages 8-10), but that report does not dwell on the design. The assumption that CCSs "are equivalent to a simple random sample selection," quoted from the TMG, is not meaningful without both (a) a well defined population and (b) either genuinely random location selection or a reasonable argument that locations can be treated *as if* they are random—that they are not subject to selection bias. In practice, station locations are typically chosen for specific a priori reasons like high traffic volume or accident rates, so selection bias is the rule rather than the exception and CCS data are not representative of a larger population of interest. We include an extended discussion of the design issues in CCS and SCS placement below. In short, to estimate AADT at SCSs it suffices if SCSs are representative of CCSs (via random partioning, for example), but that assumption is not sufficient to estimate AADT at locations with no station whatsoever.

Beyond the factor group method, prediction algorithms—including ordinary least squares, generalized linear models, Kalman filters, decision trees, random forests, boosting, nearest neighbors, and ARIMA—have long been applied to motorized traffic estimation (Boukerche and Wang, 2020). These are surveyed in some detail in NCHRP (2024). More recently, researchers have deployed a host of sophisticated prediction methods often dubbed *deep learning* or *artificial intelligence* and primarily based on variants of neural networks (e.g., Li et al (2018), Yu et al (2018)).

Agencies have become increasingly interested in measuring and estimating AADT for AT (in addition to motor-vehicle traffic) since the early 2000s (Nordback et al, 2016). Since 2013, the FHWA *Traffic Monitoring Guide* has included an entire chapter on AT counting technologies. The AT field has generally borrowed methods from the motor vehicle sector, including the factor group method for AADT estimation. Similarly, researchers have used algorithmic regression methods for estimating AT utilization at various points (Hankey et al, 2012; Chen et al, 2017; Strauss and Miranda-Moreno, 2013). Recent work has attempted to map AADT for AT using LBS data and deep learning (e.g., Gupta et al (2024)). As discussed above, there are systematic biases in LBS data (Venter et al, 2023) and in smartphone data more generally (Crawford, 2013) that make generalizing from LBS data unreliable. Our goal of estimating AADT at point locations using CCS and SCS data is more modest, but under design assumptions the estimates are unbiased and their uncertainty can be reliably quantified.

While translating long-used methods from the motor vehicle literature seems straightforward, there are measurement and data sparsity issues that are unique to AT. In particular, AT sensors can be plagued by various sources of measurement noise (Nordback et al, 2016), and in most places in the U.S., there is far less pedestrian and bicycle traffic than motor vehicle traffic (Gupta et al, 2024). Even a small misclassification rate (a sensor mistaking a car for a pedestrian or bicycle, or vice versa) can cause large errors in SCS and CCS counts (Nordback, 2013; Nordback et al, 2016; Glazer et al 2024).

1.4 Designing a network of counters to make reliable statistical estimates of AT traffic

It is critical to begin at the *design* stage: unless data are collected appropriately, the data cannot be used to estimate traffic reliably anywhere it is not directly measured, i.e., anywhere there is not a CCS. Randomization is the key to sound statistical inference. If CCS sites are selected using a suitable random design, the resulting data can be used to estimate average properties of traffic

for all or part of the road network, for example, for particular types of infrastructure in particular geographic areas.

Generically, we call portions of the transportation network "strata." *Pre-stratification* involves selecting count sites by partitioning the network into non-overlapping groups of locations, then selecting locations separately from each such group. Post-stratification involves combining samples from groups of CCSs that have common features, without ensuring ahead of time that each such group will contain some number of CCSs.

Pre- or post-stratification can improve the estimates for subsets of the transportation network. "Pre-stratification" occurs at the design stage, and entails placing CCSs at random locations within strata, independently across strata. Pre-stratification guarantees that the sample size in a subpopulation of interest (e.g., rural bike paths in District 1) is adequate to get estimates of the required precision for that subpopulation. It has a history of use in active transportation, including to estimate "bicycle miles traveled" in the Twin Cities region (Davis and Wicklatz, 2001) and in Washington state (Nordback and Sellinger, 2014).

"Post-stratification" occurs at the *analysis* stage, and entails estimating averages within strata defined by covariates that were not used to design the sample. It allows inferences to be drawn about a subset of the network that was not specifically targeted when the locations of CCSs were selected, but consequently cannot guarantee that there will be enough CCSs in that subset to estimate properties of traffic with adequate accuracy.

In AT sampling, strata correspond to subsets of the network where AT estimates are desired (by the agency): stratification ensures that there are enough stations in each subset to make reliable estimates of traffic in each of them. In contrast, the typical use of stratified sampling in statistics is to improve population-level estimates by taking advantage of homogeneity within subsets: the strata themselves may not be of particular interest, but are merely a tool to enhance the estimate of a global quantity (e.g., the total yearly traffic across the entire population or network). Strata are not the same as factor groups, even though both involve dividing the population up into subgroups. Factor groups are sets of CCSs that are used to generate expansion factors, which are then combined with an SCS counts to estimate an AADT. Factor groups are generally constructed by post-stratification: finding sets of CCSs with similar covariates and are thus expected to have similar expansion factors.

We will discuss estimating two kinds of quantities: an average within a pre- or post-stratum, and traffic at a particular location within a stratum where no CCS was deployed. If CCS locations are selected at random within pre-strata, the first kind of quantity can be estimated in a way that is statistically unbiased and that has known uncertainty. To estimate the second kind of quantity in a way that is statistically unbiased is in general possible only if the specific location is selected at random from the stratum, *even if a SCS is installed at that location.* Traffic at randomly selected CCSs is representative of traffic at SCS locations if the SCS locations are also selected at random. However, SCS locations are often selected because those locations are interesting, not because they are typical of their stratum or genuinely generated at random. We provide approaches whose statistical performance is guaranteed when SCS locations are "as if" selected at random. Post-stratification may help to meet this assumption, but it is ultimately up to the user to decide whether (as if) random selection is an adequate approximation to reality.

As an example, the agency might pre-stratify on broad geography (Caltrans District), infrastructure (road, trail, path, etc.), and anticipated AT traffic (e.g., fewer than 10 cyclists per day, 10-100, and 100+), selecting CCS locations at random within each stratum. Using data from those CCS

locations, the agency can measure AADT and other summaries of traffic at those CCS locations and can make unbiased estimates of the overall traffic or "average" AADT within each stratum. The agency might also care about divisions smaller than the original strata. To estimate traffic in those areas, the agency might then poststratify on finer geography (City of Oakland, unincorporated Humboldt, etc.) or other aspects (proximity to schools, proximity to roadways, estimated motor vehicle traffic, etc) and use the average within a poststratum to estimate "average" AADT within that post-stratum; under these assumptions, such estimates are also statistically unbiased and their uncertainty is well understood.

Furthermore, an unbiased estimate of average AADT in a pre-stratum or post-stratum is an unbiased estimates of the same parameter for a location *selected at random* from the pre-stratum or post-stratum, and uncertainties can be estimated from the variability of the CCS measurements within the stratum.

However, if the location within the pre- or post-stratum is not selected at random but because it is of special interest, any estimate derived from the CCS data within the stratum can be biased and its uncertainty cannot be quantified without assumptions about how much traffic can vary between CCS locations. In this situation, it is common to augment CCS data in the stratum with SCS data collected at the particular locations of interest. To estimate AADT at purposively chosen sites, we make the convenient but strained assumption that a given site is selected "as if at random" from the pre-stratum or post-stratum. The statistical bias of the resulting estimates and the accuracy of their uncertainties depend on whether that assumption is approximately correct.

As an example, suppose an SCS is deliberately placed at a location near a school. The expansion factors for that SCS could be estimated from (a) all CCSs in the stratum or (b) all CCS in the stratum that are near a school. Choice (b) may lead to an AADT estimate with lower bias and a better sense of associated uncertainty because the SCS in question is better represented by the subset of CCSs near schools than by all CCSs.

We recommend stratified sampling to ensure there is an adequate number of CCSs in the main divisions of the population, e.g., rural and urban areas, bike paths and bike lanes. "Adequate number" depends on the desired precision of the estimates and on the variability of traffic across locations within each stratum (which typically will not be known until stations are deployed, although one might deploy SCSs to get pilot estimates). There is no magic number, but at a *minimum*, we would recommend 30 CCSs per stratum. That might be inadequate for many purposes, for instance, if the variability of AADT within the stratum is large or if there will be post-stratification into finer geography at the analysis stage. The selection of strata should be informed by subject matter experts at the agency to ensure that it is possible to make adequately precise estimates for the places of interest to management. We will assume that post-stratification is a finer division of the sampling strata: no post-stratum has any locations in common with more than one initial stratum. It is possible to construct estimates for post-strata that intersect multiple pre-strata, but we do not address the mechanics of that here.

2 Formalizing the questions statistically

As mentioned above, CCS and SCS data have errors and uncertainties: counters need to be calibrated. The development below assumes that count measurements are perfect, i.e., that CCS and SCS counts are the true numbers of bicycles and/or pedestrians that passed that location while the counters were in place.

Our exposition focuses on a single stratum or post-stratum and assumes that the CCS locations were selected at random uniformly within that stratum. Generally, this involves deploying new counters: previously deployed CCS locations were typically chosen deliberately rather than randomly. The development below explains how one would analyze data from an as-yet-unbuilt network of CCSs designed to support statistical inferences.

Average AADT (A-AADT) means the spatial average of the AADT within a stratum, uniformly weighted across all points in the stratum. A-AADT for a given year is a fixed, unknown parameter. We will assume that CCS locations are selected uniformly at random within a stratum, i.e., that every location in a stratum is equally likely to receive a CCS. We will also assume that the CCS data are complete (there is no missing data, such as temporal gaps in measurement) and perfect (there is no measurement error, e.g. missed counts or added counts).

We consider three problems:

- Estimating A-AADT from CCS data, which can be achieved by taking the usual (unweighted) sample mean within the pre-stratum or post-stratum of interest.
- Estimating AADT at a particular location in a stratum where there is no CCS from CCS data collected from the stratum, which might involve modeling how AADT varies within a stratum.
- Estimating AADT at a particular location in a stratum using CCS data augmented with SCS data, which typically involves adjusting for temporal patterns and taking into account observable features of the target location (*covariates*), for instance using the factor group method.

We introduce formal notation and propose a variety of estimation strategies in what follows.

2.1 Statistical aside: parameters, estimators, estimates

A *parameter* is a property of the population; for instance, the true AADT at a particular location or the true A-AADT across a stratum. Parameters do not depend on the observed data. An *estimator* is a function of the observed data that does not depend on any unknown aspects of the population. For example, multiplying the total measured traffic at a location for a week by 52 is an estimator of the annual traffic at that location–although it might not be a good estimator. When the data come from a random sample, the estimator is random and can be described by a probability distribution (its *sampling distribution*). An *estimate* is the realized value of an estimator. For instance, if the measured traffic in the previous example is 500 bicycles, then the estimate is $500 \times 52 = 26{,}000$ bicycles. The verb "calculate" should be associated with a parameter while "estimate" should be associated with an estimator.

There is some confusion in the literature about whether AADT is a parameter or an estimate; we treat it as a parameter. Because we are assuming that CCS data are perfectly accurate, AADT can be *calculated* wherever there is a CCS, but can only be *estimated* where there is no CCS. Finding a good estimator of AADT from incomplete or noisy data requires appropriate statistical analysis.

2.2 Notation: Population, parameters, estimators, and estimates

In general, we will use bold font Roman letters (e.g., $\mathbf{y}$) to denote vectors. For instance if $\mathbf{y} = [1, 2, 5]$, $\mathbf{y}$ consists of the numbers 1, 2, and 5, in that order. We generally use vectors to represent multiple quantities simultaneously, e.g. a vector may represent the week 1 count for every CCS in the stratum or the list of expansion factors for every time period for a particular CCS. We will

use overbars to denote averages ($\bar{y}$, $\bar{\bar{y}}$), and carets ("hats") to denote estimators and estimates ($\hat{y}$). The symbol r denotes location in space, and t denotes time. We typically use lower-case Roman letters for constants and parameters. The expected value and variance of a random variable Y are denoted $\mathbb{E}[Y]$ and $\mathbb{V}[Y]$, respectively.

2.2.1 Population

There is some network of infrastructure–roads, paths, trails, and so on. Each point in that network can be uniquely identified with a value between 0 and 1. To do so, imagine laying the segments of the road network end-to-end, with the first point on the first segment lying at 0 and the last point of the last segment lying at mile x; dividing every point by x then squashes the network to $[0, 1]$ so that 0 is the start of the network and 1 is the end. For instance, if there are 1,000 miles of infrastructure, milepost 0 of the road whose name comes first in the alphabet could be identified with the point 0; the first mile of that road maps to the interval $[0, 1/1000]$. If that road is m miles long, the road maps to the interval $[0, m/1000]$. The next road alphabetically starts at $m/1000$, and so on. In this way, we can think of any aspect of traffic (for instance, AADT) as a function defined on the interval $[0, 1]$. We stress that the interval $[0, 1]$ is merely one way to *represent* or *embed* a continuous road network. We could just as easily represent the road network by the interval $[0, 500]$. Alternatively, the road network could be represented in two dimensions, e.g., by (x, y) with x the latitude and y the longitude for every point in the network. However, compared to the standard interval $[0, 1]$, these alternative embeddings add unnecessary complexity to conceptualizing the population of interest and to the subsequent design and analysis of count data.

The network is *partitioned* into K strata in such a way that any piece of infrastructure is in exactly one stratum. In the development below, we consider a single stratum. We note that the development extends immediately to a network with multiple strata by implementing the techniques for each stratum in turn. Narrowing focus to a single stratum simplifies our notation.

Time is discrete and indexed by $t \in \{1, \ldots, t_{\max}\}$, where $t_{\max}$ is the total number of counting periods in the year of interest. The counting period might be a 15-minute increment, an hour, a day, or a week, for instance.

We assume we are interested in AADT for a particular historic year. Results from one year are immediately stale unless some aspect of AADT is approximately *stationary* from year to year. For instance, expansion factors for a particular year may continue to be useful in future years despite the fact that traffic varies from year to year, provided annual *patterns* of traffic do not change (that they are stationary from year to year). If patterns also change, all bets are off. For example, compared to motor vehicle traffic, bicycle and pedestrian traffic are particularly sensitive to seasonal, annual, and secular variations in weather. Such variations (e.g., a cooler summer and warmer, dryer winter) could effect traffic patterns, making forecasts less accurate.

Let $y_t(r)$ denote the count of traffic in the tth time interval at location r in the stratum. Let $\mathbf{y}(r) = [y_t(r)]_{t=1}^{t_{\max}}$ be a length-T vector representing all the counts at point r over the course of the year. The counts $\mathbf{y}(r) = [y_t(r)]_{t=1}^{t_{\max}}$ are fully observed at locations r where there are CCSs, partially observed at locations where there are SCSs, and not observed at all for points r with neither a CCS nor an SCS. The population comprises *all* counts $\mathbf{y}(r)$ for $r \in [0, 1]$, even at locations r with no observations. Any parameter of interest (e.g. AADT) at any point r could be calculated if the entire population were known. Finally, every location in the population has a length-p vector of *covariates* $\mathbf{x}(r) := [x_j(r)]_{j=1}^{p}$ recording features of the location.

2.2.2 AADT at a point location

The AADT at location r, is the temporal average of its count vector: the total amount of AT traffic passing location r over the course of the year divided by the number of days in the year. In symbols, we define the AADT at r as

$$\bar{y}(r) := \frac{1}{365} \sum_{t=1}^{t_{\max}} y_t(r)$$

for an ordinary year (not a leap year). We stress that the counts $y_t(r)$ **may be partially or entirely unobserved** at location r: AADT is a parameter, not an estimate based on observed data. The AADT for a given location in a given year is the *actual* total count of traffic passing that location in that year, divided by the number of days in that year. It is not the count observed by an SCS or even observed by a CCS (which may have measurement errors or gaps). Disregarding the potential for measurement error and missing data, AADT can simply be calculated wherever there is a CCS station, since the entire count vector is observed, and thus the total count is known. The collection of AADTs $\{\bar{y}(r)\}_{r\in[0,1]}$ for the whole population represents the set of target parameters that the agency might wish to know. In reality, only the AADTs at locations with CCSs are calculable; AADT at other locations must be estimated.

2.2.3 A-AADT within strata

The *average AADT*, A-AADT, is simply the spatial average of the AADTs over the entire stratum. That is, the A-AADT is the total A-AADT in the stratum divided by the size of the stratum. If location were discrete (i.e., if the stratum were represented by a finite list of locations), the A-AADT would be a simple averageL: the sum of the AADT at every location divided by the number of locations in the list. Since the stratum is a continuous network with uncountably infinite locations, the A-AADT is instead expressed in symbols as an integral over the stratum.

$$\bar{\bar{y}} := \int_0^1 \bar{y}(r) dr.$$

A-AADT (or total traffic) may be an important parameter its own right, e.g., if funding is dispersed to strata proportional to the amount of traffic in each. An A-AADT estimate can also serve as a rudimentary estimate of AADT at a given point in the stratum. This estimation technique is discussed further below.

2.2.4 AASHTO-AADT

Some studies focus on a different annual average, the AASHTO-AADT, rather than AADT as defined above. The AASHTO-AADT takes the average traffic on each day of the week each month (e.g., the average Monday in January), averages the 12 monthly day-of-week averages for each of the seven days with equal weight (e.g., the average of the average Monday in January, the average Monday in February, ..., and the average Monday in December), then averages those seven averages with equal weight. This is a different parameter because the number of Mondays varies from month to month and year to year. If desired, AASHTO-AADT at a point location or stratum average AASHTO-AADT could be the target parameter, in place of AADT and A-AADT. AADT can be calculated from annual total traffic at a site; AASHTO-AADT requires daily totals.

2.3 Data

2.3.1 Random selection

We assume that error-free and complete CCS counts are observed at n_c discrete locations $\{r_j^c\}_{j=1}^{n_c}$ selected uniformly and independently at random from the stratum. (Deviations from this assumption are discussed in the next section.) Every location within the stratum has an equal probability to be sampled for selection; it is possible, but outside the scope of this document, to sample locations with unequal (but still known) probabilities and adjust for the unequal weighting at the analysis stage (e.g., Tille and Wilhelm (2017)).

We may set n_c to be small in strata that are relatively low priority (e.g., because they are suspected to have very low traffic), and high in strata that are higher priority. Strata with relatively high variability in AADT should also receive more CCSs (especially when that variability is not well proxied by covariates) when it is important to estimate A-AADT more precisely in that stratum. For instance, consider a "less variable stratum," wherein AADT is distributed uniformly between 400 and 800 (A-AADT = 600), and a "more variable stratum," wherein AADT is distributed uniformly between 100 and 1100 (A-AADT = 600). When estimating A-AADT using CCS data, the less variable stratum would need fewer CCSs than the more variable stratum would in order to achieve a given level of precision.

Deploying CCSs at random locations makes it possible to quantify the uncertainty in estimating A-AADT (conversely, without random sampling the uncertainty cannot be quantified reliably). This model may not be realistic for CCSs that have already been deployed. It is our understanding that existing CCS locations in California were not selected at random, but were usually selected because the sites were in some sense special (for instance, because there were collisions, because new infrastructure was contemplated or had been installed, or because the agency wanted to measure motor vehicle traffic and decided to measure AT traffic at the same time). It is wishful thinking to believe that A-AADT can be estimated well from CCS data unless the CCS locations are representative. (And it is wishful thinking to believe that AADT can be estimated reliably from SCS data at some location using CCS data from other locations unless the SCS site is suitably similar to the CCS sites.)

While the locations of already-deployed CCSs were not selected at random (the *design* assumption is false for extant counters), Caltrans can deploy stations at random within strata (code is provided below) to create a statistical network of counters to make it possible to estimate A-AADT reliably, with known uncertainties.

Finally, we assume that SCSs are deployed at n_s locations $\{r_j^s\}_{j=1}^{n_s}$ in the stratum. Unless these n_s locations are also selected at random, they will be of little help in estimating A-AADT for the stratum. Unless there is a stable relationship between AADT and covariates of sites, having an SCS at location r is not necessarily helpful in estimating AADT at the point r.

For each CCS location r_i^c, $i = 1, \ldots, n_c$, the entire count vector $\mathbf{y}(r_i)$ is observed.

For each SCS location r_i^s, $i = 1, \ldots, n_s$, we only observe counts at times $\mathbf{t}_i \subset \{1, \ldots, T\}$. We will generally assume that observations at each SCS are contiguous. (E.g., we do not sample January 1st and January 15th without the intervening days. If it were practical to observe non-contiguous counts, e.g., a day in January, a day in April, a day in July, and a day in October, higher accuracy might be possible from the same number of counts, but the logistics and costs are likely prohibitive.)

From counts at a location r observed at random times, one can make an unbiased estimate of

AADT at that location. If the random times are selected independently, the uncertainty of that estimate can be characterized. In what follows, we will not necessarily assume that the times at which SCS measurements are made at a particular location are selected at random.

The count vectors for all $n_c + n_s$ sampled stations can be stacked into a matrix $\mathbf{Y}$, where row j is $\mathbf{y}(r_j)$ and CCS counts are in the first n_c rows. The observed data matrix $\mathbf{Y}^*$ is a censored version of the matrix $\mathbf{Y}$, where rows $j \in \{n_c + 1, \ldots, n_c + n_s\}$ are missing every element except in columns $\mathbf{t}_j$. The covariate vectors $\mathbf{x}_j$ can be stacked in the same order into a matrix $\mathbf{X}$, and the observed data can be written: $(\mathbf{Y}^*, \mathbf{X})$.

2.3.2 Data without random selection

What if station locations are *not* chosen by a random selection mechanism like the one above? For instance, a site might be selected because it is known to have high traffic or to be particularly dangerous. In that case, the random sampling mechanism described above may not be a good model for the real selection mechanism generating the data, and there is likely to be substantial bias from the selection.

For example, if CCSs are installed where yearly counts are suspected to be high, simple extrapolation to other locations will have an upward bias: estimates of AADT and A-AADT will tend to be too big. In some circumstances, that bias can be modeled well enough to adjust for it using covariates. In particular, post-stratifying on variables that influence selection can reduce selection bias.

However, it is generally not possible to estimate selection bias from the data alone. Contextual reasoning and experience are required to determine if the data are sufficiently accurate to inform policy decisions. In what follows, we propose using cross-validation to measure the quality of predictions of long-term counts, but **cross-validation only provides accurate quantification of prediction risk under the random sampling assumptions**. Researchers and policy advisors need to judge whether their data is sufficiently representative.

Valid conclusions may still be possible in the presence of unknown selection bias, though the scope of those conclusions may be limited. In particular, suppose both CCSs and SCSs are a weighted (i.e., biased) sample from a stratum and the weights are unknown, but identical for both groups of stations: both samples are subject to the same selection bias. For example, this occurs if SCSs and CCSs are both placed preferentially along high traffic paths, and the (hypothetical) probabilities of placement at a particular location can be assumed equal for a given SCS and CCS. Then the choice of whether a given station is a CCS or SCS is essentially arbitrary, the CCSs are representative of the SCSs, and the A-AADT for the n_s SCSs (but not the whole stratum!) can be estimated from observed data on SCSs and CCSs. Combined, the SCSs and CCSs are not necessarily representative of the stratum, and it is not possible to provide statistical guarantees about the accuracy of stratum-level A-AADT or about AADT at locations without stations.

Even these assumptions may be unrealistic considering the historic placement of count stations. Different counting technologies are often used for CCSs (infrared or inductive loops) compared to SCSs (video counts), and the technologies tend to function differently in different settings: loops work well on trails, while video counts are better for roads with bike lanes. Thus, in practice, SCSs may be preferentially placed on bike lanes while CCSs are placed on trails, so that CCS counts are not properly representative of SCS counts.

3 Estimation

A-AADT across a stratum and AADT at a point without a CCS can be estimated using data from a well-designed network of count stations. We will focus on estimation techniques that use *post-stratification*, which includes the factor group method for expanding SCS counts.

An estimate of A-AADT is denoted $\hat{\bar{\bar{y}}}$, while an estimate of the AADT for a location r_i is denoted $\hat{\bar{y}}(r_i)$. If n_0 estimates are made, the vector of estimates is $\hat{\bar{\mathbf{y}}} := [\hat{\bar{y}}(r_j)]_{j=1}^{n_0}$ (bold font) while the vector of true, unknown AADTs is $\bar{\mathbf{y}} := [\bar{y}(r_j)]_{j=1}^{n_0}$ (also bold font). Roughly speaking, an estimator of A-AADT is "good" if the predictions $\hat{\bar{\mathbf{y}}}$ are close to the true AADTs $\bar{\mathbf{y}}$ in a sense that reflects the priorities of the agency. Because there are countless estimators one could use, we suggest selecting an estimator by first choosing an error measure aligned with agency goals, then choosing an estimator that approximately minimizes that error. Additional qualitative properties such as interpretability, covariate sparsity, or privacy may also weigh into the choice of estimator.

In this section, we discuss a few possibilities for defining the distance between estimates and parameters, propose a general method to measure that distance under random sampling assumptions using only the observed data, and suggest some estimators of AADT and A-AADT.

3.1 Figures of merit (loss and risk)

3.1.1 Loss functions

We denote a generic *loss function* by ℓ. We will use the same notation for the loss function for AADT at a point r_i and for A-AADT. For instance, $\ell(\bar{y}(r_i), \hat{\bar{y}}(r_i))$ is the loss in estimating AADT at location r_i, i.e., the "cost" incurred by claiming that the truth is $\hat{\bar{y}}_i$ when it is really $\bar{y}_i$; $\ell(\bar{\bar{y}}, \hat{\bar{\bar{y}}})$ is the loss in estimating A-AADT, i.e., the "cost" incurred by claiming that the truth is $\hat{\bar{\bar{y}}}$ when it is really $\bar{\bar{y}}$.

The choice of loss function encodes priorities of the agency. In particular, consider the following two loss functions.

- **Squared-error loss** penalizes estimates based on their squared distance from the true AADTs. It takes the difference between the true AADT and the estimated AADT and squares it (so that the loss is always positive). In symbols:

$$\ell_s(\bar{y}(r_i), \hat{\bar{y}}(r_i)) := (\bar{y}(r_i) - \hat{\bar{y}}(r_i))^2.$$

  For example, if the true AADT is 30 and the estimated AADT is 33, the squared-error loss is $(30-33)^2 = (-3)^2 = 9$. Larger errors are penalized more (an error of 2ϵ incurs 4 times the loss as error ϵ), and there is no adjustment for the size of stations: an off-by-5 error in an AADT estimate receives the same penalty whether the true AADT for station i is 10 bikes or 1,000 bikes.

- **Proportional loss** normalizes by the true counts, so the error is in terms of the proportion rather than volume. It takes the absolute value (so that the loss is always positive) of the difference between the true AADT and estimated AADT and divides by the true AADT (so that the difference is expressed in proportion to the truth). In symbols:

$$\ell_p(\bar{y}(r_i), \hat{\bar{y}}(r_i)) := \frac{|\bar{y}(r_i) - \hat{\bar{y}}(r_i)|}{\bar{y}(r_i)}.$$

Multiplying the proportinal loss by 100 gives the loss on a percent scale. For example, if the true AADT is 30 and the estimated AADT is 33, the proportional loss is $|30-33|/30 = 0.10$, or, multiplying by 100, it is 10%. Proportional loss is less analytically tractable than squared-error loss, but prediction methods can be evaluated numerically for their proportional loss.

These loss functions correspond to different priorities. The choice between them is substantive, not statistical. For squared-error loss, the overall numerical error in counts is important, so predictions will be particularly attuned to high-traffic stations in urban areas. For proportional loss, percent error in counts is important, so predictions will be more uniformly accurate across stations with varying traffic volume. Thus, proportional loss leads to predictive equity across stations rather than individual riders. Furthermore, proportional loss is sensitive to errors on the order of magnitude (e.g., overestimating the AADT by a factor of 10), not on an absolute scale (e.g., overestimating by 10). Agencies may prefer estimates to be "in the ballpark" on most stations rather than absolutely accurate for large stations.

Both proportional and squared-error loss are symmetric: they penalize undershooting and overshooting by exactly the same amount. In some situations, it might be better to use an asymmetric loss. For instance, when the AADT estimates are inputs to models prioritizing safety upgrades, overestimating the AADT may cost some additional money in upgrades, while underestimating the AADT might contribute to additional accidents and even fatalities: traffic is higher in reality than the agency believes.

NCHRP (2024) proposes a few performance metrics, including homogeneity of the factors, proportional loss ("absolute percent error"), the algebraic (signed) difference $\hat{\bar{y}}(r_i) - \bar{y}(r_i)$, and the coefficient of variation $100 \times (2 \times |\hat{\bar{y}}(r_i) - \bar{y}(r_i)|)/\sqrt{2}(\hat{\bar{y}}(r_i) + \bar{y}(r_i))$. Homogeneity of the factors is related to squared-error loss under random sampling assumptions, but is not itself a loss function: all else equal, if the expansion factors for a given factor group are more homogenous, the estimate of A-AADT within that factor group is more precise and the squared-error loss is lower. The signed difference is not an appropriate loss function in our view because it can be negative, and thus will not aggregate into a sensible risk measure—overshooting an estimate for one sample can be offset by undershooting for another sample so that the expected loss is 0 even though no single estimate is particularly close to the true AADT. The coefficient of variation is valid but not particularly easy to interpret; it is equal to $100 \times \sqrt{2} \approx 10.4$ whenever $\hat{\bar{y}}(r_i) = 0$, 0 when $\hat{\bar{y}}(r_i) = \bar{y}(r_i)$, and converges back to 140 as $\hat{\bar{y}}(r_i)$ gets large.

3.1.2 Risk

Because estimates are random under the random sampling assumptions above, the loss is a random variable. To compare estimators, we evaluate their *risk*: their expected loss under the random sampling design. That is, for estimating AADT at location r_i the risk is $R(\bar{y}(r_i), \hat{\bar{y}}(r_i)) := \mathbb{E}[\ell(\bar{y}(r_i), \hat{\bar{y}}(r_i)]$ and for estimating A-AADT the risk is $R(\bar{\bar{y}}, \hat{\bar{\bar{y}}}) := \mathbb{E}[\ell(\bar{\bar{y}}, \hat{\bar{\bar{y}}})]$, where both expectations are with respect to the (deliberate) randomness in selecting where to install CCSs. We might also evaluate the risk in estimating AADT at n_0 locations $\{r_i\}_{i=1}^{n_0}$ as a weighted average of the risks for the individual locations:

$$R(\bar{\mathbf{y}}, \hat{\bar{\mathbf{y}}}) := \sum_{i=1}^{n_0} q_i R(\bar{y}(r_i), \hat{\bar{y}}(r_i)).$$

The weights $q_i := q(r_i)$ must be nonnegative and sum to 1. They encode the priority assigned to location r_i: if location r_i is higher priority, $q_i > 1/n_0$, and $q_i = 1$ implies that only the prediction accuracy at r_i matters to the agency; vice versa, weights $q_i < 1/n_0$ correspond to lower priority

locations and $q_i = 0$ implies location r_i does not matter at all. If all n_0 estimation locations are prioritizied equally, the weights are $q_i = 1/n_0$ and the aggregate risk is the standard average of the risks at individual locations.

3.1.3 Estimating Risk Using Cross-validation

In practice, the risk is unknown because the true parameters (AADT or A-AADT) are unknown. The risk must itself be estimated by a quantity $\hat{R}$, which is computed using only observed CCS data. This estimate is reasonable under the random sampling assumptions described above, because then the CCSs are representative of the entire stratum. (If stations are not sited randomly, estimates of risk based on CCS data may have large bias.) Furthermore, when estimating the risk of an AADT estimate at the point r_i, we are tacitly assuming that the location r_i was itself randomly selected by the mechanism that was used to select the CCS locations. This assumption is *never* true in practice because locations for estimation will be chosen deliberately by the agency. If this assumption is very far from true, the estimates are useless. There is no way to make this appraisal entirely algorithmic. It requires specific knowledge about the network: is the location r_i where an estimate is desired exceptional or similar to the CCSs in its stratum, especially with regard to characteristics that are not represented by its covariates? On the other hand, the risk of A-AADT estimation can be quantified with no additional assumptions.

In either case, we will generally choose an estimator that minimizes the estimated risk $\hat{R}$, but other considerations like transparency and interpretability may weigh in as well. For instance, when choosing between a black-box machine learning algorithm and a simple estimator based on taking averages over strata, we may favor the simple estimator even if it has a higher value of $\hat{R}$.

Now, the cross-validation risk estimate $\hat{R}$ is an unbiased empirical estimate of R under the random sampling assumptions presented above. Cross-validation splits the CCS data at random into K groups, "trains" the estimator on $(K-1)$ of the groups, and evaluates the performance of that estimator on the remaining group. The process is repeated K times, with each group held out in turn, and the performance averaged to form the estimate $\hat{R}$. Example code is provided below.

Cross validation is not the only sensible choice to obtain $\hat{R}$, for instance, one could instead estimate R using the bootstrap or other resampling methods. See, e.g., James et al. (2023).

3.2 Estimating A-AADT

Estimating the A-AADT $\bar{\bar{y}}$ across a stratum is the simplest of the three estimation tasks we discuss, and can be done with the smallest uncertainty. Indeed, the usual sample mean of the AADTs for the observed CCSs (dividing the sum of CCS AADTs by the number of CCSs)

$$\frac{1}{n_c}\sum_{i=1}^{n_c}\bar{y}(r_i^c)$$

is an unbiased estimate of the true A-AADT, and has a standard error equal to the heterogeneity (standard deviation) σ of AADT across the stratum divided by the sample size $\sqrt{n_c}$. Its risk under any loss function of interest can be estimated using cross-validation or other resampling methods.

CCS data could also be augmented with data from randomly located SCSs to improve A-AADT estimates. AADT is first estimated at SCS locations using a method like factor group estimation, described below. The AADT estimates at SCS locations and the AADT calculations at CCS locations are then averaged across the stratum. However, in the (hypothetical) case that CCSs are

randomly located while SCSs are purposively located, factoring SCSs into the estimates could make them worse than the sample mean of AADT across CCSs. Even if SCSs were randomly located, the usual standard error estimate (ignoring the variability in AADT estimates at the SCS sites) would underestimate the true uncertainty. The correct formula includes the uncertainty in estimating AADT at each SCS

$$\mathrm{SE}[\hat{\bar{\bar{y}}}] := \sqrt{\frac{1}{(n_c+n_s)^2}\left[n_c\sigma + \sum_{i=1}^{n_s}\mathbb{V}[\hat{\bar{y}}(r_i^s)]\right]},$$

where the variance $\mathbb{V}[\hat{\bar{y}}(r_i^s)] \geq \sigma$ in each SCS AADT estimate is a function of the random sampling (hence it is lower bounded by the heterogeneity σ) as well as specifics of the station's characteristics and the estimation procedure used. If the AADT estimate is perfect and incurs no additional error, then $\mathbb{V}[\hat{\bar{y}}(r_i^s)] = \sigma$ and $\mathrm{SE}[\hat{\bar{\bar{y}}}] = \frac{\sigma}{\sqrt{n_c+n_s}}$, which is a lower bound on the standard error of the sample mean estimate of A-AADT when combining CCSs and SCSs.

When covariates are available, they might be used to refine estimates of A-AADT. In essence, these methods work by estimating AADT at every point in the stratum before averaging those estimates to estimate A-AADT. Post-stratification may be used in this way: averages are computed within post-strata and combined by a weighted average across post-strata (the weights are proportional to the relative size of each post-stratum) to estimate A-AADT. This "adjusts" for small, random deviations in the representativeness of CCSs. A complete treatment of such strategies is outside the scope of this document (but see, e.g., Sarndal et al (1992)).

3.3 Estimating AADT at an SCS station using expansion factors

Agencies have typically estimated motor vehicle and AT AADT at SCSs using *expansion factors*. Expansion factor methods multiply an SCS count by a constant estimated from CCS data (the expansion factor) in order to estimate AADT at that SCS location. The expansion factor for a particular SCS location for a particular SCS measurement period (e.g., the 21st week of the year) is derived by averaging the corresponding ratio for some subset of the CCSs. The subsets are called *factor groups*. A factor group could represent all the CCSs in the stratum or all the CCSs within a smaller poststratum defined by additional covariates.

Formally, a factor group for the SCS i located at r_i^s is any subset of the CCS locations $\mathcal{F}_i \subset \{r_i^c\}_{i=1}^{n_c}$. Factor groups could be based on covariates, counts, priorities, hunches, or other discretionary considerations of the agency. The factor group average AADT is the sum of the CCS AADTs within the factor group divided by the number of CCSs in the factor group. Letting $|\mathcal{F}_i|$ denote the number of CCSs in $\mathcal{F}_i$, this is denoted in symbols by

$$\bar{y}_{\mathcal{F}_i} := \frac{1}{|\mathcal{F}_i|}\sum_{r_j^c \in \mathcal{F}_i}\bar{y}(r_j^c),$$

and is a known and calculable quantity because the AADTs at all the CCSs are known.

From here on, we assume that the sampling period provides a single observed count (e.g., total traffic in a week) so that the observation period is a scalar ($\mathbf{t}_i = t_i$) and the observed count can be written $[y_t(r_i^s)]_{t\in\mathbf{t}_i} = y_{t_i}(r_i^s)$. This assumpion accomodates daily or hourly counts that are summarized as weekly counts; finer (e.g. hourly) counting periods may still be used to compute covariates, e.g., the ratio of weekend to weekday traffic, which may be useful in defining factor groups.

For each location r, there are T expansion factors $\{w_t(r)\}_{t=1}^T$ that would scale the observed count for each period to the true AADT at that location. In other words, an expansion factor at location r for period t is the AADT at r divided by the count in period t. In symbols,

$$w_t(r) := \frac{\bar{y}(r)}{y_t(r)}.$$

For instance, for weekly counts, there are 52 expansion factors for each location, one for each week of the year. If the AADT at r is 100 and its count for week 1 is 5, the corresponding expansion factor is $w_1(r) = 100/5 = 20$, indicating that the AADT is 20 times higher than the week 1 traffic (this seems reasonable since bicycle traffic may be much lower in mid-winter compared to typical traffic for the year).

We emphasize that, in our framing, expansion factors are fixed, but typically unknown. In fact, expansion factors are only known when there is a CCS at $r = r_i^c$. In that case, the expansion factors at r_i^c can be calculated exactly (barring measurement error or missing data) because all the counts are known. Expansion factors (fixed parameters) may be conflated with estimates of expansion factors (uncertain functions of observed data). As above, we attempt to distinguish parameters from estimates of those parameters as much as possible.

At an SCS location r_i^s, the true expansion factor for the observed period $w_{t_i}(r_i^s)$ is unknown (because the AADT is unknown). Instead, we use an uncertain but known *estimated expansion factor* $\widehat{w}_{t_i}(r_i^s)$ to estimate the AADT r_i. In particular, the AADT estimate at r_i^s is estimated as the product of the observed count at r_i^s in period t_i and the estimated expansion factor for r_i^s. In symbols,

$$\hat{\bar{y}}(r_i^s) := \widehat{w}_{t_i}(r_i^s) \times y_{t_i}(r_i^s).$$

We next discuss how to obtain the estimated expansion factor $\widehat{w}_{t_i}(r_i^s)$.

3.3.1 Estimating expansion factors

Expansion factors for an SCS in a particular factor group are estimated by the average of the known CCS expansion factors across the factor group, i.e., by the sum of CCS expansion factors across the factor group divided by the number of CCSs in the factor group. In symbols, the estimated expansion factor is written

$$\widehat{w}_t(r_i^s) := \frac{1}{|\mathcal{F}_i|} \sum_{r_j^c \in \mathcal{F}_i} w_t(r_j^c),$$

where $w_t(r_j^c) = y_t(r_j^c)/\bar{y}(r_j^c)$ is the known expansion factor for the CCS station at r_j^c. Each CCS expansion factor is self-normalized by the volume of the CCS station, so the CCS stations in factor group $\mathcal{F}_i$ are given equal weight in determining the overall expansion factor estimate for SCS i. We call this the "averaging" estimator for expansion factors.

It isn't clear whether giving equal weight to all CCSs in $\mathcal{F}_i$, preferentially weighting by volume, or any other form of weighting (e.g., giving higher weight to CCSs in $\mathcal{F}_i$ that are closer in covariate space to SCS i) minimizes the error in AADT estimates and hence the risk of the estimator. Under the assumptions made above, cross-validation provides a principled way to choose an estimated expansion factor as well as a factor grouping method by providing an unbiased estimate of the risk under any proposed strategy.

A factor group $\mathcal{F}_i$ comprising all CCSs in the stratum The choice $\mathcal{F}_i := \{r_j^c\}_{j=1}^{n_c}$ uses all the CCSs in the stratum to compute expansion factors. For example, suppose we observe 100 bikes at SCS i in week 1 (January 1-7), so that $y_1(r_i^s) = 100$. Also suppose there are 3 CCSs in the stratum (this is very few) with week 1 expansion factors $w_1(r_1^c) = 5$, $w_1(r_2^c) = 5$, and $w_1(r_3^c) = 20$. That is, for CCSs 1 and 2, the week 1 count is 5 times less than the AADT at those stations, while for station 3 the week 1 count is 20 times less than the AADT at that station. Since the factor group for SCS i consists of all 3 CCSs in the stratum, we estimate the expansion factor for SCS i as $\widehat{w}_t(r_i^s) = (5 + 5 + 20)/3 = 10$. We then take the estimate of AADT at SCS i to be its estimated expansion factor for week 1 multiplied by its observed count for week 1:

$$\hat{\bar{y}}(r_i^s) = 10 \times 100 = 1000.$$

Refinements of $\mathcal{F}_i$ Factor groups may be defined using covariates available at SCSs, or by any other means at the discretion of the agency. Compared to using all CCSs, refinements can improve estimates of $\bar{y}(r_i^s)$ by giving greater weight to CCSs that are more similar to SCS i. However, smaller factor groups do not always mean better AADT estimates: as a rule of thumb, good factor groups are both relatively homogenous in terms of expansion factors (the CCSs are similar to each other) and large (there are many CCSs).

For example, suppose the pre-strata were defined as Caltrans Districts and SCS r_i^s is located in a small rural town in District 4 (the greater Bay Area). Rather than all CCSs in District 4, we may want to set $\mathcal{F}_i$ to include only CCSs in small rural towns in District 4. This is an example of post-stratification, where the post-strata could be derived from categorizing and combining continuous covariates like "population" or "distance from a major metropolitan center."

There are two main reasons post-stratification can improve estimates of $\bar{y}(r_i^s)$:

1. The SCS location may not have been chosen at random. For example, it may have been deliberately placed in an area with a high population or where traffic fatalities are suspected to be high. In that case, the expansion factors for the whole set of (randomly located) CCSs would not be representative of the unobserved expansion factors at the SCS. The unknown expansion factors might be better estimated by refining to CCSs with a high population, high traffic fatalities, or similarities along other features that entered into the SCS location selection process. In short, incorporating covariate information to refine the factor group could help *decrease bias*.

2. The counts can be well-approximated by a simple function of the covariates. For example, suppose station i is placed in a rural location. A working *model* for $\bar{y}(r_i^s)$ might posit that the AADT at r_i^s can be decomposed into the A-AADT, plus a deviation term for rural locations, plus an idiosyncratic (residual) deviation of that particular station from the average. In symbols, we would write

$$\bar{y}(r_i^s) = \bar{\bar{y}} + \delta_{\text{rur}} \cdot 1\{r_i^s \text{ is rural}\} + \varepsilon(r_i^s),$$

   where $\bar{\bar{y}}$ is the A-AADT, δ_{rur} is the average deviation from the A-AADT for rural stations, and $\varepsilon(r_i^s)$ is any residual deviation of the AADT at r_i^s from the average AADT of rural stations in the stratum. Comparing station i to other rural locations can yield a sharper estimate of the $\bar{y}_i$ when δ_{r} is relatively large, i.e., when urban/rural status is a strong predictor of average counts. (Correlation is enough: it is not necessary that a given covariate actively determine or influence the counts for this benefit to apply.) Thus, covariates can *increase precision*.

Both decreasing bias and increasing precision will tend to decrease R, the risk of the estimator, improving estimates in expectation. As a rule of thumb, good factor groupings are those that partition the CCSs so that heterogeneity is low within a factor group and high between factor groups: ideally, CCSs in the same factor group should have similar expansion factors to each other and different expansion factors from CCSs in other factor groups (on average). That is, the within and between variance of the CCS expansion factors provides an easy criterion for choosing factor groups. This heuristic can be taken too far, however, since the ratio of between over within group variance can always be maximized by putting every CCS into its own factor group. There is generally a happy medium between factor groups that are large and heterogeneous (tending to produce estimates with low variance but high bias) and factor groups that are small and homogenous (tending to produce estimates with low bias but high variance). This *bias-variance* tradeoff can be navigated using cross-validation to provide unbiased estimates of the risk.

Now various choices of factor group $\mathcal{F}_i$ could be generated using *a priori* knowledge, reason, and experience about what drives expansion factors or AADT. For example, the covariates could be partitioned according to:

- Whether the location is in a central business district
- The degree of urbanization
- The nature of the infrastructure (road, bike lane, dedicated bike path, etc)
- Whether the location is within some distance to a school

If SCS i possesses any of these qualities, $\mathcal{F}_i$ consists of CCSs with the same qualities. This is the usual approach to estimating AADT and other parameters of SCSs based on pre-specified factor groups.

However, each $\mathcal{F}_i$ does not need to be specified using discretion alone: it could be chosen algorithmically. For example, clustering methods can be used to produce factor groups, although the groupings must still be judged for their homogeneity on key covariates by researcher scrutiny (see NCHRP (2024) Chapter 3) and/or cross-validation. Alternatively, supervised learning methods like K-nearest neighbors may be used. In that case, for the SCS at r_i^s, the Euclidean distance between $\mathbf{x}(r_i^s)$ and each of the CCS covariates $\mathbf{x}(r_j^c)$ for $j \in \{1, \dots, n_c\}$. Then $\mathcal{F}_i$ could be set to represent the K "closest" CCSs to the SCS at r_i^s (in terms of the distance between covariates, *not* geographic distance–although geography could be a covariate). This approach is called K-nearest neighbors. The number $K \in \{1, \dots, N\}$ is a tuning parameter that indexes a set of n_c possible methods that could be used to estimate $\bar{y}(r_i^s)$, from the single most similar CCS to all CCSs in the stratum. To choose the appropriate number of neighbors, a risk estimate $\hat{R}$ could be made for each of the n_c possibilities, and we could select K^* with the smallest $\hat{R}$.

3.4 Estimating AADT at the point r when there is no station at r

Caltrans may wish to estimate an AADT $\bar{y}(r)$ when r has neither a CCS nor an SCS (we still assume that r is in a *stratum* where there are CCS counts).
To put it bluntly, **there is no way to do this without making additional assumptions that may or may not be reasonable**. Those assumptions come in two flavors: (1) the location of r is essentially random, or (2) at the point r, the AADT $\bar{y}(r)$ is essentially randomly distributed in a manner similar to the distribution of the data at the CCSs.

Assumption (1) is questionable because the location r typically will not be random: the agency does not choose where to make AADT estimates as if by throwing darts at a map. As with AADT

estimation at an SCS, the agency must consider whether it is reasonable to treat the location *as if* it were random. If features of the location that led Caltrans to want to estimate AADT there might be associated with AADT, that assumption is shaky. Examples of such features include unusually high population density, a history of traffic fatalities, proximity to a public transit hub, etc. Assumption (2) is typically even more strained. Especially for AT, AADT can vary across space in a highly idiosyncratic and discontinuous fashion that is sensitive to local particulars and not well modeled as a random draw from common probability distributions (Gaussian, Poisson, zero-inflated binomial, etc). For this reason, we suggest the agency record features that influence selection, adjust for the influence of those features where possible to reduce bias, and be aware of the propensity for AADT estimates made in this way to have large biases.

Features that influence both selection and AADT are *confounders*, and will bias estimates at purposively chosen locations. Even very simple estimates are subject to this bias. For instance, although the mean of AADTs at randomly-located CCSs is guaranteed to be an unbiased estimate of the A-AADT and in fact is an unbiased estimate of the AADT at any *random* location in the stratum, it is **not** an unbiased estimate of the AADT at a purposively chosen location.

If relevant features are recorded as covariates, bias can be reduced by post-stratification and other statistical techniques usually called *regression* or *supervised learning*. In particular, we could posit a model for AADT that claims that the AADT at location r is some unknown function of the known covariates, plus an idiosyncratic residual or error term that does not depend on the covariates. In symbols we can write the model as

$$\bar{y}(r) = f(\mathbf{x}(r)) + \epsilon(r),$$

where f is an unknown (potentially non-linear or discontinuous) function of the observed covariate vector $\mathbf{x}(r)$ and $\epsilon(r)$ is the idiosyncratic deviation of site r from that function. An estimate $\hat{f}$ of the unknown function f can be used to estimate the AADT $\bar{y}(r)$ at r by plugging the covariates $\mathbf{x}(r)$ into $\hat{f}$. In symbols, the estimate is written

$$\hat{\bar{y}}(r) := \hat{f}(\mathbf{x}(r)).$$

There are no statistical guarantees about the predictions unless we make assumptions about $\epsilon(r)$, e.g., the unrealistic assumption that $\epsilon(r)$ is drawn from a normal distribution with mean 0 and standard deviation σ_ϵ for all r.

A very simple set of assumptions holds that (a) f is entirely constant over the stratum and (b) the variation $\epsilon(r)$ is randomly drawn from a distribution with mean zero. Taken together, these assumptions do not imply that AADT is exactly constant (which is obviously untrue), but they do imply that AADT can be treated as essentially random with a mean that does not vary systemtically over geography or other features of the locations. Because AADT obviously depends on many features of the network (e.g. the type of infrastructure, motor vehicle traffic, population density, etc), the assumption is unrealistic. Nevertheless, if these assumptions hold, then it follows that $f = \bar{\bar{y}}$ (the true function f is just equal to the true A-AADT), and furthermore the A-AADT estimate $\hat{\bar{\bar{y}}}$ is a good predictor of $\bar{y}(r)$ at any location r. Let $\mathcal{C} := \{r_i^c\}_{i=1}^{n_c}$ be the set of all CCS locations, and recall that $\bar{y}_\mathcal{F}$ is the mean AADT of the locations in $\mathcal{F}$. If the CCSs were randomly selected, the mean of *all* the CCS AADTs $\bar{y}_\mathcal{C}$ is known and, if f is indeed constant, is a good estimate of $\hat{f}$. Under that assumption, the location and any covariate information is irrelevant: we estimate the AADT for any r to be $\bar{y}_\mathcal{C}$.

We might instead assume that f is a piecewise constant function of the covariates; for instance, that f is the same for all bikepaths in the same city but different between any two cities (locations outside any city might all be lumped together and assumed equal). Then post-stratification might be used to estimate f. In particular, if the stratum is partioned into post-strata, and the set $\mathcal{C}_s$ denotes CCS locations in post-stratum s, then the post-stratification estimate for any location in post-stratum s would be the average of CCS counts across post-stratum s:

$$\hat{\bar{y}}(r) := \bar{y}_{\mathcal{C}_s}.$$

This estimate might improve on the A-AADT estimate in terms of reducing the risk at r. It might be particularly accurate if location r can be treated as-if it were a uniform random draw from the post-stratum, i.e. if it is indistinguishable from other locations in the post-stratum from the point of view of the agency. This would be the case if, for instance, the agency actually used the covariates (and only the covariates!) to choose the location r. In reality, it seems likely that r will be chosen for reasons that are not entirely represented in the covariates (e.g., because a manager with some local knowledge not encoded in the database asked for the AADT at location r, or because new infrastructure is being planned at location r).

If we do not wish to assume that f is piecewise constant, other supervised learning techniques should be used to estimate f, including linear regression (ordinary least squares), nearest neighbors, random forests, boosting, neural networks, and so on. We will not elaborate on these techniques here. The first order concern in AADT estimate at points is selection bias, which requires careful scrutiny and transparent estimation techniques to address. There is a premium on simplicity. We leave it to future researchers to investigate the value of more sophisticated estimation methods.

4 References

1. Albright, David. 1991. "History of Estimating and Evaluating Annual Traffic Volume Statistics." Transportation Research Record, no. 1305. https://trid.trb.org/View/365623.
2. Bhowmick, Debjit, Meead Saberi, Mark Stevenson, Jason Thompson, Meghan Winters, Trisalyn Nelson, Simone Zarpelon Leao, Sachith Seneviratne, Christopher Pettit, Hai L. Vu, Kerry Nice, Ben Beck. 2023. "A Systematic Scoping Review of Methods for Estimating Link-Level Bicycling Volumes." Transport Reviews, July. https://www.tandfonline.com/doi/full/10.1080/01441647.2022.2147240.
3. Boukerche, Azzedine, and Jiahao Wang. 2020. "Machine Learning-Based Traffic Prediction Models for Intelligent Transportation Systems." Computer Networks 181. https://doi.org/10.1016/j.comnet.2020.107530.
4. Chen, Peng, Jiangping Zhou, and Feiyang Sun. 2017. "Built Environment Determinants of Bicycle Volume: A Longitudinal Analysis." Journal of Transport and Land Use 10 (1). https://doi.org/10.5198/jtlu.2017.892.
5. Crawford, Kate. 2013. "The Hidden Biases in Big Data." Harvard Business Review, April 1, 2013. https://hbr.org/2013/04/the-hidden-biases-in-big-data.
6. Davis, Gary A, and Trina Wicklatz. 2001. "Sample-Based Estimation of Bicycle Miles Traveled (BMT)." Report for the Minnesota Department of Transportation. MN/RC – 2001-23. https://cts-d8resmod-prd.oit.umn.edu/pdf/mndot-2001-23.pdf
7. Drusch, Robert L. 1966. "Estimating Annual Average Daily Traffic from Short-Term Traffic Counts." Highway Research Record. https://trid.trb.org/View/120596.
8. Ferster, Colin, Trisalyn Nelson, Karen Laberee, and Meghan Winters. 2021. "Mapping Bicycling Exposure and Safety Risk Using Strava Metro." Applied Geography 127.

https://doi.org/10.1016/j.apgeog.2021.102388.
9. Glazer, Amanda, Philip B. Stark, Md. Mintu Miah, Krita Nordback, Julia B. Griswold, and Alexander Skabardonis "Data checks for bicycle and pedestrian counts". Poster presented at Transportation Research Board 103rd Annual Meeting, Washington, DC, January 9, 2024. https://www.researchgate.net/publication/377362665_DATA_CHECKS_FOR_BICYCLE_AND_PEDES
10. Gupta, Mohit, Debjit Bhowmick, Meead Saberi, Shirui Pan, and Ben Beck. 2024. "Evaluating the Effects of Data Sparsity on the Link-Level Bicycling Volume Estimation: A Graph Convolutional Neural Network Approach." ArXiv preprint. https://doi.org/10.48550/arXiv.2410.08522.
11. Hankey, Steve, Greg Lindsey, Xize Wang, Jason Borah, Kristopher Hoff, Brad Utecht, and Zhiyi Xu. 2012. "Estimating Use of Non-Motorized Infrastructure: Models of Bicycle and Pedestrian Traffic in Minneapolis, MN." Landscape and Urban Planning 107. https://doi.org/10.1016/j.landurbplan.2012.06.005.
12. James, Gareth, Daniela Witten, Trevor Hastie, Robert Tibshirani, and Jonathan Taylor. 2023. An Introduction to Statistical Learning: With Applications in Python. Springer. https://link.springer.com/book/10.1007/978-3-031-38747-0.
13. Li, Yaguang, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. "Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting." International Conference on Learning Representations (ICLR), 2018. https://openreview.net/forum?id=SJiHXGWAZ.
14. McNally, Michael G., James E. Marca, Craig R. Rindt, and Angela M. Koos. 2003. "TRACER: In-Vehicle, GPS-Based Wireless Technology for Traffic Surveillance and Management." California PATH Research Report. UCB-ITS-PRR-2003-23. https://escholarship.org/uc/item/19f152mt.
15. Nordback, Krista. 2014. "Guide to Bicycle & Pedestrian Count Programs." Transportation Research and Education Center. https://trec.pdx.edu/sites/default/files/smol-IBPI%20Guide%20to%20Bicycle%20%26%20Pedestrian%20Count%20Programs_final.pdf.
16. Nordback, Krista, Sirisha Kothuri, Miguel A. Figliozzi, Taylor Phillips, Carson Gorecki, Andrew Schrope, and Portland State University. 2016. "Investigation of Bicycle and Pedestrian Continuous and Short Duration Count Technologies in Oregon : Final Report : SPR 772." FHWA-OR-RD-16-15. https://rosap.ntl.bts.gov/view/dot/30956.
17. Nordback, Krista, and Michael Sellinger. 2014. "Methods for Estimating Bicycling and Walking in Washington State." Report for The State of Washington Department of Transportation. WA-RD 828.1. https://www.wsdot.wa.gov/research/reports/fullreports/828.1.pdf.
18. Selala, M. K., and W. Musakwa. 2016. "The Potential of Strava Data to Contribute in Non-Motorised Transport (NMT) Planning in Johannesburg." The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. https://doi.org/10.5194/isprs-archives-XLI-B2-587-2016.
19. Strauss, Jillian, and Luis F. Miranda-Moreno. 2013. "Spatial Modeling of Bicycle Activity at Signalized Intersections." Journal of Transport and Land Use 6. https://doi.org/10.5198/jtlu.v6i2.296.
20. Tillé, Yves, and Matthieu Wilhelm. 2017. "Probability Sampling Designs: Principles for Choice of Design and Balancing." Statistical Science 32. https://doi.org/10.1214/16-STS606.
21. Tsapakis, Ioannis, Paul Anderson, Zihang Wei, Shawn Turner, Anik Das, Mark Hallenbeck, Ben Chen, and Elizabeth Stolz. 2024. Guide on Methods for Assigning Counts to Adjustment Factor Groups. National Academies Press. https://doi.org/10.17226/27925.
22. United States Department of Transportation Federal Highway Administration. 2024. "Traffic Monitoring Guide." FHWA-PL-022-026. https://rosap.ntl.bts.gov/view/dot/74643.
23. United States Federal Highway Administration Office of Highway Policy Informa-

tion. 2018. "Traffic Data Computation Method: Pocket Guide." FHWA-PL-18-027. https://rosap.ntl.bts.gov/view/dot/57335.
24. Venter, Zander S., Vegard Gundersen, Samantha L. Scott, and David N. Barton. 2023. "Bias and Precision of Crowdsourced Recreational Activity Data from Strava." Landscape and Urban Planning 232. https://doi.org/10.1016/j.landurbplan.2023.104686.
25. Yu, Bing, Haoteng Yin, and Zhanxing Zhu. 2018. "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting." In Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18). https://dl.acm.org/doi/10.5555/3304222.3304273.

5 Software

The following section provides software to accomplish a variety of tasks:

1. Sample a set of CCSs and SCSs from a population partioned into strata.
2. Calculate AADT at a point with a CCS station
3. Estimate A-AADT
4. Estimate AADT at a point with an SCS location using expansion factors
5. Estimate the risk of AADT estimation at a randomly located SCS using cross-validation
6. Estimate AADT at a point where there is no station by using the A-AADT estimate or poststratification

We first load in necessary Python packages and define necessary functions. We then demonstrate each task in order. Except for the first, the tasks are demonstrated using data from bicycle counters in California in 2019. The data was processed and prepared by Amanda Glazer (Glazer et al, 2024).

5.1 Setup

5.1.1 Load external packages

[1]:
```
# import necessary packages

import numpy as np
import pandas as pd
import scipy as sp
import warnings

#suppress some warnings

warnings.filterwarnings("ignore", message="numpy.dtype size changed")
warnings.filterwarnings("ignore", message="numpy.ufunc size changed")
warnings.filterwarnings('ignore', "Intel MKL WARNING")
```

[2]:
```
# package versions

print(sp.__version__)
print(np.__version__)
print(pd.__version__)
```

1.14.1
2.2.0
2.2.3

5.1.2 Load functions

Risk and loss

[3]:
```
# tools to estimate risk empirically

def sample_loss(true: np.array=None, predicted: np.array=None, loss_func:␣
↪callable=None, weights: np.array = None) -> float:
    '''
    compute the loss function for estimated AADT(s) when the truth is known;␣
↪aggregate by averaging if multiple AADTs are passed
    this is the loss for a single sample. The true "risk" is defined as an␣
↪expected value involving the design
    this function can be used to estimate the true risk by simulation,␣
↪resampling, cross-validation, etc

    parameters
    ----------
        true: np.array of n_s floats, where n_s is the number of short-term␣
↪stations
            the true values
        predicted: length-n_s np.array of floats
            the predicted values
        loss_func: callable
            the loss function
        weights: length-n_s np.array
            optional vector of weights that sum to 1, defaults to 1/n_s

    returns
    -------
        a float representing the risk
    '''
    n_s = len(predicted)
    weights = ( weights if weights is not None
                else (1/n_s)*np.ones_like(predicted)
              )
    assert len(weights) == n_s, "Length of weights does not equal length of␣
↪predictions"
    return np.sum(weights * loss_func(true, predicted))

# some componentwise loss functions
def squared_error_loss(true, predicted):
    return (true - predicted)**2
```

```
def proportional_loss(true, predicted):
    return np.abs(true - predicted) / true


# unit tests
# sample loss
true = np.array([1])
predicted = np.array([1])
assert sample_loss(true, predicted, squared_error_loss) == 0
assert sample_loss(true, predicted, proportional_loss) == 0
true = np.array([1,2])
predicted = np.array([1,2])
assert sample_loss(true, predicted, squared_error_loss) == 0
assert sample_loss(true, predicted, proportional_loss) == 0
```

Factor groups

[4]:
```
def define_factor_groups_allgroups(n_s: int=None, n_c: int=None) -> list:
    '''
    returns a factor group F_i for each of the n_s SCSs; each factor group␣
 ↪contains every CCS, i.e. |F_i| = n_c

    parameters
    ----------
        n_s: int
            the number of SCSs
        n_c: int
            the number of CCSs
    returns
    -------
        a length-n_s list of factor groups F_i, each one of which is [0,...
 ↪,(n_c - 1)], i.e. an index of all CCSs
    '''
    F = [np.arange(n_c)] * n_s
    return F

def define_factor_groups_poststratification(SCS_strata: np.array=None,␣
 ↪CCS_strata: np.array=None) -> list:
    '''
    Construct a factor group for each SCS. The factor group for an SCS is the␣
 ↪list of indices of CCSs in its post-stratum

    parameters
    ----------
        SCS_strata: length-n_s np.array of ints or strs
            identifiers of the post-strata to which the SCSs belong
        CCS_strata: length-n_c np.array of ints or strs
```

```
            identifiers of the post-strata to which CCSs belong

    returns
    -------
        list of np.arrays: a length-n_s list of factor groups F_i, each of␣
 ↪which is an np.array containing the
        indices of the CCSs in the same poststratum as the SCS
    '''
    F = []
    n_c = len(CCS_strata)
    for SCS_stratum in SCS_strata:
        F_i = np.where(SCS_stratum == CCS_strata)[0]
        if F_i.size == 0:
            F_i = np.arange(n_c) # if the SCS stratum is not in the CCSs, its␣
 ↪factor group is all CCSs
            warnings.warn(f'SCS stratum {SCS_stratum} contains no CCSs: using␣
 ↪default factor group containing all CCSs.')
        F.append(F_i)
    return F


def define_factor_groups_nearestneighbor(
        SCS_X: np.array=None, CCS_X: np.array=None, K: int=None, distance:␣
 ↪str='mahalanobis') -> list:
    '''
    Construct n_s factor groups, one for each SCS, comprising the K CCSs␣
 ↪closest in covariate space in the pseudometric `distance`

    Covariates are assumed to be float-valued, so categorical covariates should␣
 ↪be encoded using dummy variables.

    parameters
    ----------
        SCS_X: size-(n_s, p) np.array (matrix) of floats
            the covariate matrix for the SCSs
        CCS_X: size-(n_c, p) np.array of floats
            the corresponding covariate matrix for the CCSs (columns must align␣
 ↪with columns of SCS_X)
        K: int
            the number of nearest neighbors to match on, i.e. the desired size␣
 ↪of each factor group
        distance: str
            the name of the distance function in covariate space for finding␣
 ↪the nearest neighbors.
            defaults to Mahalanobis distance
```

```
            see the documentation for sp.spatial.distance.cdist for more␣
 ↪information on distances that can be used
            https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.
 ↪distance.cdist.html

    returns
    -------
        a length-n_s list of factor groups F_i, one for each SCS, each of size K
    '''
    dists = sp.spatial.distance.cdist(SCS_X, CCS_X, metric = distance)
    # Get the indices of the K nearest neighbors for each row in SCS_X
    neighbors_indices = np.argsort(dists, axis=1)[:, :K]
    F = [neighbors_indices[i,:] for i in range(neighbors_indices.shape[0])]
    return F


# unit tests
# test allgroups
F = define_factor_groups_allgroups(n_s = 1, n_c = 3)
assert np.all(F[0] == np.array([0,1,2]))
F = define_factor_groups_allgroups(n_s = 3, n_c = 3)
assert all([np.all(F[i] == np.array([0,1,2])) for i in range(3)])

# test poststratification
scs_strata = np.array(["pear", "apple"])
ccs_strata = np.array(["pear", "pear", "apple"])
F = define_factor_groups_poststratification(SCS_strata = scs_strata, CCS_strata␣
 ↪= ccs_strata)
assert np.all(F[0] == np.array([0,1]))
assert np.all(F[1] == np.array([2]))

# test nearest neighbors
X_scs = np.array([[0.,0], [1,1], [1,0]])
X_ccs = np.array([[0.,0], [1,1]])
F = define_factor_groups_nearestneighbor(SCS_X = X_scs, CCS_X = X_ccs, K = 1)
assert np.all(F[0] == np.array([0]))
assert np.all(F[1] == np.array([1]))
F = define_factor_groups_nearestneighbor(SCS_X = X_scs, CCS_X = X_ccs, K = 2)
assert np.all(F[0] == np.array([0, 1]))
assert np.all(F[1] == np.array([1, 0]))
```

Form CCS expansion factors

[5]:
```
def estimate_expansion_factors(F_i: np.array=None, y_c: np.array=None,␣
 ↪factor_method: str="averaging") -> np.array:
    '''
```

```
    estimate expansion factors for factor group F_i from matrix of CCS counts␣
 ↪y_c

    parameters
    ----------
        F_i: np.array of ints
            the indices of the factor group for SCS i in the set of CCSs
            e.g., 1,...,n_c corresponds to all CCSs
        y_c: size-(n_c, t_max) np.array (i.e., a matrix)
            the complete counts for all CCSs
        factor_method: str in ["averaging", "ratio"]
            the method used to compute the expansion factor
            "averaging" sets w_i for week t to the average of the n_c expansion␣
 ↪factors for week t across all CCSs in F_i
             "ratio" sets w_i for week t to the total annual count across all␣
 ↪CCSs in F_i divided by the total count
                 across all CCSs in F_i in week t; it is not recommended at␣
 ↪this time
    returns:
    --------
        length-t_max np.array, the expansion factors for factor group F_i
    '''
    days_per_year = 365
    y_F_i = y_c[F_i,:] # all CCS counts in factor group F_i
    if factor_method == "ratio":
        AADT_bar = np.mean((1/days_per_year) * np.sum(y_F_i, axis = 1)) #␣
 ↪average AADT across all CCSs in factor group F_i
        week_avg = np.mean(y_F_i, axis = 0) # average weekly counts across CCSs␣
 ↪in factor group F_i
        w_it = AADT_bar / week_avg # ratio expansion factors
    elif factor_method == "averaging":
        AADT_F_i = (1/days_per_year) * y_F_i.sum(axis = 1, keepdims = True) #␣
 ↪AADTs for CCSs in group F_i
        w_F_i = AADT_F_i / y_F_i # expansion factors for each week and each CCS␣
 ↪in group F_i
        w_it = np.mean(w_F_i, axis = 0) # averaging expansion factors
    else:
        raise ValueError("Invalid value for factor_method argument; must be␣
 ↪'ratio' or 'averaging'")
    return w_it

def estimate_all_expansion_factors(F: list=None, y_c: np.array=None,␣
 ↪factor_method: str='averaging') -> np.array:
    '''
    wrapper for estimate_expansion_factors: estimate all expansion factors for␣
 ↪factor groups in F from matrix of CCS counts y_c
```

```
    parameters
    ----------
        F_i: list of np.arrays of ints
            the indices in the set of CCSs for the factor group for each SCS i␣
 ↪in 1...n_s
        y_c: size-(n_c, t_max) np.array (i.e., a matrix)
            the complete counts for all CCSs
            the method used to compute the expansion factor
            "averaging" sets w_i for week t to the average of the n_c expansion␣
 ↪factors for week t across all CCSs in F_i
            "ratio" sets w_i for week t to the total annual count across all␣
 ↪CCSs in F_i divided by the total count
                 across all CCSs in F_i in week t; it is not recommended at␣
 ↪this time
    returns
    -------
        length-t_max np.array, the expansion factors for factor group F_i
    '''
    t_max = y_c.shape[1]
    n_s = len(F)
    w = np.zeros((n_s, t_max))
    for i in range(n_s):
        w[i,:] = estimate_expansion_factors(F[i], y_c, factor_method)
    return w

# unit tests

# test estimate_expansion_factors
F_i = np.array([0, 1, 2]) # factor group is all CCSs
y_c = np.array([[100] * 52, [200] * 52, [300] * 52]) # constant weekly counts␣
 ↪for each CCS
assert np.all(estimate_expansion_factors(F_i, y_c) == (52/365) * np.ones(52))

# test estimate_all_expansion_factors
F = [np.array([0, 1, 2]), np.array([0])]
y_c = np.array([[100] * 52, [200] * 52, [300] * 52]) # constant weekly counts␣
 ↪for each CCS
w = estimate_all_expansion_factors(F, y_c)
assert np.all(w[0] == (52/365) * np.ones(52))
assert np.all(w[1] == (52/365) * np.ones(52))
```

Expansion factor estimate of AADT

[6]:
```
def estimate_aadt_factor(Y_i: float=None, T_i: float=None, w_it: float=None) ->␣
 ↪float:
    '''
```

```
    construct an estimate of the AADT for SCS i using the factor group method

    parameters
    ----------
        Y_i: positive int or float
            the observed count for SCS i
        T_i: int or float in 1,...t_max
            the time at which station i was observed
        w_it: length-52 np.array of floats
            the expansion factors for SCS i

    returns
    -------
        estimate of the AADT for station i
    '''
    y_hat = w_it[T_i] * Y_i
    return y_hat


def estimate_all_aadts_factor(Y: np.array=None, T: np.array=None, w: np.
 ↪array=None) -> list:
    '''
    wrapper for estimate_aadt_factor: factor group estimates for all SCSs from␣
 ↪observed counts, weeks, and expansion factors

    parameters
    ------------
        Y: length-n_s np.array
            the observed counts for each SCS
        T: length-n_s np.array of ints in 1,...t_max
            the times at which each station was observed
        w: (n_s, 52) np.array
            vector (in columns) of expansion factors for each SCS (in rows)
    returns
    --------
        list of floats: estimates of the AADT for each SCS
    '''
    n_s = len(Y)
    y_hats = []
    for i in range(n_s):
        y_hat_i = w[i, T[i]] * Y[i]
        y_hats.append(y_hat_i)
    return y_hats

# unit tests

# test estimate_aadt_factor
```

```
w = (52 / 365) * np.ones(52)
Y = 100
T = 1
assert estimate_aadt_factor(Y, T, w) == 100 * (52 / 365 )

# test estimate_all_aadts_factor
w = (52 / 365) * np.ones((3, 52))
Y = 100 * np.ones(3)
T = np.ones(3).astype(int)
estimates = estimate_all_aadts_factor(Y, T, w)
assert all([estimates[i] == (100 * (52 / 365)) for i in range(3)])
```

Cross-validation estimate of risk for expansion factor estimation

[7]:
```
def K_fold_CV_factorgroups(
        group_method: str=None,
        K: int=None,
        y: np.array=None,
        T: np.array=None,
        X: np.array=None,
        factor_method: str='averaging',
        k_nn: int=None,
        loss_func: callable=squared_error_loss
    ) -> tuple:
    '''
    estimate the risk of a factor group estimate of AADT from SCS counts using␣
 ↪K-fold cross-validation
    the input data is from CCS stations only
    simulates SCS data by randomly splitting the CCS data

    parameters
    ----------
        group_method: string in ['all', 'poststratification',␣
 ↪'nearest_neighbors']
            the factor group method to be used
        K: int >= 2
            the number of folds (splits of the data) for cross-validation
        y: shape-(n,52) np.array of floats
            weekly counts for all stations
        T: length-n np.array of ints in 0,...,51
            the week in which each station was "observed"
            could be a fixed week, or a random one (e.g. for certain kinds of␣
 ↪averaging across weeks)
        X: shape-(n,p) np.array of floats
            the covariates for all stations (auxiliary features)
        factor_method: str in ['averaging', 'ratio']
            the factor method to be used
```

```
        k_nn: int
            the number of nearest-neighbors (if the nearest-neighbor method is␣
↪used).
        loss_func: callable in [squared_error_loss, proportional_loss]
            the loss function to be used in evaluating the risk
    returns
    -------
        pair of floats: (estimate of the risk, standard error of that estimate)
    '''
    days_per_year = 365
    n = y.shape[0]
    AADT = (1/days_per_year) * np.sum(y, axis = 1)
    Y = y[np.arange(n), T]
    folds = np.random.choice(np.repeat(np.arange(K), np.ceil(n/K))[0:n], size =␣
↪n, replace = False)
    risk_hats = np.zeros(K)
    for k in range(K):
        # split the data
        ix_c = np.where(folds != k)[0]
        ix_s = np.where(folds == k)[0]
        y_c = y[ix_c,:] # pseudo-CCS counts
        y_s = y[ix_s,:]  # pseudo-SCS counts
        # AADTs
        AADT_s = AADT[ix_s]
        AADT_c = AADT[ix_c]
        # covariates
        if X is not None:
            X_s = X[ix_s,:]
            X_c = X[ix_c,:]
        # observed week
        Y_s = Y[ix_s]
        T_s = T[ix_s]

        if group_method == "all":
            F = define_factor_groups_allgroups(len(ix_s), len(ix_c))
        elif group_method == "poststratification":
            assert X.shape[1] == 1, "X can only have 1 categorical column (the␣
↪strata)"
            F = define_factor_groups_poststratification(X_s, X_c)
        elif group_method == "nearest_neighbors":
            assert k_nn is not None, "Input an integer number of␣
↪nearest-neighbors to use as k_nn"
            F = define_factor_groups_nearestneighbor(X_s, X_c, K = k_nn)
        else:
            raise ValueError(f"group_method {group_method} not supported")
        w_hat = estimate_all_expansion_factors(F, y_c, factor_method =␣
↪factor_method)
```

```
        y_hats = estimate_all_aadts_factor(Y = Y_s, T = T_s, w = w_hat)
        risk_hats[k] = sample_loss(predicted = np.array(y_hats), true = AADT_s,␣
 ↪loss_func = loss_func)
    risk_hat = np.mean(risk_hats)
    se_risk_hat = np.std(risk_hats)
    return risk_hat, se_risk_hat


# unit tests

# factor groups as all CCSs
y = np.ones((10, 52)) # all the counts are the same so there should be no error
T = np.ones(10).astype(int)
assert K_fold_CV_factorgroups(group_method = "all", K = 2, y = y, T = T) == (0,␣
 ↪0)
assert K_fold_CV_factorgroups(group_method = "all", K = 5, y = y, T = T) == (0,␣
 ↪0)
assert K_fold_CV_factorgroups(group_method = "all", K = 10, y = y, T = T,␣
 ↪factor_method = 'ratio') == (0, 0)

# factor groups by post-stratification
strata = np.transpose(np.array([["pear", "apple"]*5]))
assert K_fold_CV_factorgroups(group_method = "poststratification", K = 2, y =␣
 ↪y, X = strata, T = T) == (0, 0)
assert K_fold_CV_factorgroups(group_method = "poststratification", K = 5, y =␣
 ↪y, X = strata, T = T) == (0, 0)
assert K_fold_CV_factorgroups(group_method = "poststratification", K = 10, y =␣
 ↪y, X = strata, T = T) == (0, 0)

# factor groups by nearest neighbors
strata = np.random.normal(size = (10, 5))
assert K_fold_CV_factorgroups(group_method = "nearest_neighbors", K = 2, y = y,␣
 ↪X = strata, T = T, k_nn = 2) == (0, 0)
assert K_fold_CV_factorgroups(group_method = "nearest_neighbors", K = 5, y = y,␣
 ↪X = strata, T = T, k_nn = 2) == (0, 0)
assert K_fold_CV_factorgroups(group_method = "nearest_neighbors", K = 10, y =␣
 ↪y, X = strata, T = T, k_nn = 2) == (0, 0)
assert K_fold_CV_factorgroups(group_method = "nearest_neighbors", K = 2, y = y,␣
 ↪X = strata, T = T, k_nn = 5) == (0, 0)
assert K_fold_CV_factorgroups(group_method = "nearest_neighbors", K = 5, y = y,␣
 ↪X = strata, T = T, k_nn = 5) == (0, 0)
assert K_fold_CV_factorgroups(group_method = "nearest_neighbors", K = 10, y =␣
 ↪y, X = strata, T = T, k_nn = 5) == (0, 0)
```

6 Example workflow

The following code shows how to use python and the functions defined above to

1. Sample a set of CCSs and SCSs from a population partioned into strata.
2. Calculate AADT at a point with a CCS station
3. Estimate A-AADT in a stratum
4. Estimate AADT at a point with an SCS location
5. Estimate the risk of AADT estimation at a randomly located SCS using cross-validation
6. Estimate AADT at a point where there is no station

6.1 Generate random locations for CCSs

6.1.1 Functions for random sampling

[8]:
```
def map_point_along_segments(point: float=None, segment_lengths: np.array=None)␣
↪-> tuple:
    '''
    given a point on [0,1], and an np.array of segment lengths,
    map the point to a segment, returning the index of the segment and the
    distance along that segment at which the point falls

    parameters
    ----------
        point: float in [0,1]
        segment_lengths: np.array
            an np.array of segment lengths
    returns
    -------
        the index and the distance (in the original units) along the segment on␣
↪which the point falls
    '''
    # line the segments up and index them by their start and end points␣
↪canonically mapped to the interval [0,1]
    segment_ends = np.cumsum(segment_lengths) / np.sum(segment_lengths) # index␣
↪segments end-to-end and then squish to [0,1]
    segment_starts =  np.insert(np.array(segment_ends)[:-1], 0, 0) # segment␣
↪starts are just shifted ends, with 0 appended as the start of the first␣
↪segment
    ix = np.where((segment_starts <= point) & (point <= segment_ends))[0][0] #␣
↪index of the segment where the point falls; note that if there is a tie the␣
↪end of the earlier segment is taken (this happens with probability zero when␣
↪points are randomly sampled)
    distance_on_segment = segment_lengths[ix] * (point - segment_starts[ix]) /␣
↪(segment_ends[ix] - segment_starts[ix]) # compute the distance along chosen␣
↪segment at which the point falls, distance is in original units (not [0,1]␣
↪scale)
    return ix, distance_on_segment
```

```
def map_points(points: np.array=None, segment_lengths: np.array=None) -> np.
 ↪array:
    '''
    wrapper for map_point_to_mile, loops over a set of points and maps all of␣
 ↪them

    parameters
    ----------
        points: np.array of floats in [0,1]
        segment_lengths: np.array
            an np.array of segment lengths
    returns
    -------
        np.array of segment indices and the distances along each segment (in␣
 ↪the original units) on which the points fall
    '''
    out = []
    # loop over points
    for i in range(points.shape[0]):
        out.append(np.array(map_point_along_segments(points[i],␣
 ↪segment_lengths)))
    return np.array(out)

# unit tests for functions defined above

# point should fall halfway through (that is, 1 unit into) the second segment
point = 3/4
segment_lengths = np.array([2, 2])
assert map_point_along_segments(point, segment_lengths) == (1, 1)

point = 0.5
segment_lengths = np.array([1, 1])
assert map_point_along_segments(point, segment_lengths) == (0, 1)

points = np.array([0, 0.25, 0.3125, 0.625, 0.6251, 1])
segment_lengths = np.array([1, 1, 1, 2, 3])
mapped_points = map_points(points, segment_lengths)
assert np.all(mapped_points[0,:] == (0, 0))
assert np.all(mapped_points[1,:] == (1, 1))
assert np.all(mapped_points[2,:] == (2, 0.5))
assert np.all(mapped_points[3,:] == (3, 2))
assert np.all(mapped_points[4,:] >= (4, 0))
assert np.all(mapped_points[5,:] == (4, 3))
```

[9]:
```
# load highway segment data

road_data = pd.read_excel("../Data/CA Highways 10.02.2023.xlsx") # data from␣
 ↪Caltrans
road_data = road_data[["THY_ID", "THY_DISTRICT_CODE", "THY_COUNTY_CODE",␣
 ↪"THY_ROUTE_NAME", "THY_BEGIN_PM_AMT", "THY_END_PM_AMT",␣
 ↪"THY_LENGTH_MILES_AMT"]] # only select columns with id, district, or␣
 ↪county-route-postmile information
road_data
```

[9]:

	THY_ID	THY_DISTRICT_CODE	THY_COUNTY_CODE	THY_ROUTE_NAME	\
0	72098940	12	ORA	1	
1	72098942	12	ORA	1	
2	72096786	12	ORA	1	
3	72096787	12	ORA	1	
4	72098949	12	ORA	1	
…	…	…	…	…	
59805	71429782	4	ALA	980	
59806	71429783	4	ALA	980	
59807	71429784	4	ALA	980	
59808	71429785	4	ALA	980	
59809	71429775	4	ALA	980	

	THY_BEGIN_PM_AMT	THY_END_PM_AMT	THY_LENGTH_MILES_AMT
0	0.12900	0.17000	0.04100
1	0.17000	0.20400	0.03400
2	0.20400	0.23100	0.02700
3	0.23100	0.25300	0.02200
4	0.25300	0.28300	0.03000
…	…	…	…
59805	2.01000	2.01200	0.00200
59806	2.01200	2.02300	0.01100
59807	2.02300	2.02500	0.00200
59808	2.02500	2.03598	0.01098
59809	2.03598	2.03600	0.00002

[59810 rows x 7 columns]

[10]:
```
# example of stratified sampling from Caltrans highway dataframe

# initialize the pseudo-random number generator
seed = 314152653 # use an actual random seed, e.g., from die rolls
np.random.seed(seed)

# define design parameters (stratum and sample size)
stratum = 12 # define the stratum from which to draw (here, District 12 is the␣
 ↪stratum)
```

```
n_c = 10 # number of random CCSs locations to generate

# draw random samples from interval [0,1]
random_samples = np.random.uniform(size = n_c)

# filter to stratum and subset columns
stratum_data = road_data[road_data['THY_DISTRICT_CODE'] == stratum] # filter␣
 ↪data to the stratum of interest
stratum_data = stratum_data[['THY_ID', 'THY_LENGTH_MILES_AMT']] # subset to␣
 ↪columns necessary to draw sample

# map randomly sampled points to segments, and append distance along segment
mapped_points = map_points(points = random_samples, segment_lengths = np.
 ↪array(stratum_data['THY_LENGTH_MILES_AMT'])) # locate the randomly sampled␣
 ↪points in the stratum

# locate segments in dataframe
sampled_segment_indices = mapped_points[:,0] # take the indices
sampled_segments = stratum_data.iloc[sampled_segment_indices,:].copy() # data␣
 ↪sliced using the indices

# append distances for each segment
sampled_segments['POINT_ON_SEGMENT'] = mapped_points[:,1] # distance along each␣
 ↪segment of each sampled point
sampled_segments
```

[10]:

	THY_ID	THY_LENGTH_MILES_AMT	POINT_ON_SEGMENT
14544	72140188	0.057	0.034964
43932	72100178	1.836	0.347604
4935	72129302	0.029	0.006462
25745	72142025	0.204	0.135583
31356	72102615	0.926	0.672080
22075	72143288	4.163	3.660293
4820	72128433	0.042	0.029605
52025	72105720	0.066	0.050678
56440	72135575	0.352	0.066656
22128	72144300	0.084	0.015324

[11]:
```
# import weekly bicycle counts

bike_covariates = pd.read_csv("../Data/cluster_data/bike_data.csv") # covariates
bike_weekly_counts = pd.read_csv("../Data/cluster_data/weekly_bike_volume.csv")␣
 ↪# weekly counts

# filter to data from 2019

bike_weekly_counts_2019 = bike_weekly_counts[bike_weekly_counts['year'] == 2019]
```

```
bike_covariates_2019 = bike_covariates[bike_covariates['year'] == 2019]


# This code calibrates expansion factors using only stations with complete data;
 ↪ one could also use a method to impute some
# missing data, but that will introduce additional uncertainties and biases.
# Drop CCSs with missing data

number_of_weeks = bike_weekly_counts_2019.groupby(['id']).size() # count the␣
 ↪number of weeks at each station
bike_weekly_counts_2019 = bike_weekly_counts_2019.groupby(['id']).filter(lambda␣
 ↪x: len(x) == 52) # filter to CCSs with all 52 weeks observed
## next line generates a warning
bike_covariates_2019 = bike_covariates_2019[bike_covariates_2019['id'].isin(np.
 ↪array(bike_weekly_counts_2019['id']))] # filter covariates
bike_covariates_2019 = pd.get_dummies(bike_covariates_2019,␣
 ↪columns=['Bicycle_Fa']) # make dummy variables from categorical covariates

dummy_columns = [col for col in bike_covariates.columns if col.
 ↪startswith('Bicycle_Fa_')] # note dummy columns
bike_weekly_counts_2019_wide = bike_weekly_counts_2019.pivot(
    index = 'id',
    columns = 'weekno',
    values = 'weekly_total_Volume') # pivot weekly data to wide format
bike_weekly_counts_2019_wide = bike_weekly_counts_2019_wide.
 ↪loc[bike_weekly_counts_2019_wide.sum(axis=1) != 0] # drop station with 0␣
 ↪counts to avoid divide-by-zerio problems in example (NB: don't do this in␣
 ↪practice without further consideration of why counts are zero)

# set index columns to align covariates with counts

#bike_weekly_counts_2019_wide.set_index('id', inplace = True) # set index␣
 ↪column for counts
bike_covariates_2019.set_index('id', inplace = True) # set index column for␣
 ↪covariates
bike_covariates_2019 = bike_covariates_2019.
 ↪reindex(bike_weekly_counts_2019_wide.index) # reorder so covariates and␣
 ↪counts are in the same order

# simpler names for the rest of the notebook

covariates = bike_covariates_2019 # rename
weekly_counts = bike_weekly_counts_2019_wide # rename
```

[12]:
```
covariates
```

[12]:

id	Unnamed: 0	site_name	\
100000671	0	BART AT SPIRE	
100003560	24	Inland Rail Trail (San Marcos)	
100003561	60	Oceanside SLR River Trail EB & WB	
100003565	84	San Diego: Sorrento Valley Rd NB & SB	
100003568	96	San Diego: River Bike Path EB & WB	
…	…	…	
100041348	2700	PedBikeCorpYard	
100043437	2724	Old Town Trail	
100043453	2784	La Mesa: University Ave WB - Bike	
100044068	2820	Bob Jones Trail	
100044161	2844	(18) Fell St. Between Scott & Divisadero	

id	Distance_to_water_area	distanc_sch_coll_uni	location	year	month	\
100000671	0.293825	0.283585	urban	2019	1	
100003560	0.328181	0.338044	urban	2019	1	
100003561	0.352247	0.845815	urban	2019	1	
100003565	1.090513	0.588773	urban	2019	1	
100003568	0.793997	0.856428	urban	2019	1	
…	…	…	…	…	…	
100041348	0.080085	1.105047	urban	2019	1	
100043437	0.150630	0.052433	urban	2019	1	
100043453	1.475801	0.768590	urban	2019	1	
100044068	0.643191	0.815764	urban	2019	1	
100044161	0.084511	0.354334	urban	2019	1	

id	MTT	days_in_month	MADBT	…	AADBT	lat	\
100000671	1233.0	31	39.774194	…	48.221918	37.791160	
100003560	3350.0	31	108.064516	…	125.630137	33.146637	
100003561	14932.0	31	481.677419	…	556.131507	33.203396	
100003565	4895.0	31	157.903226	…	176.276557	32.917549	
100003568	8212.0	31	264.903226	…	306.587171	32.760265	
…	…	…	…	…	…	…	
100041348	2189.0	31	70.612903	…	78.419178	40.855014	
100043437	1828.0	31	58.967742	…	90.939726	36.829650	
100043453	240.0	31	7.741935	…	9.531507	32.767470	
100044068	7190.0	31	231.935484	…	292.383562	35.185000	
100044161	37460.0	31	1208.387097	…	1480.380822	37.774200	

id	long	MOYF	Strata	Bicycle_Fa_Bike Lane	\
100000671	-122.459440	0.824816	University	False	
100003560	-117.183560	0.860180	University	False	
100003561	-117.387390	0.866121	Waterbody	False	

```
100003565 -117.238245  0.895770        Path                 False
100003568 -117.202200  0.864039        Path                 False
…               …        …          …                    …
100041348 -124.088709  0.900455   Waterbody                False
100043437 -119.700158  0.648427  University                False
100043453 -117.020800  0.812247   Bike lane                 True
100044068 -120.704000  0.793258        Path                False
100044161 -122.436295  0.816268  University                 True

          Bicycle_Fa_Bike Route  Bicycle_Fa_Cycle Track  Bicycle_Fa_Path  \
id
100000671                 False                   False             True
100003560                 False                   False             True
100003561                 False                   False             True
100003565                 False                   False             True
100003568                 False                   False             True
…                         …                       …                 …
100041348                  True                   False            False
100043437                 False                   False            False
100043453                 False                   False            False
100044068                 False                   False             True
100044161                 False                   False            False

           Bicycle_Fa_Trail
id
100000671             False
100003560             False
100003561             False
100003565             False
100003568             False
…                     …
100041348             False
100043437              True
100043453             False
100044068             False
100044161             False

[94 rows x 22 columns]
```

[13]:
```
weekly_counts
```

[13]:
```
weekno         1       2       3        4       5       6       7       8   \
id
100000671   339.0   309.0   163.0    362.0   241.0   209.0   177.0   349.0
100003560   759.0   693.0   571.0   1046.0   713.0   583.0   676.0   641.0
100003561  4052.0  3076.0  2880.0   4347.0  2346.0  2044.0  2853.0  3153.0
100003565  1074.0  1233.0   926.0   1265.0   972.0   866.0   879.0  1022.0
```

```
100003568  1952.0  1980.0  1608.0   2279.0  1505.0  1468.0  1518.0  1551.0
…             …     …       …        …       …       …       …
100041348   554.0   558.0   207.0    775.0   386.0   344.0   173.0   320.0
100043437   375.0   396.0   372.0    555.0   331.0   304.0   200.0   495.0
100043453    55.0    63.0    55.0     49.0    44.0    57.0    54.0    66.0
100044068  2382.0  1118.0  1380.0   2470.0   740.0   806.0   960.0  1787.0
100044161  7239.0  8754.0  5665.0  11735.0  8044.0  7430.0  6268.0  9910.0

weekno          9       10  …       43       44       45       46       47  \
id                          …
100000671   212.0   220.0  …    300.0    324.0    287.0    309.0    341.0
100003560   763.0   725.0  …    875.0    935.0   1028.0    929.0    653.0
100003561  2686.0  2938.0  …   3623.0   3679.0   3598.0   3823.0   2867.0
100003565  1058.0   902.0  …   1287.0   1106.0   1464.0   1147.0   1028.0
100003568  1716.0  1716.0  …   2009.0   2256.0   2314.0   2347.0   1855.0
…             …     …  …       …        …        …        …        …
100041348   326.0   360.0  …    679.0    716.0    484.0    528.0    519.0
100043437   375.0   468.0  …    716.0    602.0    659.0    605.0    579.0
100043453    60.0    73.0  …     74.0     75.0     66.0     56.0     59.0
100044068   697.0   879.0  …   1612.0   1530.0   1958.0   1939.0   1677.0
100044161  6116.0  6785.0  …  12027.0  11543.0  11738.0  11966.0  11859.0

weekno          48      49      50      51      52
id
100000671   227.0   195.0   246.0   223.0   256.0
100003560   525.0   585.0   865.0   733.0   511.0
100003561  2482.0  1689.0  2819.0  2555.0  2566.0
100003565   919.0   752.0   966.0   915.0   693.0
100003568  1629.0  1257.0  1888.0  1707.0  1385.0
…             …     …       …       …
100041348   345.0   330.0   276.0   244.0   387.0
100043437   200.0   247.0   371.0   318.0   308.0
100043453    43.0    49.0    44.0    58.0    41.0
100044068  1085.0   522.0   910.0   821.0  1422.0
100044161  4267.0  7008.0  7873.0  7619.0  4081.0

[94 rows x 52 columns]
```

6.2 Calculate AADT at a CCS location

[14]:
```
# AADT can be calculated at CCSs (in the idealization that CCS counts are␣
  ↪perfectly accurate)
year_totals = np.sum(weekly_counts, axis = 1) # totals for the year at each CCS
AADTs = (1/365) * year_totals # average annual daily traffic at each CCS
np.round(AADTs, decimals=1) # round to one decimal place
```

[14]:
```
id
100000671      48.2
100003560     125.6
100003561     556.1
100003565     176.2
100003568     306.3
              …
100041348      78.4
100043437      90.9
100043453       9.5
100044068     292.4
100044161    1480.4
Length: 94, dtype: float64
```

6.3 Estimate A-AADT

[15]:
```
# the average AADT (A-AADT) across the stratum is estimated by the sample mean␣
 ↪of AADTs at the CCSs
# (in this case, pretending that CCSs were sampled uniformly at random from a␣
 ↪larger stratum of interest)

A_AADT_estimate = np.mean(AADTs) # sample mean estimate of A-AADT from all CCSs
np.round(A_AADT_estimate, decimals=2)
```

[15]:
```
np.float64(481.13)
```

6.4 Estimate AADT at a location with an SCS

[16]:
```
# Simulates estimating AADT at a random location where an SCS has been␣
 ↪installed.
# First randomly choose station(s) to be SCS(s) (we will censor most of the␣
 ↪counts at those stations)

N = weekly_counts.shape[0] # size of data
n_s = 1 # the number of pseudo-SCSs at which to make estimates (currently, a␣
 ↪single SCS)
n_c = N - 1 # number of CCSs
SCS_ix = np.random.choice(range(N), size = n_s, replace = False) # indices of␣
 ↪SCSs
CCS_ix = np.setdiff1d(range(N), SCS_ix) # indices of CCSs
T = np.random.choice(np.arange(0,52), size = n_s, replace = True) # the week␣
 ↪each SCS count is observed (different weeks in general)
```

[17]:
```
# construct numpy arrays of covariates and counts

# covariates
X = np.concatenate(
```

```
    (np.array(covariates[[
    'Bicycle_Fa_Bike Lane',
    'Bicycle_Fa_Bike Route',
    'Bicycle_Fa_Cycle Track',
    'Bicycle_Fa_Path',
    'Bicycle_Fa_Trail']]),
    np.array(covariates[[
    'Distance_to_water_area',
    'distanc_sch_coll_uni',
    'lat',
    'long']])), axis=1) # covariates as a numpy array (matrix)

strata = np.array(covariates['Strata']) # for post-stratification, an array of␣
 ↪post-strata names

# counts
y_s = np.array(weekly_counts.iloc[SCS_ix]) # all counts for SCSs (unobserved)
y_c = np.array(weekly_counts.iloc[CCS_ix]) # all counts for CCSs
y_bar_s = (1/365) * np.sum(y_s, axis = 1) # AADTs for SCSs (unobserved)
y_bar_c = (1/365) * np.sum(y_c, axis = 1) # AADTs for CCSs
Y = np.array(y_s)[np.arange(len(SCS_ix)), T] # single week counts for SCSs␣
 ↪(observed)

# the whole shebang
D = (X, y_c, Y, T, SCS_ix) # tuple of all observed data
```

[18]:
```
# estimate expansion factors using all CCSs
# we will estimate the AADT and then compute the actual loss for this␣
 ↪particular sample; in practice we cannot do this because we will not know␣
 ↪the true AADT

# expansion factors
F = define_factor_groups_allgroups(n_s, n_c) # the factor group is just all the␣
 ↪CCSs
w_allgroups = estimate_all_expansion_factors(F, y_c, factor_method =␣
 ↪'averaging') # construct expansion factors for CCSs

# estimated counts
y_hats = estimate_all_aadts_factor(Y = Y, T = T, w = w_allgroups) # estimate␣
 ↪AADT at the SCS

# loss for this sample
rmse_all_groups = np.sqrt(sample_loss(predicted = np.array(y_hats), true =␣
 ↪y_bar_s, loss_func = squared_error_loss)) # find the estimation error␣
 ↪expressed as root-mean-squared-error (RMSE) at that SCS
```

```
pe_all_groups = sample_loss(predicted = np.array(y_hats), true = y_bar_s,␣
 ↪loss_func = proportional_loss) # estimation error expressed as proportional␣
 ↪loss

f'true AADT:{y_bar_s[0]: .1f}; estimated AADT:{y_hats[0]: .1f}; rmse for all␣
 ↪groups:{rmse_all_groups: .3f}; relative error:{pe_all_groups: .3f}'
```

[18]:
```
'true AADT: 126.3; estimated AADT: 132.1; rmse for all groups: 5.782; relative
error: 0.046'
```

[19]:
```
# estimate expansion factors using post-stratification

SCS_strata = strata[SCS_ix] # strata for the SCSs
CCS_strata = strata[CCS_ix] # strata for the CCSs

# define factor group(s) based on post-strata
F = define_factor_groups_poststratification(SCS_strata = SCS_strata, CCS_strata␣
 ↪= CCS_strata)

# estimate expansion factors
w_poststratification = estimate_all_expansion_factors(F, y_c, factor_method =␣
 ↪'averaging')

# estimate AADT(s) at SCS(s)
y_hats = estimate_all_aadts_factor(Y = Y, T = T, w = w_poststratification)

# RMSE under post-stratification
rmse_poststratification = np.sqrt(sample_loss(predicted = np.array(y_hats),␣
 ↪true = y_bar_s, loss_func = squared_error_loss))

# proportional loss under post-stratification
pe_poststratification = sample_loss(predicted = np.array(y_hats), true =␣
 ↪y_bar_s, loss_func = proportional_loss)
f'true AADT:{y_bar_s[0]: .1f}; estimated AADT:{y_hats[0]: .1f}; rmse for all␣
 ↪groups:{rmse_poststratification: .3f}; relative error:{pe_poststratification:
 ↪ .3f}'
```

[19]:
```
'true AADT: 126.3; estimated AADT: 134.5; rmse for all groups: 8.234; relative
error: 0.065'
```

[20]:
```
# expansion factors estimated by nearest-neighbors; examine effect of the␣
 ↪number of nearest neighbors

SCS_X = X[SCS_ix, 1:] # covariate matrix for SCSs
CCS_X = X[CCS_ix, 1:] # covariate matrix for CCSs

K_max = 10
```

```
K_grid = np.arange(1, K_max+1) # range K from 1 to 10 nearest neighbors
rmse_nn = [] # list to store RMSEs
pe_nn = [] # list to store proportional errors
y_hats_list = [] # list to store estimates
for k in K_grid:
    F = define_factor_groups_nearestneighbor(SCS_X, CCS_X, K=k)
    w_nn = estimate_all_expansion_factors(F, y_c, factor_method = 'averaging')
    y_hats = estimate_all_aadts_factor(Y = Y, T = T, w = w_nn)
    y_hats_list.append(y_hats)
    rmse_nn.append(np.sqrt(sample_loss(predicted = np.array(y_hats), true =␣
 ↪y_bar_s, loss_func = squared_error_loss)))
    pe_nn.append(sample_loss(predicted = np.array(y_hats), true = y_bar_s,␣
 ↪loss_func = proportional_loss))

columns = ['K', 'rmse', 'relative error']

print(f'true AADT:{y_bar_s[0]: .1f}')
for i in range(K_max):
    print(f'K:{i+1} | estimate:{y_hats_list[i][0]: .1f} | rmse:{rmse_nn[i]: .
 ↪3f} pe:{pe_nn[i]: .3f}')
```

```
true AADT: 126.3
K:1 | estimate: 125.8 | rmse: 0.516 pe: 0.004
K:2 | estimate: 124.4 | rmse: 1.890 pe: 0.015
K:3 | estimate: 121.2 | rmse: 5.033 pe: 0.040
K:4 | estimate: 120.4 | rmse: 5.921 pe: 0.047
K:5 | estimate: 119.8 | rmse: 6.454 pe: 0.051
K:6 | estimate: 119.5 | rmse: 6.808 pe: 0.054
K:7 | estimate: 120.4 | rmse: 5.856 pe: 0.046
K:8 | estimate: 121.1 | rmse: 5.147 pe: 0.041
K:9 | estimate: 121.7 | rmse: 4.588 pe: 0.036
K:10 | estimate: 122.1 | rmse: 4.143 pe: 0.033
```

6.5 Estimate the risk of AADT estimation at a randomly located SCS using cross-validation

[21]:
```
# 5-fold cross-validation estimates when using all CCSs as the factor group and␣
 ↪the first week of counts are observed at SCSs

#rmse
rmse_risk_est = K_fold_CV_factorgroups("all", K = 5, y = np.
 ↪array(weekly_counts), factor_method = "averaging", T = np.repeat(int(1), N)␣
 ↪, X = None, loss_func = proportional_loss)
print(f'rmse risk estimate for all groups: {rmse_risk_est[0]: .3f}; standard␣
 ↪error of risk estimate: {rmse_risk_est[1]: .3f}')

#proportional error
```

```
pe_risk_est = K_fold_CV_factorgroups("all", K = 5, y = np.array(weekly_counts),␣
 ↪factor_method = "averaging", T = np.repeat(int(1), N) , X = None, loss_func␣
 ↪= proportional_loss)
print(f'proportional error risk estimate for all groups: {pe_risk_est[0]: .3f};␣
 ↪standard error of risk estimate: {pe_risk_est[1]: .3f}')
```

```
rmse risk estimate for all groups:  0.152; standard error of risk estimate:
0.028
proportional error risk estimate for all groups:  0.154; standard error of risk
estimate:  0.014
```

[22]:
```
# 5-fold cross-validation estimates for post-stratification

#rmse
rmse_risk_est = K_fold_CV_factorgroups("poststratification", K = 5, y = np.
 ↪array(weekly_counts), factor_method = "averaging", T = np.repeat(int(1), N)␣
 ↪, X = np.expand_dims(strata, 1), loss_func = proportional_loss)
print(f'rmse risk estimate for all poststratification: {rmse_risk_est[0]: .3f};␣
 ↪standard error of risk estimate: {rmse_risk_est[1]: .3f}')

#proportional error
pe_risk_est = K_fold_CV_factorgroups("all", K = 5, y = np.array(weekly_counts),␣
 ↪factor_method = "averaging", T = np.repeat(int(1), N) , X = np.
 ↪expand_dims(strata, 1), loss_func = proportional_loss)
print(f'proportional error risk estimate for poststratification:␣
 ↪{pe_risk_est[0]: .3f}; standard error of risk estimate: {pe_risk_est[1]: .
 ↪3f}')
```

```
rmse risk estimate for all poststratification:  0.160; standard error of risk
estimate:  0.026
proportional error risk estimate for poststratification:  0.153; standard error
of risk estimate:  0.018
```

```
/var/folders/zt/byjw_s1s4g5b01_cjmqw49480000gn/T/ipykernel_29380/3945395430.py:4
0: UserWarning: SCS stratum ['Other'] contains no CCSs: using default factor
group containing all CCSs.
  warnings.warn(f'SCS stratum {SCS_stratum} contains no CCSs: using default
factor group containing all CCSs.')
/var/folders/zt/byjw_s1s4g5b01_cjmqw49480000gn/T/ipykernel_29380/3945395430.py:4
0: UserWarning: SCS stratum ['Bike lane'] contains no CCSs: using default factor
group containing all CCSs.
  warnings.warn(f'SCS stratum {SCS_stratum} contains no CCSs: using default
factor group containing all CCSs.')
```

[23]:
```
# 5-fold cross-validation risk estimates 2 nearest-neighbors

#rmse
```

```
rmse_risk_est = K_fold_CV_factorgroups("nearest_neighbors", K = 5, y = np.
 ↪array(weekly_counts), factor_method = "averaging", T = np.repeat(int(1), N),␣
 ↪X = X, k_nn = 2, loss_func = proportional_loss)
print(f'rmse risk estimate for 2 nearest neighbors: {rmse_risk_est[0]: .3f};␣
 ↪standard error of risk estimate: {rmse_risk_est[1]: .3f}')

#proportional error
pe_risk_est = K_fold_CV_factorgroups("nearest_neighbors", K = 5, y = np.
 ↪array(weekly_counts), factor_method = "averaging", T = np.repeat(int(1), N),␣
 ↪X = X, k_nn = 2, loss_func = proportional_loss)
print(f'proportional error risk estimate for 2 nearest neighbors:␣
 ↪{pe_risk_est[0]: .3f}; standard error of risk estimate: {pe_risk_est[1]: .
 ↪3f}')
```

```
rmse risk estimate for 2 nearest neighbors:  0.168; standard error of risk
estimate:  0.031
proportional error risk estimate for 2 nearest neighbors:  0.154; standard error
of risk estimate:  0.020
```

[24]:
```
# risk estimate for 8 nearest neighbors

#rmse
rmse_risk_est = K_fold_CV_factorgroups("nearest_neighbors", K = 5, y = np.
 ↪array(weekly_counts), factor_method = "averaging", T = np.repeat(int(1), N),␣
 ↪X = X, k_nn = 2, loss_func = proportional_loss)
print(f'rmse risk estimate for 8 nearest neighbors: {rmse_risk_est[0]: .3f};␣
 ↪standard error of risk estimate: {rmse_risk_est[1]: .3f}')

#proportional error
pe_risk_est = K_fold_CV_factorgroups("nearest_neighbors", K = 5, y = np.
 ↪array(weekly_counts), factor_method = "averaging", T = np.repeat(int(1), N),␣
 ↪X = X, k_nn = 2, loss_func = proportional_loss)
print(f'proportional error risk estimate for 8 nearest neighbors:␣
 ↪{pe_risk_est[0]: .3f}; standard error of risk estimate: {pe_risk_est[1]: .
 ↪3f}')
```

```
rmse risk estimate for 8 nearest neighbors:  0.154; standard error of risk
estimate:  0.036
proportional error risk estimate for 8 nearest neighbors:  0.154; standard error
of risk estimate:  0.020
```

6.6 Estimate AADT at a location with no count station

[25]:
```
# there are various ways to estimate AADT at a location r with no CCS or SCS
# cross-validation or other means could also be used to estimate the␣
 ↪uncertainty, but the uncertainty is only valid if the location was selected␣
 ↪at random, which is not true
```

```
# using A-AADT estimate for the stratum
est_r_AAADT = np.mean(AADTs) # estimate that locations AADT as the average␣
 ↪A-AADT across the stratum

# using post-stratification
poststratum_r = np.array(['University']) # define a post-stratum in which that␣
 ↪location lives
est_r_poststratification = np.mean(AADTs[strata == poststratum_r])


f'estimate using A-AADT: {est_r_AAADT: .1f}; estimate using poststratification:␣
 ↪{est_r_poststratification: .1f}'
```

[25]: 'estimate using A-AADT: 481.1; estimate using poststratification: 560.2'