LSCOG 2050 LRTP - Base Year SE Data Development

Pukar Bhandari

2025-07-05

Table of contents

# 1. Set up the environment

This section establishes the computational environment for processing socioeconomic data inputs for the Lower Savannah Council of Governments regional travel demand model using both R and Python platforms.

## 1.1 Install and load packages

The package installation process incorporates essential libraries for comprehensive geospatial data analysis.

The R environment utilizes the pacman package manager to streamline the installation of multiple packages simultaneously, including tidyverse for data manipulation, sf for spatial data handling, tidycensus for Census Bureau data access, and lehdr for Longitudinal Employer-Household Dynamics data retrieval.

The Python environment focuses on core data science libraries including pandas for data manipulation, geopandas for spatial analysis, and pygris for Census data queries. These packages form the analytical backbone for processing demographic, employment, and geographic data required for travel demand modeling.

#### R

# Install and load the pacman package  
if (!require("pacman")) {  
 install.packages("pacman")  
}

Loading required package: pacman

library("pacman") # Package Management Tool CRAN v0.5.1  
  
# Install and load multiple desired packages at once  
pacman::p\_load(  
 tidyverse, # Easily Install and Load the 'Tidyverse'  
 sf, # Simple Features for R  
 sfdep, # Spatial Dependence for Simple Features  
 tidycensus, # Load US Census Boundary and Attribute Data  
 lehdr, # Grab Longitudinal Employer-Household Dynamics (LEHD)  
 arcgis, # ArcGIS Location Services Meta-Package  
 mapview, # Interactive Viewing of Spatial Data  
 RColorBrewer, # Color Palettes  
 janitor # Simple Tools for Examining and Cleaning Dirty Data  
)

#### Python

# Install required packages if not available  
# pip install numpy pandas geopandas matplotlib seaborn folium pathlib zipfile requests urllib warnings pygris  
  
# Data and Visualization  
import numpy as np  
import pandas as pd  
import geopandas as gpd  
import matplotlib.pyplot as plt  
import seaborn as sns  
import folium  
  
# Query online data  
from pathlib import Path  
import zipfile  
import requests  
import urllib.parse  
import warnings  
warnings.filterwarnings('ignore')  
  
# Census data query  
from pygris import blocks, block\_groups  
from pygris.helpers import validate\_state, validate\_county  
from pygris.data import get\_census, get\_lodes

## 1.2 Set global options and parameters

Configuration settings optimize performance and establish spatial consistency. The tigris cache prevents redundant TIGER/Line shapefile downloads. The South Carolina State Plane coordinate system (EPSG:3361) serves as the standard projection for accurate GIS operations

#### R

# Set options  
options(tigris\_use\_cache = TRUE) # cache tiger/line shapefile for future use  
tmap::tmap\_mode("view") # interactive thematic map viewing

ℹ tmap mode set to "view".

# set project crs  
project\_crs <- "EPSG:3361"

#### Python

# Set project CRS  
project\_crs = "EPSG:3361"

## 1.3 Set census API key

API authentication enables access to detailed demographic and economic datasets from the Census Bureau. The key configuration supports both R and Python environments for automated data retrieval workflows.

💡 **Need a Census API key?** Get one for free at [census.gov/developers](https://www.census.gov/developers/)

#### R

# Set your API key into environment  
tidycensus::census\_api\_key("your\_api\_key\_here", install = TRUE)

#### Python

# Set your API key into environment  
import os  
os.environ['CENSUS\_API\_KEY'] = 'your\_api\_key\_here'

## 1.4 Project folder

The centralized directory structure organizes input data, processing files, and model outputs. The standardized root folder path ensures consistent file management across computing environments and team members.

#### R

# Set your main data folder  
root <- "M:/MA\_Project/SC\_LSCOG LRTP"

#### Python

# Set your main data folder  
root = "M:/MA\_Project/SC\_LSCOG LRTP"

# 2. Define study area

This section defines the geographic extent of the Lower Savannah Council of Governments region and loads the Traffic Analysis Zone (TAZ) geometry for spatial analysis.

## 2.1 Define state and counties

The study area encompasses six counties within South Carolina: Aiken, Allendale, Bamberg, Barnwell, Calhoun, and Orangeburg. These counties constitute the LSCOG planning region for travel demand modeling purposes.

#### R

# Define state abbreviation and county names  
state\_abb <- "SC"  
county\_names <- c(  
 "Aiken",  
 "Allendale",  
 "Bamberg",  
 "Barnwell",  
 "Calhoun",  
 "Orangeburg"  
)

#### Python

# Define state abbreviation and county names  
state\_abb = "SC"  
county\_names = [  
 "Aiken",  
 "Allendale",  
 "Bamberg",  
 "Barnwell",  
 "Calhoun",  
 "Orangeburg"  
]

FIPS code conversion translates state abbreviations and county names into standardized Federal Information Processing Standard codes. These codes enable consistent data retrieval across census datasets and ensure proper geographic matching with demographic and economic data sources.

#### R

# converting state abbreviation code to FIPS code  
state\_fips <- tidycensus:::validate\_state(state = state\_abb)  
county\_fips <- vapply(  
 county\_names,  
 function(x) tidycensus:::validate\_county(state = state\_abb, county = x),  
 character(1)  
)  
  
# converting County Names to FIPS code  
fips\_codes <- paste(state\_fips, county\_fips, sep = "")  
fips\_codes

[1] "45003" "45005" "45009" "45011" "45017" "45075"

#### Python

# Converting state abbreviation code to FIPS code  
state\_fips = validate\_state(state\_abb)

Using FIPS code '45' for input 'SC'

# Converting County Names to FIPS code  
county\_fips = [  
 validate\_county(state\_fips, county)  
 for county in county\_names  
]

Using FIPS code '003' for input 'Aiken'  
Using FIPS code '005' for input 'Allendale'  
Using FIPS code '009' for input 'Bamberg'  
Using FIPS code '011' for input 'Barnwell'  
Using FIPS code '017' for input 'Calhoun'  
Using FIPS code '075' for input 'Orangeburg'

# Converting County Names to FIPS code  
fips\_codes = [f"{state\_fips}{county}" for county in county\_fips]  
fips\_codes

['45003', '45005', '45009', '45011', '45017', '45075']

## 2.2 Load TAZ geometry

The TAZ shapefile provides the fundamental spatial framework for travel demand modeling. The geometry is loaded from the TDM exports geodatabase and filtered to include only zones within the six-county study area using FIPS code matching.

Coordinate transformation converts the TAZ geometry to the project’s standard coordinate reference system (EPSG:3361) for accurate spatial calculations. The attribute selection retains essential fields including TAZ identifiers, area measurements, area type classifications, and county assignments.

#### R

# Load TAZ Shapefile  
lscog\_taz <- sf::read\_sf(  
 file.path(root, "GIS/data\_temp/TDM Exports/TDM\_Exports.gdb"),  
 query = paste0(  
 "SELECT \* FROM \"SE\_2019\_AD\_10\_30\_2023\" WHERE countyID IN (",  
 paste0("'", fips\_codes, "'", collapse = ", "),  
 ")"  
 )  
) |>  
 sf::st\_transform(project\_crs) |>  
 dplyr::select(  
 ID,  
 Area,  
 Acres,  
 TAZ\_ID = TAZ\_IDs,  
 AREA\_TYPE,  
 COUNTY,  
 COUNTYID = countyID  
 )  
  
str(lscog\_taz)

sf [585 × 8] (S3: sf/tbl\_df/tbl/data.frame)  
 $ ID : int [1:585] 9050130 9050132 75050131 75050045 75050182 75050183 75050172 75050204 75050200 75050201 ...  
 $ Area : num [1:585] 13.31 39.19 9.03 15.52 17.52 ...  
 $ Acres : num [1:585] 8518 25084 5778 9934 11216 ...  
 $ TAZ\_ID : int [1:585] 9050130 9050132 75050131 75050045 75050182 75050183 75050172 75050204 75050200 75050201 ...  
 $ AREA\_TYPE: chr [1:585] "RURAL" "RURAL" "RURAL" "RURAL" ...  
 $ COUNTY : chr [1:585] "Bamberg SC" "Bamberg SC" "Orangeburg S" "Orangeburg S" ...  
 $ COUNTYID : int [1:585] 45009 45009 45075 45075 45075 45075 45075 45075 45075 45075 ...  
 $ SHAPE :sfc\_MULTIPOLYGON of length 585; first list element: List of 1  
 ..$ :List of 1  
 .. ..$ : num [1:533, 1:2] 2046194 2046134 2045670 2044097 2043629 ...  
 ..- attr(\*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"  
 - attr(\*, "sf\_column")= chr "SHAPE"  
 - attr(\*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA  
 ..- attr(\*, "names")= chr [1:7] "ID" "Area" "Acres" "TAZ\_ID" ...

#### Python

# Load TAZ Shapefile  
lscog\_taz = gpd.read\_file(  
 Path(root) / "GIS/data\_temp/TDM Exports/TDM\_Exports.gdb",  
 layer="SE\_2019\_AD\_10\_30\_2023",  
 where=f"countyID IN ({', '.join([f"'{fips}'" for fips in fips\_codes])})"  
)  
  
lscog\_taz = lscog\_taz.to\_crs(project\_crs)  
  
lscog\_taz = lscog\_taz.rename(  
 columns={  
 'TAZ\_IDs': 'TAZ\_ID',  
 'countyID': 'COUNTYID'  
})[['ID', 'Area', 'Acres', 'TAZ\_ID', 'AREA\_TYPE', 'COUNTY', 'COUNTYID', 'geometry']]  
  
lscog\_taz.info()

<class 'geopandas.geodataframe.GeoDataFrame'>  
RangeIndex: 585 entries, 0 to 584  
Data columns (total 8 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 ID 585 non-null int32   
 1 Area 585 non-null float64   
 2 Acres 585 non-null float64   
 3 TAZ\_ID 585 non-null int32   
 4 AREA\_TYPE 585 non-null object   
 5 COUNTY 585 non-null object   
 6 COUNTYID 585 non-null int32   
 7 geometry 585 non-null geometry  
dtypes: float64(2), geometry(1), int32(3), object(2)  
memory usage: 29.8+ KB

The interactive map visualization displays the TAZ structure colored by county, providing spatial context for the analysis area and enabling quality assurance of the geometric data loading process.

#### R

# Create interactive map  
mapview::mapview(lscog\_taz, zcol = "COUNTY", lwd = 1.6, map.types = "CartoDB.Positron", col.regions = RColorBrewer::brewer.pal(6, "Dark2"))

#### Python

# Create interactive map  
lscog\_taz.explore(column="COUNTY", categorical=True, legend=True, tiles="CartoDB positron", zoom\_start=8)

# 3. Fetch raw data

This section retrieves demographic, economic, and employment data from multiple Census Bureau sources at the appropriate geographic scales for travel demand modeling.

## 3.1 2020 Decennial census

The 2020 Decennial Census provides population and housing data at the census block level, offering the finest spatial resolution for demographic analysis. Population variables include total population, group quarters population, and household population derived by subtraction. Housing variables encompass total dwelling units and household counts by size categories.

Household size distributions are consolidated into four categories: 1-person, 2-person, 3-person, and 4-or-more-person households. The 4-or-more category aggregates larger household sizes to simplify model implementation while maintaining essential demographic stratification for trip generation analysis.

#### R

# Define variables to download  
dec\_variables <- c(  
 TOTPOP = "P1\_001N", # Total Population  
 GQPOP = "P18\_001N", # Population living in Group Quarters  
 DU = "H1\_001N", # Dwelling Units  
 HH\_1 = "H9\_002N", # 1-person household  
 HH\_2 = "H9\_003N", # 2-person household  
 HH\_3 = "H9\_004N", # 3-person household  
 # HH\_4 = "H9\_005N", # 4-person household  
 # HH\_5 = "H9\_006N", # 5-person household  
 # HH\_6 = "H9\_007N", # 6-person household  
 # HH\_7 = "H9\_008N", # 7-or-more-person household  
 HH = "H9\_001N" # Total Number of Households  
 )  
  
# Load Population and Household Data  
lscog\_dec <- tidycensus::get\_decennial(  
 year = 2020,  
 sumfile = "dhc",  
 geography = "block",  
 state = state\_abb,  
 county = county\_names,  
 output = "wide",  
 cb = FALSE,  
 geometry = TRUE,  
 keep\_geo\_vars = TRUE,  
 variables = dec\_variables  
) |>  
 sf::st\_transform(project\_crs) |>  
 dplyr::mutate(  
 HHPOP = TOTPOP - GQPOP,  
 HH\_4 = HH - (HH\_1 + HH\_2 + HH\_3)  
 ) |>  
 dplyr::select(GEOID, TOTPOP, GQPOP, HHPOP, HH, HH\_1, HH\_2, HH\_3, HH\_4, DU)

Getting data from the 2020 decennial Census

Using the Demographic and Housing Characteristics File

Note: 2020 decennial Census data use differential privacy, a technique that  
introduces errors into data to preserve respondent confidentiality.  
ℹ Small counts should be interpreted with caution.  
ℹ See https://www.census.gov/library/fact-sheets/2021/protecting-the-confidentiality-of-the-2020-census-redistricting-data.html for additional guidance.  
This message is displayed once per session.

str(lscog\_dec)

sf [13,961 × 11] (S3: sf/tbl\_df/tbl/data.frame)  
 $ GEOID : chr [1:13961] "450179504004051" "450179504003011" "450179502011045" "450179504001020" ...  
 $ TOTPOP : num [1:13961] 0 0 54 0 13 10 0 6 0 18 ...  
 $ GQPOP : num [1:13961] 0 0 4 0 0 0 0 0 0 0 ...  
 $ HHPOP : num [1:13961] 0 0 50 0 13 10 0 6 0 18 ...  
 $ HH : num [1:13961] 0 0 16 0 4 3 0 4 0 0 ...  
 $ HH\_1 : num [1:13961] 0 0 3 0 0 3 0 0 0 0 ...  
 $ HH\_2 : num [1:13961] 0 0 1 0 3 0 0 1 0 0 ...  
 $ HH\_3 : num [1:13961] 0 0 3 0 0 0 0 2 0 0 ...  
 $ HH\_4 : num [1:13961] 0 0 9 0 1 0 0 1 0 0 ...  
 $ DU : num [1:13961] 0 0 18 0 4 8 0 4 0 1 ...  
 $ geometry:sfc\_MULTIPOLYGON of length 13961; first list element: List of 1  
 ..$ :List of 1  
 .. ..$ : num [1:13, 1:2] 2084301 2084461 2084474 2084523 2084529 ...  
 ..- attr(\*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"  
 - attr(\*, "sf\_column")= chr "geometry"  
 - attr(\*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA  
 ..- attr(\*, "names")= chr [1:10] "GEOID" "TOTPOP" "GQPOP" "HHPOP" ...

#### Python

# Define variables to download  
dec\_variables = {  
 'P1\_001N': 'TOTPOP', # Total Population  
 'P18\_001N': 'GQPOP', # Population living in Group Quarters  
 'H1\_001N': 'DU', # Dwelling Units  
 'H9\_002N': 'HH\_1', # 1-person household  
 'H9\_003N': 'HH\_2', # 2-person household  
 'H9\_004N': 'HH\_3', # 3-person household  
 # 'H9\_005N': 'HH\_4', # 4-person household  
 # 'H9\_006N': 'HH\_5', # 5-person household  
 # 'H9\_007N': 'HH\_6', # 6-person household  
 # 'H9\_008N': 'HH\_7', # 7-or-more-person household  
 'H9\_001N': 'HH' # Total Number of Households  
}  
  
# get census block geometries  
lscog\_cb = blocks(  
 state=state\_fips,  
 county=county\_fips,  
 year=2020,  
 cache=True  
)  
  
# Download decennial census data at block level  
lscog\_dec = get\_census(  
 dataset="dec/dhc",  
 year=2020,  
 variables=list(dec\_variables.keys()),  
 params={  
 "for": f"block:\*",  
 "in": f"state:{state\_fips} county:{','.join(county\_fips)}"  
 },  
 return\_geoid=True,  
 guess\_dtypes=True  
)  
  
# join data to geometry  
lscog\_dec = lscog\_cb[['GEOID20', 'geometry']].merge(lscog\_dec, left\_on = "GEOID20", right\_on = "GEOID")  
  
# Rename columns  
lscog\_dec = lscog\_dec.rename(columns=dec\_variables)  
  
# Transform CRS  
lscog\_dec = lscog\_dec.to\_crs(project\_crs)  
  
# Calculate derived variables  
lscog\_dec['HHPOP'] = lscog\_dec['TOTPOP'] - lscog\_dec['GQPOP']  
lscog\_dec['HH\_4'] = lscog\_dec['HH'] - (  
 lscog\_dec['HH\_1'] + lscog\_dec['HH\_2'] + lscog\_dec['HH\_3']  
)  
  
# Select final columns  
lscog\_dec = lscog\_dec[['GEOID', 'TOTPOP', 'GQPOP', 'HHPOP',  
 'HH', 'HH\_1', 'HH\_2', 'HH\_3', 'HH\_4', 'DU', 'geometry']]  
  
lscog\_dec.info()

<class 'geopandas.geodataframe.GeoDataFrame'>  
RangeIndex: 13961 entries, 0 to 13960  
Data columns (total 11 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 GEOID 13961 non-null object   
 1 TOTPOP 13961 non-null int64   
 2 GQPOP 13961 non-null int64   
 3 HHPOP 13961 non-null int64   
 4 HH 13961 non-null int64   
 5 HH\_1 13961 non-null int64   
 6 HH\_2 13961 non-null int64   
 7 HH\_3 13961 non-null int64   
 8 HH\_4 13961 non-null int64   
 9 DU 13961 non-null int64   
 10 geometry 13961 non-null geometry  
dtypes: geometry(1), int64(9), object(1)  
memory usage: 1.2+ MB

## 3.2 2020 ACS estimates

The American Community Survey 5-year estimates provide household income data at the block group level. Income categories are aggregated into three broad ranges: under $15,000, $15,000-$49,999, and $50,000 and above. This stratification aligns with travel behavior research indicating distinct mobility patterns across income levels.

The block group geography represents the finest spatial resolution available for ACS income data, providing sufficient detail for socioeconomic modeling while maintaining statistical reliability through the 5-year aggregation period.

#### R

# Define variables to download  
acs\_variables <- c(  
 INC\_CAT\_02 = "B19001\_002", # Less than $10,000  
 INC\_CAT\_03 = "B19001\_003", # $10,000 to $14,999  
 INC\_CAT\_04 = "B19001\_004", # $15,000 to $19,999  
 INC\_CAT\_05 = "B19001\_005", # $20,000 to $24,999  
 INC\_CAT\_06 = "B19001\_006", # $25,000 to $29,999  
 INC\_CAT\_07 = "B19001\_007", # $30,000 to $34,999  
 INC\_CAT\_08 = "B19001\_008", # $35,000 to $39,999  
 INC\_CAT\_09 = "B19001\_009", # $40,000 to $44,999  
 INC\_CAT\_10 = "B19001\_010", # $45,000 to $49,999  
 # INC\_CAT\_11 = "B19001\_011", # $50,000 to $59,999  
 # INC\_CAT\_12 = "B19001\_012", # $60,000 to $74,999  
 # INC\_CAT\_13 = "B19001\_013", # $75,000 to $99,999  
 # INC\_CAT\_14 = "B19001\_014", # $100,000 to $124,999  
 # INC\_CAT\_15 = "B19001\_015", # $125,000 to $149,999  
 # INC\_CAT\_16 = "B19001\_016", # $150,000 to $199,999  
 # INC\_CAT\_17 = "B19001\_017", # $200,000 or more  
 INC\_CAT\_01 = "B19001\_001" # Total  
 )  
  
# Load Household Income Data  
lscog\_acs <- tidycensus::get\_acs(  
 year = 2020,  
 survey = "acs5",  
 geography = "block group",  
 state = state\_fips,  
 county = county\_fips,  
 output = "wide",  
 cb = FALSE,  
 geometry = TRUE,  
 variables = acs\_variables  
) |>  
 sf::st\_transform(project\_crs) |>  
 dplyr::mutate(  
 INC\_14999 = INC\_CAT\_02E + INC\_CAT\_03E,  
 INC\_49999 = INC\_CAT\_04E +  
 INC\_CAT\_05E +  
 INC\_CAT\_06E +  
 INC\_CAT\_07E +  
 INC\_CAT\_08E +  
 INC\_CAT\_09E +  
 INC\_CAT\_10E,  
 INC\_50000 = INC\_CAT\_01E - (INC\_14999 + INC\_49999)  
 ) |>  
 dplyr::select(GEOID, INC\_TOTAL = INC\_CAT\_01E, INC\_14999, INC\_49999, INC\_50000)

Getting data from the 2016-2020 5-year ACS

str(lscog\_acs)

Classes 'sf' and 'data.frame': 262 obs. of 6 variables:  
 $ GEOID : chr "450030207011" "450030212052" "450030212051" "450030213001" ...  
 $ INC\_TOTAL: num 467 949 857 612 1191 ...  
 $ INC\_14999: num 82 0 21 103 176 95 56 37 37 77 ...  
 $ INC\_49999: num 171 256 250 129 303 240 302 158 344 247 ...  
 $ INC\_50000: num 214 693 586 380 712 438 723 742 377 649 ...  
 $ geometry :sfc\_MULTIPOLYGON of length 262; first list element: List of 1  
 ..$ :List of 1  
 .. ..$ : num [1:335, 1:2] 1701402 1701441 1701616 1701789 1701819 ...  
 ..- attr(\*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"  
 - attr(\*, "sf\_column")= chr "geometry"  
 - attr(\*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA  
 ..- attr(\*, "names")= chr [1:5] "GEOID" "INC\_TOTAL" "INC\_14999" "INC\_49999" ...

#### Python

# Define variables to download  
acs\_variables = {  
 'B19001\_002E': 'INC\_CAT\_02', # Less than $10,000  
 'B19001\_003E': 'INC\_CAT\_03', # $10,000 to $14,999  
 'B19001\_004E': 'INC\_CAT\_04', # $15,000 to $19,999  
 'B19001\_005E': 'INC\_CAT\_05', # $20,000 to $24,999  
 'B19001\_006E': 'INC\_CAT\_06', # $25,000 to $29,999  
 'B19001\_007E': 'INC\_CAT\_07', # $30,000 to $34,999  
 'B19001\_008E': 'INC\_CAT\_08', # $35,000 to $39,999  
 'B19001\_009E': 'INC\_CAT\_09', # $40,000 to $44,999  
 'B19001\_010E': 'INC\_CAT\_10', # $45,000 to $49,999  
 # 'B19001\_011E': 'INC\_CAT\_11', # $50,000 to $59,999  
 # 'B19001\_012E': 'INC\_CAT\_12', # $60,000 to $74,999  
 # 'B19001\_013E': 'INC\_CAT\_13', # $75,000 to $99,999  
 # 'B19001\_014E': 'INC\_CAT\_14', # $100,000 to $124,999  
 # 'B19001\_015E': 'INC\_CAT\_15', # $125,000 to $149,999  
 # 'B19001\_016E': 'INC\_CAT\_16', # $150,000 to $199,999  
 # 'B19001\_017E': 'INC\_CAT\_17', # $200,000 or more  
 'B19001\_001E': 'INC\_CAT\_01' # Total  
}  
  
# get blockgroup geometries  
lscog\_bg = block\_groups(  
 state=state\_fips,  
 county=county\_fips,  
 year=2020,  
 cache=True  
)  
  
# Download household income data at block group level  
lscog\_acs = get\_census(  
 dataset="acs/acs5",  
 year=2020,  
 variables=list(acs\_variables.keys()),  
 params={  
 "for": f"block group:\*",  
 "in": f"state:{state\_fips} county:{','.join(county\_fips)}"  
 },  
 return\_geoid=True,  
 guess\_dtypes=True  
)  
  
# join data to geometry  
lscog\_acs = lscog\_bg[['GEOID', 'geometry']].merge(lscog\_acs, on = "GEOID")  
  
# Rename columns  
lscog\_acs = lscog\_acs.rename(columns=acs\_variables)  
  
# Transform CRS  
lscog\_acs = lscog\_acs.to\_crs(project\_crs)  
  
# Calculate derived variables  
lscog\_acs['INC\_14999'] = lscog\_acs['INC\_CAT\_02'] + lscog\_acs['INC\_CAT\_03']  
lscog\_acs['INC\_49999'] = (  
 lscog\_acs['INC\_CAT\_04'] +  
 lscog\_acs['INC\_CAT\_05'] +  
 lscog\_acs['INC\_CAT\_06'] +  
 lscog\_acs['INC\_CAT\_07'] +  
 lscog\_acs['INC\_CAT\_08'] +  
 lscog\_acs['INC\_CAT\_09'] +  
 lscog\_acs['INC\_CAT\_10']  
)  
lscog\_acs['INC\_50000'] = lscog\_acs['INC\_CAT\_01'] - (  
 lscog\_acs['INC\_14999'] + lscog\_acs['INC\_49999']  
)  
  
# Select final columns  
lscog\_acs = lscog\_acs.rename(columns={'INC\_CAT\_01': 'INC\_TOTAL'})  
lscog\_acs = lscog\_acs[['GEOID', 'INC\_TOTAL', 'INC\_14999', 'INC\_49999', 'INC\_50000', 'geometry'  
]]  
  
lscog\_acs.info()

<class 'geopandas.geodataframe.GeoDataFrame'>  
RangeIndex: 262 entries, 0 to 261  
Data columns (total 6 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 GEOID 262 non-null object   
 1 INC\_TOTAL 262 non-null int64   
 2 INC\_14999 262 non-null int64   
 3 INC\_49999 262 non-null int64   
 4 INC\_50000 262 non-null int64   
 5 geometry 262 non-null geometry  
dtypes: geometry(1), int64(4), object(1)  
memory usage: 12.4+ KB

## 3.3 2019 LEHD data

The Longitudinal Employer-Household Dynamics Workplace Area Characteristics data provides employment counts by industry sector at the census block level. Employment categories follow the North American Industry Classification System and are aggregated into transportation-relevant sectors including retail, services, manufacturing, and public administration.

The 2019 reference year represents pre-pandemic employment patterns, providing a stable baseline for long-term transportation planning. Employment data at the block level enables precise spatial allocation of work destinations within the travel demand model framework.

#### R

# Download LEHD WAC data at block level  
lscog\_emp <- lehdr::grab\_lodes(  
 version = "LODES8",  
 state = tolower(state\_abb),  
 lodes\_type = "wac",  
 segment = "S000",  
 job\_type = "JT00",  
 year = 2019,  
 state\_part = "",  
 agg\_geo = "block",  
 use\_cache = TRUE  
) |>  
 dplyr::filter(grepl(  
 paste("^(", paste(fips\_codes, collapse = "|"), ")", sep = ""),  
 w\_geocode  
 )) |>  
 # check the documentation at: https://lehd.ces.census.gov/data/lodes/LODES8/LODESTechDoc8.0.pdf  
 dplyr::mutate(  
 GEOID = as.character(w\_geocode),  
 TOTAL\_EMP = C000, # Total Employment  
 AGR\_FOR\_FI = CNS01, # Agricultural, forestry, and fishing employment  
 MINING = CNS02, # Mining employment  
 CONSTRUCTI = CNS04, # Construction employment  
 MANUFACTUR = CNS05, # Manufacturing employment  
 TRANSP\_COM = CNS08 + CNS09, # Transportation, communication employment  
 WHOLESALE = CNS06, # Wholesale employment  
 RETAIL = CNS07, # Retail employment  
 FIRE = CNS10 + CNS11, # Finance / Insurance / Real Estate employment  
 SERVICES = CNS03 +  
 CNS12 +  
 CNS13 +  
 CNS14 + # Service employment  
 CNS15 +  
 CNS16 +  
 CNS17 +  
 CNS18 +  
 CNS19,  
 PUBLIC\_ADM = CNS20 # Public Administration employment  
 ) |>  
 dplyr::select(  
 GEOID,  
 TOTAL\_EMP,  
 AGR\_FOR\_FI,  
 MINING,  
 CONSTRUCTI,  
 MANUFACTUR,  
 TRANSP\_COM,  
 WHOLESALE,  
 RETAIL,  
 FIRE,  
 SERVICES,  
 PUBLIC\_ADM  
 )

Using cached version of file found in:  
C:\Users\pukar\AppData\Local\R\cache\R\lehdr\sc\_wac\_S000\_JT00\_2019.csv.gz

str(lscog\_emp)

tibble [2,450 × 12] (S3: tbl\_df/tbl/data.frame)  
 $ GEOID : chr [1:2450] "450030201001001" "450030201001017" "450030201001037" "450030201001049" ...  
 $ TOTAL\_EMP : num [1:2450] 6 4 12 1 11 1 9 18 1 4 ...  
 $ AGR\_FOR\_FI: num [1:2450] 6 0 0 0 11 0 0 0 0 4 ...  
 $ MINING : num [1:2450] 0 0 0 0 0 0 0 0 0 0 ...  
 $ CONSTRUCTI: num [1:2450] 0 0 12 0 0 0 0 18 1 0 ...  
 $ MANUFACTUR: num [1:2450] 0 0 0 0 0 0 0 0 0 0 ...  
 $ TRANSP\_COM: num [1:2450] 0 4 0 0 0 0 0 0 0 0 ...  
 $ WHOLESALE : num [1:2450] 0 0 0 0 0 0 0 0 0 0 ...  
 $ RETAIL : num [1:2450] 0 0 0 1 0 0 0 0 0 0 ...  
 $ FIRE : num [1:2450] 0 0 0 0 0 0 0 0 0 0 ...  
 $ SERVICES : num [1:2450] 0 0 0 0 0 1 9 0 0 0 ...  
 $ PUBLIC\_ADM: num [1:2450] 0 0 0 0 0 0 0 0 0 0 ...

#### Python

# Download LEHD WAC data at block level  
lscog\_emp = get\_lodes(  
 state=state\_abb,  
 year=2019,  
 version="LODES8",  
 lodes\_type="wac",  
 part="main",  
 segment="S000",  
 job\_type="JT00",  
 agg\_level="block",  
 cache=True,  
 return\_geometry=True  
)

Requesting feature geometry.  
Using FIPS code '45' for input 'sc'

# Filter for specific FIPS codes  
lscog\_emp = lscog\_emp[lscog\_emp['w\_geocode'].str.match(f"^({'|'.join(fips\_codes)})")]  
  
# Create new columns with employment categories  
# Check documentation at: https://lehd.ces.census.gov/data/lodes/LODES8/LODESTechDoc8.0.pdf  
lscog\_emp = lscog\_emp.assign(  
 GEOID=lscog\_emp['w\_geocode'].astype(str),  
 TOTAL\_EMP=lscog\_emp['C000'], # Total Employment  
 AGR\_FOR\_FI=lscog\_emp['CNS01'], # Agricultural, forestry, and fishing employment  
 MINING=lscog\_emp['CNS02'], # Mining employment  
 CONSTRUCTI=lscog\_emp['CNS04'], # Construction employment  
 MANUFACTUR=lscog\_emp['CNS05'], # Manufacturing employment  
 TRANSP\_COM=lscog\_emp['CNS08'] + lscog\_emp['CNS09'], # Transportation, communication employment  
 WHOLESALE=lscog\_emp['CNS06'], # Wholesale employment  
 RETAIL=lscog\_emp['CNS07'], # Retail employment  
 FIRE=lscog\_emp['CNS10'] + lscog\_emp['CNS11'], # Finance / Insurance / Real Estate employment  
 SERVICES=(lscog\_emp['CNS03'] +  
 lscog\_emp['CNS12'] +  
 lscog\_emp['CNS13'] +  
 lscog\_emp['CNS14'] +  
 lscog\_emp['CNS15'] +  
 lscog\_emp['CNS16'] +  
 lscog\_emp['CNS17'] +  
 lscog\_emp['CNS18'] +  
 lscog\_emp['CNS19']), # Service employment  
 PUBLIC\_ADM=lscog\_emp['CNS20'] # Public Administration employment  
)  
  
# Transform CRS  
lscog\_emp = lscog\_emp.to\_crs(project\_crs)  
  
# Select only the desired columns  
lscog\_emp = lscog\_emp[['GEOID', 'TOTAL\_EMP', 'AGR\_FOR\_FI', 'MINING', 'CONSTRUCTI',  
 'MANUFACTUR', 'TRANSP\_COM', 'WHOLESALE', 'RETAIL', 'FIRE',  
 'SERVICES', 'PUBLIC\_ADM']]  
  
# Display structure/info about the dataframe  
lscog\_emp.info()

<class 'pandas.core.frame.DataFrame'>  
Index: 2450 entries, 44 to 36981  
Data columns (total 12 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 GEOID 2450 non-null object  
 1 TOTAL\_EMP 2450 non-null int64   
 2 AGR\_FOR\_FI 2450 non-null int64   
 3 MINING 2450 non-null int64   
 4 CONSTRUCTI 2450 non-null int64   
 5 MANUFACTUR 2450 non-null int64   
 6 TRANSP\_COM 2450 non-null int64   
 7 WHOLESALE 2450 non-null int64   
 8 RETAIL 2450 non-null int64   
 9 FIRE 2450 non-null int64   
 10 SERVICES 2450 non-null int64   
 11 PUBLIC\_ADM 2450 non-null int64   
dtypes: int64(11), object(1)  
memory usage: 248.8+ KB

## 3.4 2020 NCES school and college enrollment data

The National Center for Education Statistics provides comprehensive educational institution data including enrollment and staffing information for transportation planning analysis.

### Public schools

Public school data is retrieved from the NCES ArcGIS REST service for the 2019-2020 academic year. The dataset includes total student enrollment and full-time equivalent teacher counts for each institution within the six-county region. Public schools represent major trip generation sources for both student and employee travel, requiring precise spatial location data for accurate modeling.

#### R

# Public School Location data 2019-2020  
lscog\_pub\_sch\_enroll <- arcgislayers::arc\_read(  
 url = "https://nces.ed.gov/opengis/rest/services/K12\_School\_Locations/EDGE\_ADMINDATA\_PUBLICSCH\_1920/MapServer/0",  
 where = paste0(  
 "LSTATE = '",  
 state\_abb,  
 "' AND NMCNTY IN (",  
 paste0("'", paste0(county\_names, " County"), "'", collapse = ", "),  
 ")"  
 ),  
 alias = "label",  
 crs = project\_crs  
) |>  
 dplyr::select(  
 INSTITUTION\_ID = NCESSCH,  
 NAME = SCH\_NAME,  
 STATE = LSTATE,  
 STUDENT\_COUNT\_PUB = TOTAL,  
 TEACHER\_COUNT\_PUB = FTE  
 )

Registered S3 method overwritten by 'jsonify':  
 method from   
 print.json jsonlite

str(lscog\_pub\_sch\_enroll)

Classes 'sf' and 'data.frame': 98 obs. of 6 variables:  
 $ INSTITUTION\_ID : chr "450108001163" "450075000064" "450075001184" "450075001415" ...  
 ..- attr(\*, "label")= chr "Unique School ID"  
 $ NAME : chr "Barnwell Elementary" "Allendale Fairfax High" "Allendale Elementary" "Allendale-Fairfax Middle" ...  
 ..- attr(\*, "label")= chr "School name"  
 $ STATE : chr "SC" "SC" "SC" "SC" ...  
 ..- attr(\*, "label")= chr "Location state"  
 $ STUDENT\_COUNT\_PUB: num 462 283 245 268 381 188 162 149 353 0 ...  
 ..- attr(\*, "label")= chr "Total students, all grades (includes AE)"  
 $ TEACHER\_COUNT\_PUB: num 32 28.9 18 18 28.5 15 20 10 25 12 ...  
 ..- attr(\*, "label")= chr "Total Teachers"  
 $ geometry :sfc\_POINT of length 98; first list element: 'XY' num 1891531 517379  
 - attr(\*, "sf\_column")= chr "geometry"  
 - attr(\*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA  
 ..- attr(\*, "names")= chr [1:5] "INSTITUTION\_ID" "NAME" "STATE" "STUDENT\_COUNT\_PUB" ...

#### Python

# Create function to read ArcGIS FeatureLayer or Table  
def arc\_read(url, where="1=1", outFields="\*", outSR=4326, \*\*kwargs):  
 """  
 Read an ArcGIS FeatureLayer or Table to a GeoDataFrame.  
  
 Parameters:  
 url (str): The ArcGIS REST service URL ending with /MapServer/0 or /FeatureServer/0  
 where (str): SQL WHERE clause for filtering. Default: "1=1" (all records)  
 outFields (str): Comma-separated field names or "\*" for all fields. Default: "\*"  
 outSR (int): Output spatial reference EPSG code. Default: 4326  
 \*\*kwargs: Additional query parameters passed to the ArcGIS REST API  
  
 Returns:  
 geopandas.GeoDataFrame: Spatial data from the service  
 """  
  
 # Ensure URL ends with /query  
 if not url.endswith('/query'):  
 url = url.rstrip('/') + '/query'  
  
 # Build query parameters  
 params = {  
 'where': where,  
 'outFields': outFields,  
 'returnGeometry': 'true',  
 # 'geometryType': 'esriGeometryPoint',  
 'outSR': outSR,  
 'f': 'geojson'  
 }  
  
 # Add any additional parameters  
 params.update(kwargs)  
  
 # Make request  
 response = requests.get(url, params=params)  
  
 # Read as GeoDataFrame  
 return gpd.read\_file(response.text)

# Public School Enrollment data 2019-2020  
lscog\_pub\_sch\_enroll = arc\_read(  
 url="https://nces.ed.gov/opengis/rest/services/K12\_School\_Locations/EDGE\_ADMINDATA\_PUBLICSCH\_1920/MapServer/0",  
 where=f"LSTATE = '{state\_abb}' AND NMCNTY IN ('{"', '".join([f"{name} County" for name in county\_names])}')",  
 outFields='NCESSCH,SCH\_NAME,LSTATE,TOTAL,FTE'  
)  
  
# Transform CRS  
lscog\_pub\_sch\_enroll = lscog\_pub\_sch\_enroll.to\_crs(project\_crs)  
  
# Select and rename columns  
lscog\_pub\_sch\_enroll = lscog\_pub\_sch\_enroll.rename(columns={  
 'NCESSCH': 'INSTITUTION\_ID',  
 'SCH\_NAME': 'NAME',  
 'LSTATE': 'STATE',  
 'TOTAL': 'STUDENT\_COUNT\_PUB',  
 'FTE': 'TEACHER\_COUNT\_PUB'  
})  
  
lscog\_pub\_sch\_enroll.info()

<class 'geopandas.geodataframe.GeoDataFrame'>  
RangeIndex: 98 entries, 0 to 97  
Data columns (total 6 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 INSTITUTION\_ID 98 non-null object   
 1 NAME 98 non-null object   
 2 STATE 98 non-null object   
 3 STUDENT\_COUNT\_PUB 98 non-null int32   
 4 TEACHER\_COUNT\_PUB 98 non-null float64   
 5 geometry 98 non-null geometry  
dtypes: float64(1), geometry(1), int32(1), object(3)  
memory usage: 4.3+ KB

### Private schools

Private school enrollment data is accessed from the NCES Private School Survey archived dataset. The data is spatially enabled using latitude and longitude coordinates and filtered to include only institutions within the study area TAZ boundaries. Private schools contribute to the regional education trip matrix and must be incorporated alongside public institutions for comprehensive coverage.

#### R

# Private School Enrollment data 2019-2020  
lscog\_pvt\_sch\_enroll <- vroom::vroom(  
 unz(  
 file.path(  
 root,  
 "GIS/data\_external/20250315 NCES/PSS - Private/2019-20/pss1920\_pu\_csv.zip"  
 ),  
 "pss1920\_pu.csv"  
 ),  
 col\_types = vroom::cols\_only(  
 PPIN = vroom::col\_character(),  
 PINST = vroom::col\_character(),  
 PL\_STABB = vroom::col\_character(),  
 PCNTNM = vroom::col\_character(),  
 SIZE = vroom::col\_double(),  
 NUMTEACH = vroom::col\_double(),  
 LATITUDE20 = vroom::col\_double(),  
 LONGITUDE20 = vroom::col\_double()  
 )  
) |>  
 sf::st\_as\_sf(coords = c("LONGITUDE20", "LATITUDE20"), crs = "EPSG:4326") |>  
 sf::st\_transform(project\_crs) |>  
 sf::st\_filter(lscog\_taz, .predicate = st\_intersects) |>  
 dplyr::select(  
 INSTITUTION\_ID = PPIN,  
 NAME = PINST,  
 STATE = PL\_STABB,  
 STUDENT\_COUNT\_PVT = SIZE,  
 TEACHER\_COUNT\_PVT = NUMTEACH  
 )  
  
str(lscog\_pvt\_sch\_enroll)

sf [24 × 6] (S3: sf/tbl\_df/tbl/data.frame)  
 $ INSTITUTION\_ID : chr [1:24] "K9305823" "01264947" "A9106158" "01263568" ...  
 $ NAME : chr [1:24] "AIKENS FBC PRESCHOOL" "ANDREW JACKSON ACADEMY" "BARNWELL CHRISTIAN SCHOOL" "CALHOUN ACADEMY" ...  
 $ STATE : chr [1:24] "SC" "SC" NA "SC" ...  
 $ STUDENT\_COUNT\_PVT: num [1:24] 1 3 2 3 1 1 1 1 1 1 ...  
 $ TEACHER\_COUNT\_PVT: num [1:24] 1.3 13.3 5.8 26.6 3.1 2.8 6 5 2.9 1 ...  
 $ geometry :sfc\_POINT of length 24; first list element: 'XY' num [1:2] 1781276 629441  
 - attr(\*, "sf\_column")= chr "geometry"  
 - attr(\*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA  
 ..- attr(\*, "names")= chr [1:5] "INSTITUTION\_ID" "NAME" "STATE" "STUDENT\_COUNT\_PVT" ...

#### Python

# Private School Enrollment data 2019-2020  
zip\_path = Path(root) / "GIS/data\_external/20250315 NCES/PSS - Private/2019-20/pss1920\_pu\_csv.zip"  
  
with zipfile.ZipFile(zip\_path, 'r') as zip\_ref:  
 with zip\_ref.open('pss1920\_pu.csv') as csv\_file:  
 lscog\_pvt\_sch\_enroll = pd.read\_csv(  
 csv\_file,  
 usecols=['PPIN', 'PINST', 'PL\_STABB', 'PCNTNM', 'SIZE', 'NUMTEACH', 'LATITUDE20', 'LONGITUDE20'],  
 dtype={'PPIN': 'str', 'PINST': 'str', 'PL\_STABB': 'str', 'PCNTNM': 'str', 'SIZE': 'float64', 'NUMTEACH': 'float64'}  
 )  
  
lscog\_pvt\_sch\_enroll = gpd.GeoDataFrame(  
 lscog\_pvt\_sch\_enroll,  
 geometry=gpd.points\_from\_xy(lscog\_pvt\_sch\_enroll['LONGITUDE20'], lscog\_pvt\_sch\_enroll['LATITUDE20']),  
 crs='EPSG:4326'  
).to\_crs(project\_crs)  
  
lscog\_pvt\_sch\_enroll = gpd.sjoin(lscog\_pvt\_sch\_enroll, lscog\_taz, how='inner', predicate='intersects')[  
 ['PPIN', 'PINST', 'PL\_STABB', 'SIZE', 'NUMTEACH', 'geometry']  
].rename(columns={  
 'PPIN': 'INSTITUTION\_ID',  
 'PINST': 'NAME',  
 'PL\_STABB': 'STATE',  
 'SIZE': 'STUDENT\_COUNT\_PVT',  
 'NUMTEACH': 'TEACHER\_COUNT\_PVT'  
})  
  
lscog\_pvt\_sch\_enroll.info()

<class 'geopandas.geodataframe.GeoDataFrame'>  
Index: 24 entries, 17469 to 17733  
Data columns (total 6 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 INSTITUTION\_ID 24 non-null object   
 1 NAME 24 non-null object   
 2 STATE 7 non-null object   
 3 STUDENT\_COUNT\_PVT 24 non-null float64   
 4 TEACHER\_COUNT\_PVT 24 non-null float64   
 5 geometry 24 non-null geometry  
dtypes: float64(2), geometry(1), object(3)  
memory usage: 1.3+ KB

### Post-secondary institutions

Post-secondary institution locations are obtained from the NCES Postsecondary School Locations service, filtered by state and county FIPS codes. These institutions generate significant travel demand through student commuting, employee travel, and visitor trips, making them essential components of the regional transportation network analysis.

#### R

# Post-Secondary Location data 2019-2020  
lscog\_college\_loc <- arcgislayers::arc\_read(  
 url = "https://nces.ed.gov/opengis/rest/services/Postsecondary\_School\_Locations/EDGE\_GEOCODE\_POSTSECONDARYSCH\_1920/MapServer/0",  
 where = paste0(  
 "STATE = '",  
 state\_abb,  
 "' AND CNTY IN (",  
 paste0("'", fips\_codes, "'", collapse = ", "),  
 ")"  
 ),  
 alias = "label",  
 crs = project\_crs  
)  
  
str(lscog\_college\_loc)

Classes 'sf' and 'data.frame': 11 obs. of 25 variables:  
 $ OBJECTID : num 3411 3424 3431 3439 3448 ...  
 ..- attr(\*, "label")= chr "OBJECTID"  
 $ UNITID : chr "217615" "217873" "217989" "218159" ...  
 ..- attr(\*, "label")= chr "School identification number"  
 $ NAME : chr "Aiken Technical College" "Claflin University" "Denmark Technical College" "Kenneth Shuler School of Cosmetology-North Augusta" ...  
 ..- attr(\*, "label")= chr "Name of institution"  
 $ STREET : chr "2276 Jefferson Davis Highway" "400 Magnolia Street" "1126 Solomon Blatt Blvd" "1113 Knox Avenue" ...  
 ..- attr(\*, "label")= chr "Reported street address"  
 $ CITY : chr "Graniteville" "Orangeburg" "Denmark" "North Augusta" ...  
 ..- attr(\*, "label")= chr "Reported city"  
 $ STATE : chr "SC" "SC" "SC" "SC" ...  
 ..- attr(\*, "label")= chr "Reported state"  
 $ ZIP : chr "29829" "29115-4498" "29042" "29841" ...  
 ..- attr(\*, "label")= chr "Reported ZIP code"  
 $ STFIP : chr "45" "45" "45" "45" ...  
 ..- attr(\*, "label")= chr "State FIPS"  
 $ CNTY : chr "45003" "45075" "45009" "45003" ...  
 ..- attr(\*, "label")= chr "County FIPS"  
 $ NMCNTY : chr "Aiken County" "Orangeburg County" "Bamberg County" "Aiken County" ...  
 ..- attr(\*, "label")= chr "County name"  
 $ LOCALE : chr "41" "32" "41" "21" ...  
 ..- attr(\*, "label")= chr "Locale code"  
 $ LAT : num 33.5 33.5 33.3 33.5 33.5 ...  
 ..- attr(\*, "label")= chr "Latitude of school location"  
 $ LON : num -81.8 -80.9 -81.1 -82 -80.8 ...  
 ..- attr(\*, "label")= chr "Longitude of school location"  
 $ CBSA : chr "12260" "36700" "N" "12260" ...  
 ..- attr(\*, "label")= chr "Core Based Statistical Area"  
 $ NMCBSA : chr "Augusta-Richmond County, GA-SC" "Orangeburg, SC" "N" "Augusta-Richmond County, GA-SC" ...  
 ..- attr(\*, "label")= chr "Core Based Statistical Area name"  
 $ CBSATYPE : chr "1" "2" "0" "1" ...  
 ..- attr(\*, "label")= chr "Metropolitan or Micropolitan Statistical Area indicator"  
 $ CSA : chr "N" "192" "N" "N" ...  
 ..- attr(\*, "label")= chr "Combined Statistical Area"  
 $ NMCSA : chr "N" "Columbia-Orangeburg-Newberry, SC" "N" "N" ...  
 ..- attr(\*, "label")= chr "Combined Statistical Area name"  
 $ NECTA : chr "N" "N" "N" "N" ...  
 ..- attr(\*, "label")= chr "New England City and Town Area"  
 $ NMNECTA : chr "N" "N" "N" "N" ...  
 ..- attr(\*, "label")= chr "New England City and Town Area name"  
 $ CD : chr "4502" "4506" "4506" "4502" ...  
 ..- attr(\*, "label")= chr "Congressional District"  
 $ SLDL : chr "45084" "45095" "45090" "45083" ...  
 ..- attr(\*, "label")= chr "State Legislative District - Lower"  
 $ SLDU : chr "45025" "45039" "45040" "45024" ...  
 ..- attr(\*, "label")= chr "State Legislative District - Upper"  
 $ SCHOOLYEAR: chr "2019-2020" "2019-2020" "2019-2020" "2019-2020" ...  
 ..- attr(\*, "label")= chr "School year"  
 $ geometry :sfc\_POINT of length 11; first list element: 'XY' num 1743560 619744  
 - attr(\*, "sf\_column")= chr "geometry"  
 - attr(\*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA ...  
 ..- attr(\*, "names")= chr [1:24] "OBJECTID" "UNITID" "NAME" "STREET" ...

#### Python

# Post-Secondary Location data 2019-2020  
lscog\_college\_loc = arc\_read(  
 url="https://nces.ed.gov/opengis/rest/services/Postsecondary\_School\_Locations/EDGE\_GEOCODE\_POSTSECONDARYSCH\_1920/MapServer/0",  
 where=f"STATE = '{state\_abb}' AND CNTY IN ('{"', '".join([f"{fip}" for fip in fips\_codes])}')",  
 outFields='\*',  
 outSR=project\_crs  
)  
  
lscog\_college\_loc = lscog\_college\_loc.to\_crs(project\_crs)  
  
lscog\_college\_loc.info()

<class 'geopandas.geodataframe.GeoDataFrame'>  
RangeIndex: 11 entries, 0 to 10  
Data columns (total 25 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 OBJECTID 11 non-null int32   
 1 UNITID 11 non-null object   
 2 NAME 11 non-null object   
 3 STREET 11 non-null object   
 4 CITY 11 non-null object   
 5 STATE 11 non-null object   
 6 ZIP 11 non-null object   
 7 STFIP 11 non-null object   
 8 CNTY 11 non-null object   
 9 NMCNTY 11 non-null object   
 10 LOCALE 11 non-null object   
 11 LAT 11 non-null float64   
 12 LON 11 non-null float64   
 13 CBSA 11 non-null object   
 14 NMCBSA 11 non-null object   
 15 CBSATYPE 11 non-null object   
 16 CSA 11 non-null object   
 17 NMCSA 11 non-null object   
 18 NECTA 11 non-null object   
 19 NMNECTA 11 non-null object   
 20 CD 11 non-null object   
 21 SLDL 11 non-null object   
 22 SLDU 11 non-null object   
 23 SCHOOLYEAR 11 non-null object   
 24 geometry 11 non-null geometry  
dtypes: float64(2), geometry(1), int32(1), object(21)  
memory usage: 2.2+ KB

# 4. Clean data

The data cleaning process involves harmonizing multiple Census data sources to create a comprehensive socioeconomic dataset at the census block level. This requires careful interpolation and integration of American Community Survey (ACS) estimates with Decennial Census counts to maintain spatial consistency and statistical accuracy.

## 4.1 Household-weighted interpolation

The interpolation process transfers ACS block group data to individual census blocks using household counts as weights. This method ensures that socioeconomic characteristics are distributed proportionally based on residential density rather than simple geometric overlay. The tidycensus package provides robust interpolation functionality that preserves the extensive nature of count variables while maintaining spatial relationships.

#### R

# Interpolate ACS data to Decennial Census blocks  
lscog\_acs\_cb <- tidycensus::interpolate\_pw(  
 from = lscog\_acs,  
 to = lscog\_dec,  
 to\_id = "GEOID",  
 extensive = TRUE,  
 weights = lscog\_dec,  
 crs = project\_crs,  
 weight\_column = "HH"  
)  
  
str(lscog\_acs\_cb)

sf [13,961 × 6] (S3: sf/tbl\_df/tbl/data.frame)  
 $ GEOID : chr [1:13961] "450179504004051" "450179504003011" "450179502011045" "450179504001020" ...  
 $ geometry :sfc\_MULTIPOLYGON of length 13961; first list element: List of 1  
 ..$ :List of 1  
 .. ..$ : num [1:13, 1:2] 2084301 2084461 2084474 2084523 2084529 ...  
 ..- attr(\*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"  
 $ INC\_TOTAL: num [1:13961] 0 0 20.56 0 5.78 ...  
 $ INC\_14999: num [1:13961] 0 0 4.27 0 1.94 ...  
 $ INC\_49999: num [1:13961] 0 0 7.562 0 0.924 ...  
 $ INC\_50000: num [1:13961] 0 0 8.72 0 2.92 ...  
 - attr(\*, "sf\_column")= chr "geometry"  
 - attr(\*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA  
 ..- attr(\*, "names")= chr [1:5] "GEOID" "INC\_TOTAL" "INC\_14999" "INC\_49999" ...

#### Python

# Spatially interpolate ACS data to decennial census blocks  
lscog\_acs\_cb = gpd.sjoin(lscog\_dec, lscog\_acs, how="left", predicate="intersects")  
  
# Apply population-weighted interpolation  
lscog\_acs\_cb = (  
 lscog\_acs\_cb.assign(weight=lambda df: df["HH"] / df["HH"].sum())  
 .apply(lambda row: row.drop(["geometry", "HH"]) \* row["weight"], axis=1)  
)  
  
# Convert back to a GeoDataFrame with original block geometries  
lscog\_acs\_cb = gpd.GeoDataFrame(lscog\_dec[["GEOID", "geometry"]].join(lscog\_acs\_cb))

The comparative visualization reveals the increased spatial resolution achieved through interpolation. Block-level data provides more granular detail for transportation modeling applications, enabling better representation of local variations in income distribution across the study area.

#### R

# Compare before and after interpolation  
mapview::mapview(lscog\_acs\_cb, zcol = "INC\_49999", color = NA) |  
 mapview::mapview(lscog\_acs, zcol = "INC\_49999", color = NA)

#### Python

# Compare before and after interpolation  
lscog\_acs\_cb.explore(column="INC\_49999", color="blue", legend=True, tiles="CartoDB positron") |\  
 lscog\_acs.explore(column="INC\_49999", color="red", legend=True, tiles="CartoDB positron")

## 4.2 Combine population and households

The integration step merges the interpolated ACS socioeconomic data with the Decennial Census population and household counts. This join operation creates a unified dataset containing both demographic totals and detailed characteristics at the census block level. The left join ensures that all census blocks retain their geographic boundaries while incorporating available socioeconomic attributes.

#### R

## Combine ACS Data to Decennial data  
lscog\_pop\_hh <- lscog\_dec |>  
 dplyr::left\_join(  
 sf::st\_drop\_geometry(lscog\_acs\_cb),  
 by = dplyr::join\_by(GEOID)  
 )  
  
str(lscog\_pop\_hh)

sf [13,961 × 15] (S3: sf/tbl\_df/tbl/data.frame)  
 $ GEOID : chr [1:13961] "450179504004051" "450179504003011" "450179502011045" "450179504001020" ...  
 $ TOTPOP : num [1:13961] 0 0 54 0 13 10 0 6 0 18 ...  
 $ GQPOP : num [1:13961] 0 0 4 0 0 0 0 0 0 0 ...  
 $ HHPOP : num [1:13961] 0 0 50 0 13 10 0 6 0 18 ...  
 $ HH : num [1:13961] 0 0 16 0 4 3 0 4 0 0 ...  
 $ HH\_1 : num [1:13961] 0 0 3 0 0 3 0 0 0 0 ...  
 $ HH\_2 : num [1:13961] 0 0 1 0 3 0 0 1 0 0 ...  
 $ HH\_3 : num [1:13961] 0 0 3 0 0 0 0 2 0 0 ...  
 $ HH\_4 : num [1:13961] 0 0 9 0 1 0 0 1 0 0 ...  
 $ DU : num [1:13961] 0 0 18 0 4 8 0 4 0 1 ...  
 $ geometry :sfc\_MULTIPOLYGON of length 13961; first list element: List of 1  
 ..$ :List of 1  
 .. ..$ : num [1:13, 1:2] 2084301 2084461 2084474 2084523 2084529 ...  
 ..- attr(\*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"  
 $ INC\_TOTAL: num [1:13961] 0 0 20.56 0 5.78 ...  
 $ INC\_14999: num [1:13961] 0 0 4.27 0 1.94 ...  
 $ INC\_49999: num [1:13961] 0 0 7.562 0 0.924 ...  
 $ INC\_50000: num [1:13961] 0 0 8.72 0 2.92 ...  
 - attr(\*, "sf\_column")= chr "geometry"  
 - attr(\*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA ...  
 ..- attr(\*, "names")= chr [1:14] "GEOID" "TOTPOP" "GQPOP" "HHPOP" ...

#### Python

## Combine ACS Data to Decennial data  
lscog\_pop\_hh = lscog\_dec.merge(  
 lscog\_acs\_cb.drop(columns=['geometry']),  
 on='GEOID',  
 how='left'  
)  
  
lscog\_pop\_hh.info()

Income category adjustments reconcile the interpolated ACS estimates with actual household counts from the Decennial Census. The proportional allocation method redistributes income categories based on the ratio of interpolated totals to observed household counts, maintaining consistency between data sources. The three-tier income classification (under $15,000, $15,000-$49,999, and $50,000 and above) provides sufficient granularity for travel demand modeling while ensuring statistical reliability.

#### R

## Combine adjusted HH income level to Decennial census instead of ACS  
lscog\_pop\_hh <- lscog\_pop\_hh |>  
 dplyr::mutate(  
 INC\_49999 = tidyr::replace\_na(round(INC\_49999 / INC\_TOTAL \* HH, 0), 0),  
 INC\_50000 = tidyr::replace\_na(round(INC\_50000 / INC\_TOTAL \* HH, 0), 0),  
 INC\_14999 = HH - (INC\_49999 + INC\_50000)  
 ) |>  
 dplyr::select(-INC\_TOTAL)  
  
str(lscog\_pop\_hh)

sf [13,961 × 14] (S3: sf/tbl\_df/tbl/data.frame)  
 $ GEOID : chr [1:13961] "450179504004051" "450179504003011" "450179502011045" "450179504001020" ...  
 $ TOTPOP : num [1:13961] 0 0 54 0 13 10 0 6 0 18 ...  
 $ GQPOP : num [1:13961] 0 0 4 0 0 0 0 0 0 0 ...  
 $ HHPOP : num [1:13961] 0 0 50 0 13 10 0 6 0 18 ...  
 $ HH : num [1:13961] 0 0 16 0 4 3 0 4 0 0 ...  
 $ HH\_1 : num [1:13961] 0 0 3 0 0 3 0 0 0 0 ...  
 $ HH\_2 : num [1:13961] 0 0 1 0 3 0 0 1 0 0 ...  
 $ HH\_3 : num [1:13961] 0 0 3 0 0 0 0 2 0 0 ...  
 $ HH\_4 : num [1:13961] 0 0 9 0 1 0 0 1 0 0 ...  
 $ DU : num [1:13961] 0 0 18 0 4 8 0 4 0 1 ...  
 $ geometry :sfc\_MULTIPOLYGON of length 13961; first list element: List of 1  
 ..$ :List of 1  
 .. ..$ : num [1:13, 1:2] 2084301 2084461 2084474 2084523 2084529 ...  
 ..- attr(\*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"  
 $ INC\_14999: num [1:13961] 0 0 3 0 1 0 0 1 0 0 ...  
 $ INC\_49999: num [1:13961] 0 0 6 0 1 2 0 2 0 0 ...  
 $ INC\_50000: num [1:13961] 0 0 7 0 2 1 0 1 0 0 ...  
 - attr(\*, "sf\_column")= chr "geometry"  
 - attr(\*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA ...  
 ..- attr(\*, "names")= chr [1:13] "GEOID" "TOTPOP" "GQPOP" "HHPOP" ...

#### Python

## Combine adjusted HH income level to Decennial census instead of ACS  
lscog\_pop\_hh["INC\_49999"] = ((lscog\_pop\_hh["INC\_49999"] / lscog\_pop\_hh["INC\_TOTAL"]) \* lscog\_pop\_hh["HH"]).round().fillna(0)  
lscog\_pop\_hh["INC\_50000"] = ((lscog\_pop\_hh["INC\_50000"] / lscog\_pop\_hh["INC\_TOTAL"]) \* lscog\_pop\_hh["HH"]).round().fillna(0)  
lscog\_pop\_hh["INC\_14999"] = lscog\_pop\_hh["HH"] - (lscog\_pop\_hh["INC\_49999"] + lscog\_pop\_hh["INC\_50000"])  
  
lscog\_pop\_hh = lscog\_pop\_hh.drop(columns="INC\_TOTAL")  
lscog\_pop\_hh.info()

## 4.3 Employment data

The employment integration incorporates LEHD (Longitudinal Employer-Household Dynamics) workplace area characteristics into the combined dataset. This addition provides employment counts by census block, enabling the development of trip attraction models and work-based travel pattern analysis. The merge operation maintains the geographic integrity of census blocks while adding employment variables essential for comprehensive transportation planning.

#### R

# Join LEHD Data to the Decennial data  
lscog\_pop\_hh\_emp <- lscog\_pop\_hh |>  
 dplyr::left\_join(lscog\_emp, by = dplyr::join\_by(GEOID))  
  
str(lscog\_pop\_hh\_emp)

sf [13,961 × 25] (S3: sf/tbl\_df/tbl/data.frame)  
 $ GEOID : chr [1:13961] "450179504004051" "450179504003011" "450179502011045" "450179504001020" ...  
 $ TOTPOP : num [1:13961] 0 0 54 0 13 10 0 6 0 18 ...  
 $ GQPOP : num [1:13961] 0 0 4 0 0 0 0 0 0 0 ...  
 $ HHPOP : num [1:13961] 0 0 50 0 13 10 0 6 0 18 ...  
 $ HH : num [1:13961] 0 0 16 0 4 3 0 4 0 0 ...  
 $ HH\_1 : num [1:13961] 0 0 3 0 0 3 0 0 0 0 ...  
 $ HH\_2 : num [1:13961] 0 0 1 0 3 0 0 1 0 0 ...  
 $ HH\_3 : num [1:13961] 0 0 3 0 0 0 0 2 0 0 ...  
 $ HH\_4 : num [1:13961] 0 0 9 0 1 0 0 1 0 0 ...  
 $ DU : num [1:13961] 0 0 18 0 4 8 0 4 0 1 ...  
 $ geometry :sfc\_MULTIPOLYGON of length 13961; first list element: List of 1  
 ..$ :List of 1  
 .. ..$ : num [1:13, 1:2] 2084301 2084461 2084474 2084523 2084529 ...  
 ..- attr(\*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"  
 $ INC\_14999 : num [1:13961] 0 0 3 0 1 0 0 1 0 0 ...  
 $ INC\_49999 : num [1:13961] 0 0 6 0 1 2 0 2 0 0 ...  
 $ INC\_50000 : num [1:13961] 0 0 7 0 2 1 0 1 0 0 ...  
 $ TOTAL\_EMP : num [1:13961] NA NA NA NA NA NA NA NA NA NA ...  
 $ AGR\_FOR\_FI: num [1:13961] NA NA NA NA NA NA NA NA NA NA ...  
 $ MINING : num [1:13961] NA NA NA NA NA NA NA NA NA NA ...  
 $ CONSTRUCTI: num [1:13961] NA NA NA NA NA NA NA NA NA NA ...  
 $ MANUFACTUR: num [1:13961] NA NA NA NA NA NA NA NA NA NA ...  
 $ TRANSP\_COM: num [1:13961] NA NA NA NA NA NA NA NA NA NA ...  
 $ WHOLESALE : num [1:13961] NA NA NA NA NA NA NA NA NA NA ...  
 $ RETAIL : num [1:13961] NA NA NA NA NA NA NA NA NA NA ...  
 $ FIRE : num [1:13961] NA NA NA NA NA NA NA NA NA NA ...  
 $ SERVICES : num [1:13961] NA NA NA NA NA NA NA NA NA NA ...  
 $ PUBLIC\_ADM: num [1:13961] NA NA NA NA NA NA NA NA NA NA ...  
 - attr(\*, "sf\_column")= chr "geometry"  
 - attr(\*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA ...  
 ..- attr(\*, "names")= chr [1:24] "GEOID" "TOTPOP" "GQPOP" "HHPOP" ...

#### Python

# Join LEHD Data to the Decennial data  
lscog\_pop\_hh\_emp = lscog\_pop\_hh.merge(  
 lscog\_emp,  
 on='GEOID',  
 how='left'  
)  
  
lscog\_pop\_hh\_emp.info()

# 5. Export raw data

The data export process creates standardized datasets for travel demand model development, maintaining both tabular and spatial formats to support various modeling applications. All exports follow consistent file naming conventions and directory structures to facilitate model integration and data management workflows.

## 5.1 TAZ data

The Traffic Analysis Zone (TAZ) boundary export provides the fundamental geographic framework for the regional travel demand model. The blank TAZ file serves as a template for subsequent socioeconomic data allocation, containing only zone identification fields and geometric boundaries without attribute data.

**Export as CSV flat file**

#### R

# Export as CSV  
lscog\_taz |>  
 sf::st\_drop\_geometry() |>  
 readr::write\_csv(  
 file.path(  
 root,  
 "Task 1 TDM Development/Base Year/\_raw/lscog\_taz\_blank.csv"  
 ),  
 append = FALSE  
 )

#### Python

# Export as CSV  
lscog\_taz.to\_csv(  
 Path(root) / "Task 1 TDM Development" / "Base Year" / "\_raw" / "lscog\_taz\_blank.csv",  
 index=False  
)

**Export as Geodatabase layer**

#### R

# Export as GDB  
lscog\_taz |>  
 sf::write\_sf(  
 file.path(  
 root,  
 "Task 1 TDM Development/Base Year/\_gis/LSCOG\_2020Base\_SE.gdb"  
 ),  
 layer = "lscog\_taz\_blank",  
 append = FALSE  
 )

#### Python

# Export as GDB  
lscog\_taz.to\_file(  
 Path(root) / "Task 1 TDM Development" / "Base Year" / "\_gis" / "LSCOG\_2020Base\_SE.gdb",  
 layer='lscog\_taz\_blank',  
 driver='FileGDB'  
)

## 5.2 Census blocks to TAZ conversion

The block-to-TAZ conversion table establishes the critical linkage between fine-scale Census geography and the modeling zone system. This crosswalk file enables the aggregation of block-level socioeconomic data to TAZ boundaries while maintaining traceability to source geographies.

**Export as CSV flat file**

#### R

# Export as CSV  
lscog\_cb |>  
 sf::st\_join(lscog\_taz) |>  
 dplyr::select(GEOID20, ID, TAZ\_ID) |>  
 sf::st\_drop\_geometry() |>  
 readr::write\_csv(  
 file.path(  
 root,  
 "Task 1 TDM Development/Base Year/\_raw/lscog\_block\_taz.csv"  
 ),  
 append = FALSE  
 )

#### Python

# Export as CSV  
lscog\_cb.merge(lscog\_taz[['ID', 'TAZ\_ID']], left\_on='GEOID20', right\_on='ID')[['GEOID20', 'ID', 'TAZ\_ID']].to\_csv(  
 Path(root) / "Task 1 TDM Development" / "Base Year" / "\_raw" / "lscog\_block\_taz.csv",  
 index=False  
)

## 5.3 Decennial census data at census block level

The decennial census block export captures the foundational demographic counts used throughout the modeling process. This dataset provides the most reliable population and household totals at the finest geographic resolution, serving as the base for all subsequent data integration and validation steps.

**Export as CSV flat file**

#### R

# Export as CSV  
lscog\_dec |>  
 sf::st\_drop\_geometry() |>  
 readr::write\_csv(  
 file.path(  
 root,  
 "Task 1 TDM Development/Base Year/\_raw/lscog\_dec\_block.csv"  
 ),  
 append = FALSE  
 )

#### Python

# Export as CSV  
lscog\_dec.to\_csv(  
 Path(root) / "Task 1 TDM Development" / "Base Year" / "\_raw" / "lscog\_dec\_block.csv",  
 index=False  
)

**Export as Geodatabase layer**

#### R

# Export as GDB  
lscog\_dec |>  
 sf::write\_sf(  
 file.path(  
 root,  
 "Task 1 TDM Development/Base Year/\_gis/LSCOG\_2020Base\_SE.gdb"  
 ),  
 layer = "lscog\_dec\_block",  
 append = FALSE  
 )

#### Python

# Export as GDB  
lscog\_dec.to\_file(  
 Path(root) / "Task 1 TDM Development" / "Base Year" / "\_gis" / "LSCOG\_2020Base\_SE.gdb",  
 layer='lscog\_dec\_block',  
 driver='FileGDB'  
)

## 5.4 ACS estimates at census block level

The interpolated ACS data export delivers income distribution estimates at the census block level, providing the socioeconomic stratification necessary for trip generation modeling. This processed dataset represents the final product of the household-weighted interpolation methodology, ready for direct integration into the travel demand model framework.

**Export as CSV flat file**

#### R

# Export as CSV  
lscog\_acs\_cb |>  
 dplyr::select(GEOID, INC\_14999, INC\_49999, INC\_50000) |>  
 sf::st\_drop\_geometry() |>  
 readr::write\_csv(  
 file.path(  
 root,  
 "Task 1 TDM Development/Base Year/\_raw/lscog\_acs\_block.csv"  
 ),  
 append = FALSE  
 )

#### Python

# Export as CSV  
lscog\_acs\_cb[['GEOID', 'INC\_14999', 'INC\_49999', 'INC\_50000']].to\_csv(  
 Path(root) / "Task 1 TDM Development" / "Base Year" / "\_raw" / "lscog\_acs\_block.csv",  
 index=False  
)

**Export as Geodatabase layer**

#### R

# Export as GDB  
lscog\_acs\_cb |>  
 dplyr::select(GEOID, INC\_14999, INC\_49999, INC\_50000) |>  
 sf::write\_sf(  
 file.path(  
 root,  
 "Task 1 TDM Development/Base Year/\_gis/LSCOG\_2020Base\_SE.gdb"  
 ),  
 layer = "lscog\_acs\_block",  
 append = FALSE  
 )

#### Python

# Export as GDB  
lscog\_acs\_cb[['GEOID', 'INC\_14999', 'INC\_49999', 'INC\_50000']].to\_file(  
 Path(root) / "Task 1 TDM Development" / "Base Year" / "\_gis" / "LSCOG\_2020Base\_SE.gdb",  
 layer='lscog\_acs\_block',  
 driver='FileGDB'  
)

## 5.5 LEHD data at census block level

The employment data export provides comprehensive workplace characteristics by industry sector at the census block level. This dataset captures the spatial distribution of employment opportunities across the study region, supporting both trip attraction modeling and economic impact analysis.

**Export as CSV flat file**

#### R

# Export as CSV  
lscog\_pop\_hh\_emp |>  
 dplyr::select(GEOID, TOTAL\_EMP:PUBLIC\_ADM) |>  
 sf::st\_drop\_geometry() |>  
 readr::write\_csv(  
 file.path(  
 root,  
 "Task 1 TDM Development/Base Year/\_raw/lscog\_emp\_block.csv"  
 ),  
 append = FALSE  
 )

#### Python

# Export as CSV  
lscog\_pop\_hh\_emp[['GEOID', 'TOTAL\_EMP', 'AGR\_FOR\_FI', 'MINING', 'CONSTRUCTI',  
 'MANUFACTUR', 'TRANSP\_COM', 'WHOLESALE', 'RETAIL', 'FIRE',  
 'SERVICES', 'PUBLIC\_ADM']].to\_csv(  
 Path(root) / "Task 1 TDM Development" / "Base Year" / "\_raw" / "lscog\_emp\_block.csv",  
 index=False  
)

**Export as Geodatabase layer**

#### R

# Export as GDB  
lscog\_pop\_hh\_emp |>  
 dplyr::select(GEOID, TOTAL\_EMP:PUBLIC\_ADM) |>  
 sf::write\_sf(  
 file.path(  
 root,  
 "Task 1 TDM Development/Base Year/\_gis/LSCOG\_2020Base\_SE.gdb"  
 ),  
 layer = "lscog\_emp\_block",  
 append = FALSE  
 )

#### Python

# Export as GDB  
lscog\_pop\_hh\_emp[['GEOID', 'TOTAL\_EMP', 'AGR\_FOR\_FI', 'MINING', 'CONSTRUCTI',  
 'MANUFACTUR', 'TRANSP\_COM', 'WHOLESALE', 'RETAIL', 'FIRE',  
 'SERVICES', 'PUBLIC\_ADM']].to\_file(  
 Path(root) / "Task 1 TDM Development" / "Base Year" / "\_gis" / "LSCOG\_2020Base\_SE.gdb",  
 layer='lscog\_emp\_block',  
 driver='FileGDB'  
 )

## 5.6 Public schools to TAZ

The public school location export integrates educational facility data with the TAZ system, providing essential inputs for school-related trip modeling. Student and teacher counts by facility support the development of specialized trip generation rates for educational purposes.

**Export as CSV flat file**

#### R

# Export as CSV  
lscog\_pub\_sch\_enroll |>  
 sf::st\_join(lscog\_taz |> dplyr::select(ID, TAZ\_ID)) |>  
 sf::st\_drop\_geometry() |>  
 readr::write\_csv(  
 file.path(  
 root,  
 "Task 1 TDM Development/Base Year/\_raw/lscog\_pubsch\_loc.csv"  
 ),  
 append = FALSE  
 )

#### Python

# Export as CSV  
lscog\_pub\_sch\_enroll.sjoin(  
 lscog\_taz[['ID', 'TAZ\_ID']],  
 how='left'  
)[['INSTITUTION\_ID', 'NAME', 'STATE', 'STUDENT\_COUNT\_PUB', 'TEACHER\_COUNT\_PUB', 'geometry']] \  
 .to\_csv(  
 Path(root) / "Task 1 TDM Development" / "Base Year" / "\_raw" / "lscog\_pubsch\_loc.csv",  
 index=False  
 )

**Export as Geodatabase layer**

#### R

# Export as GDB  
lscog\_pub\_sch\_enroll |>  
 sf::st\_join(lscog\_taz |> dplyr::select(ID, TAZ\_ID)) |>  
 sf::write\_sf(  
 file.path(  
 root,  
 "Task 1 TDM Development/Base Year/\_gis/LSCOG\_2020Base\_SE.gdb"  
 ),  
 layer = "lscog\_pubsch\_loc",  
 append = FALSE  
 )

#### Python

# Export as GDB  
lscog\_pub\_sch\_enroll.sjoin(  
 lscog\_taz[['ID', 'TAZ\_ID']],  
 how='left'  
).to\_file(  
 Path(root) / "Task 1 TDM Development" / "Base Year" / "\_gis" / "LSCOG\_2020Base\_SE.gdb",  
 layer='lscog\_pubsch\_loc',  
 driver='FileGDB'  
)

## 5.7 Private schools to TAZ

The private school dataset complements the public education data by capturing enrollment patterns in private educational institutions. This comprehensive coverage of educational facilities ensures that all school-related travel demand is properly represented in the regional model.

**Export as CSV flat file**

#### R

# Export as CSV  
lscog\_pvt\_sch\_enroll |>  
 sf::st\_join(lscog\_taz |> dplyr::select(ID, TAZ\_ID)) |>  
 sf::st\_drop\_geometry() |>  
 readr::write\_csv(  
 file.path(  
 root,  
 "Task 1 TDM Development/Base Year/\_raw/lscog\_pvtsch\_loc.csv"  
 ),  
 append = FALSE  
 )

#### Python

# Export as CSV  
lscog\_pvt\_sch\_enroll.sjoin(  
 lscog\_taz[['ID', 'TAZ\_ID']],  
 how='left'  
)[['INSTITUTION\_ID', 'NAME', 'STATE', 'STUDENT\_COUNT\_PVT', 'TEACHER\_COUNT\_PVT', 'geometry']] \  
 .to\_csv(  
 Path(root) / "Task 1 TDM Development" / "Base Year" / "\_raw" / "lscog\_pvtsch\_loc.csv",  
 index=False  
 )

**Export as Geodatabase layer**

#### R

# Export as GDB  
lscog\_pvt\_sch\_enroll |>  
 sf::st\_join(lscog\_taz |> dplyr::select(ID, TAZ\_ID)) |>  
 sf::write\_sf(  
 file.path(  
 root,  
 "Task 1 TDM Development/Base Year/\_gis/LSCOG\_2020Base\_SE.gdb"  
 ),  
 layer = "lscog\_pvtsch\_loc",  
 append = FALSE  
 )

#### Python

# Export as GDB  
lscog\_pvt\_sch\_enroll.sjoin(  
 lscog\_taz[['ID', 'TAZ\_ID']],  
 how='left'  
).to\_file(  
 Path(root) / "Task 1 TDM Development" / "Base Year" / "\_gis" / "LSCOG\_2020Base\_SE.gdb",  
 layer='lscog\_pvtsch\_loc',  
 driver='FileGDB'  
)

# 6. Aggregate data to TAZ

The aggregation process transforms fine-scale census block data into TAZ-level inputs suitable for travel demand modeling. This spatial aggregation preserves the total counts while organizing data according to the modeling zone structure required for trip generation and distribution analysis.

## 6.1 Population, households, and employment

The spatial join operation aggregates all demographic, housing, and employment variables from census blocks to their corresponding TAZs using centroid-based assignment. This process ensures that each block’s socioeconomic characteristics are properly allocated to the appropriate modeling zone while maintaining data integrity through comprehensive summation of all relevant variables.

#### R

# Aggregate population, households, and employment to TAZ  
lscog\_taz\_pop <- lscog\_taz |>  
 sf::st\_join(  
 lscog\_pop\_hh\_emp |> sf::st\_centroid(of\_largest\_polygon = TRUE)  
 ) |>  
 dplyr::group\_by(  
 ID,  
 Area,  
 TAZ\_ID,  
 COUNTY,  
 AREA\_TYPE,  
 COUNTYID,  
 .drop = FALSE  
 ) |>  
 dplyr::summarize(  
 .groups = "drop",  
 # Population and Household Size  
 TOTPOP = sum(TOTPOP, na.rm = TRUE),  
 GQPOP = sum(GQPOP, na.rm = TRUE),  
 HHPOP = sum(HHPOP, na.rm = TRUE),  
 HH = sum(HH, na.rm = TRUE),  
 HH\_1 = sum(HH\_1, na.rm = TRUE),  
 HH\_2 = sum(HH\_2, na.rm = TRUE),  
 HH\_3 = sum(HH\_3, na.rm = TRUE),  
 HH\_4 = sum(HH\_4, na.rm = TRUE),  
 DU = sum(DU, na.rm = TRUE),  
 # Household Income  
 INC\_14999 = sum(INC\_14999, na.rm = TRUE),  
 INC\_49999 = sum(INC\_49999, na.rm = TRUE),  
 INC\_50000 = sum(INC\_50000, na.rm = TRUE),  
 # Employment  
 TOTAL\_EMP = sum(TOTAL\_EMP, na.rm = TRUE),  
 AGR\_FOR\_FI = sum(AGR\_FOR\_FI, na.rm = TRUE),  
 MINING = sum(MINING, na.rm = TRUE),  
 CONSTRUCTI = sum(CONSTRUCTI, na.rm = TRUE),  
 MANUFACTUR = sum(MANUFACTUR, na.rm = TRUE),  
 TRANSP\_COM = sum(TRANSP\_COM, na.rm = TRUE),  
 WHOLESALE = sum(WHOLESALE, na.rm = TRUE),  
 RETAIL = sum(RETAIL, na.rm = TRUE),  
 FIRE = sum(FIRE, na.rm = TRUE),  
 SERVICES = sum(SERVICES, na.rm = TRUE),  
 PUBLIC\_ADM = sum(PUBLIC\_ADM, na.rm = TRUE)  
 )

Warning: st\_centroid assumes attributes are constant over geometries

str(lscog\_taz\_pop)

sf [585 × 30] (S3: sf/tbl\_df/tbl/data.frame)  
 $ ID : int [1:585] 3010700 3010701 3010702 3010703 3010704 3010705 3010706 3010707 3010708 3010709 ...  
 $ Area : num [1:585] 0.846 1.098 0.395 0.338 0.381 ...  
 $ TAZ\_ID : int [1:585] 3010700 3010701 3010702 3010703 3010704 3010705 3010706 3010707 3010708 3010709 ...  
 $ COUNTY : chr [1:585] "Aiken SC" "Aiken SC" "Aiken SC" "Aiken SC" ...  
 $ AREA\_TYPE : chr [1:585] "SUBURBAN" "URBAN" "URBAN" "URBAN" ...  
 $ COUNTYID : int [1:585] 45003 45003 45003 45003 45003 45003 45003 45003 45003 45003 ...  
 $ TOTPOP : num [1:585] 66 1299 657 593 462 ...  
 $ GQPOP : num [1:585] 0 0 0 1 0 0 0 0 0 0 ...  
 $ HHPOP : num [1:585] 66 1299 657 592 462 ...  
 $ HH : num [1:585] 20 482 268 237 261 189 62 194 67 71 ...  
 $ HH\_1 : num [1:585] 2 84 46 67 152 64 5 50 19 30 ...  
 $ HH\_2 : num [1:585] 0 211 130 82 65 61 16 92 22 25 ...  
 $ HH\_3 : num [1:585] 5 97 49 21 18 29 17 18 16 14 ...  
 $ HH\_4 : num [1:585] 13 90 43 67 26 35 24 34 10 2 ...  
 $ DU : num [1:585] 22 512 274 257 279 193 71 196 79 74 ...  
 $ INC\_14999 : num [1:585] 0 12 5 7 49 3 1 0 0 0 ...  
 $ INC\_49999 : num [1:585] 4 85 88 86 48 69 23 99 35 36 ...  
 $ INC\_50000 : num [1:585] 16 385 175 144 164 117 38 95 32 35 ...  
 $ TOTAL\_EMP : num [1:585] 21 13 58 13 43 54 101 17 0 0 ...  
 $ AGR\_FOR\_FI: num [1:585] 0 0 0 0 0 0 0 0 0 0 ...  
 $ MINING : num [1:585] 0 0 0 0 0 0 0 0 0 0 ...  
 $ CONSTRUCTI: num [1:585] 0 3 2 1 0 0 0 3 0 0 ...  
 $ MANUFACTUR: num [1:585] 0 0 0 0 0 0 0 0 0 0 ...  
 $ TRANSP\_COM: num [1:585] 0 0 1 0 15 0 0 0 0 0 ...  
 $ WHOLESALE : num [1:585] 0 0 0 0 0 0 0 0 0 0 ...  
 $ RETAIL : num [1:585] 19 0 0 0 15 8 9 0 0 0 ...  
 $ FIRE : num [1:585] 0 5 0 0 13 5 17 2 0 0 ...  
 $ SERVICES : num [1:585] 2 5 55 12 0 41 75 12 0 0 ...  
 $ PUBLIC\_ADM: num [1:585] 0 0 0 0 0 0 0 0 0 0 ...  
 $ SHAPE :sfc\_MULTIPOLYGON of length 585; first list element: List of 1  
 ..$ :List of 1  
 .. ..$ : num [1:100, 1:2] 1699181 1699124 1699097 1699072 1699041 ...  
 ..- attr(\*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"  
 - attr(\*, "sf\_column")= chr "SHAPE"  
 - attr(\*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA ...  
 ..- attr(\*, "names")= chr [1:29] "ID" "Area" "TAZ\_ID" "COUNTY" ...

#### Python

# Aggregate population, households, and employment to TAZ  
lscog\_taz\_pop = lscog\_taz.sjoin(  
 lscog\_pop\_hh\_emp.to\_crs(project\_crs),  
 how='left'  
).groupby(  
 ['ID', 'Area', 'TAZ\_ID', 'COUNTY', 'AREA\_TYPE', 'COUNTYID'],  
 as\_index=False  
).agg({  
 'TOTPOP': 'sum',  
 'GQPOP': 'sum',  
 'HHPOP': 'sum',  
 'HH': 'sum',  
 'HH\_1': 'sum',  
 'HH\_2': 'sum',  
 'HH\_3': 'sum',  
 'HH\_4': 'sum',  
 'DU': 'sum',  
 'INC\_14999': 'sum',  
 'INC\_49999': 'sum',  
 'INC\_50000': 'sum',  
 'TOTAL\_EMP': 'sum',  
 'AGR\_FOR\_FI': 'sum',  
 'MINING': 'sum',  
 'CONSTRUCTI': 'sum',  
 'MANUFACTUR': 'sum',  
 'TRANSP\_COM': 'sum',  
 'WHOLESALE': 'sum',  
 'RETAIL': 'sum',  
 'FIRE': 'sum',  
 'SERVICES': 'sum',  
 'PUBLIC\_ADM': 'sum'  
})  
  
lscog\_taz\_pop.info()

## 6.2 School and college enrollment

The school enrollment combination merges public and private educational institution data into a unified dataset for comprehensive coverage of student populations.

#### R

# Combine school enrollment data  
lscog\_sch\_enroll <- dplyr::bind\_rows(  
 lscog\_pub\_sch\_enroll,  
 lscog\_pvt\_sch\_enroll  
) |>  
 dplyr::mutate(  
 STUDENT\_COUNT = dplyr::coalesce(STUDENT\_COUNT\_PUB, 0) +  
 dplyr::coalesce(STUDENT\_COUNT\_PVT, 0),  
 TEACHER\_COUNT = dplyr::coalesce(TEACHER\_COUNT\_PUB, 0) +  
 dplyr::coalesce(TEACHER\_COUNT\_PVT, 0)  
 )  
  
str(lscog\_sch\_enroll)

Classes 'sf' and 'data.frame': 122 obs. of 10 variables:  
 $ INSTITUTION\_ID : chr "450108001163" "450075000064" "450075001184" "450075001415" ...  
 $ NAME : chr "Barnwell Elementary" "Allendale Fairfax High" "Allendale Elementary" "Allendale-Fairfax Middle" ...  
 $ STATE : chr "SC" "SC" "SC" "SC" ...  
 $ STUDENT\_COUNT\_PUB: num 462 283 245 268 381 188 162 149 353 0 ...  
 $ TEACHER\_COUNT\_PUB: num 32 28.9 18 18 28.5 15 20 10 25 12 ...  
 $ STUDENT\_COUNT\_PVT: num NA NA NA NA NA NA NA NA NA NA ...  
 $ TEACHER\_COUNT\_PVT: num NA NA NA NA NA NA NA NA NA NA ...  
 $ geometry :sfc\_POINT of length 122; first list element: 'XY' num 1891531 517379  
 $ STUDENT\_COUNT : num 462 283 245 268 381 188 162 149 353 0 ...  
 $ TEACHER\_COUNT : num 32 28.9 18 18 28.5 15 20 10 25 12 ...  
 - attr(\*, "sf\_column")= chr "geometry"  
 - attr(\*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA  
 ..- attr(\*, "names")= chr [1:9] "INSTITUTION\_ID" "NAME" "STATE" "STUDENT\_COUNT\_PUB" ...

#### Python

# Combine school enrollment data  
lscog\_sch\_enroll = pd.concat([  
 lscog\_pub\_sch\_enroll.assign(  
 STUDENT\_COUNT=lscog\_pub\_sch\_enroll['STUDENT\_COUNT\_PUB'],  
 TEACHER\_COUNT=lscog\_pub\_sch\_enroll['TEACHER\_COUNT\_PUB']  
 ),  
 lscog\_pvt\_sch\_enroll.assign(  
 STUDENT\_COUNT=lscog\_pvt\_sch\_enroll['STUDENT\_COUNT\_PVT'],  
 TEACHER\_COUNT=lscog\_pvt\_sch\_enroll['TEACHER\_COUNT\_PVT']  
 )  
])  
  
lscog\_sch\_enroll.info()

The subsequent TAZ aggregation counts total student enrollment within each zone, providing essential data for modeling education-related trip patterns and supporting specialized trip generation rates for school-based travel.

#### R

# count the number of school enrollment within each TAZ  
lscog\_taz\_enroll <- lscog\_taz |>  
 sf::st\_join(lscog\_sch\_enroll) |>  
 dplyr::group\_by(  
 ID,  
 Area,  
 TAZ\_ID,  
 COUNTY,  
 AREA\_TYPE,  
 COUNTYID,  
 .drop = FALSE  
 ) |>  
 dplyr::summarize(  
 .groups = "drop",  
 STUDENT\_COUNT = sum(STUDENT\_COUNT, na.rm = TRUE)  
 )  
  
str(lscog\_taz\_enroll)

sf [585 × 8] (S3: sf/tbl\_df/tbl/data.frame)  
 $ ID : int [1:585] 3010700 3010701 3010702 3010703 3010704 3010705 3010706 3010707 3010708 3010709 ...  
 $ Area : num [1:585] 0.846 1.098 0.395 0.338 0.381 ...  
 $ TAZ\_ID : int [1:585] 3010700 3010701 3010702 3010703 3010704 3010705 3010706 3010707 3010708 3010709 ...  
 $ COUNTY : chr [1:585] "Aiken SC" "Aiken SC" "Aiken SC" "Aiken SC" ...  
 $ AREA\_TYPE : chr [1:585] "SUBURBAN" "URBAN" "URBAN" "URBAN" ...  
 $ COUNTYID : int [1:585] 45003 45003 45003 45003 45003 45003 45003 45003 45003 45003 ...  
 $ STUDENT\_COUNT: num [1:585] 0 0 0 0 0 717 0 0 0 0 ...  
 $ SHAPE :sfc\_MULTIPOLYGON of length 585; first list element: List of 1  
 ..$ :List of 1  
 .. ..$ : num [1:100, 1:2] 1699181 1699124 1699097 1699072 1699041 ...  
 ..- attr(\*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"  
 - attr(\*, "sf\_column")= chr "SHAPE"  
 - attr(\*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA  
 ..- attr(\*, "names")= chr [1:7] "ID" "Area" "TAZ\_ID" "COUNTY" ...

#### Python

# count the number of school enrollment within each TAZ  
lscog\_taz\_enroll = lscog\_taz.sjoin(lscog\_sch\_enroll, how='left') \  
 .groupby(['ID', 'Area', 'TAZ\_ID', 'COUNTY', 'AREA\_TYPE', 'COUNTYID'], as\_index=False) \  
 .agg({'STUDENT\_COUNT': 'sum'})

# 7. Combine into a single data

This step integrates all TAZ-level socioeconomic datasets into a comprehensive base year file for travel demand modeling. This consolidated dataset contains all essential variables organized in a standardized format with geographic identifiers, demographic characteristics, employment data, and educational enrollment totals for each traffic analysis zone in the study region.

#### R

# Combine population, household, employments, and school enrollment  
lscog\_se\_base <- lscog\_taz\_pop |>  
 dplyr::left\_join(  
 lscog\_taz\_enroll |> sf::st\_drop\_geometry(),  
 by = dplyr::join\_by(ID, Area, TAZ\_ID, COUNTY, AREA\_TYPE, COUNTYID)  
 ) |>  
 dplyr::select(  
 ID,  
 Area,  
 TAZ\_ID,  
 COUNTY,  
 AREA\_TYPE,  
 COUNTYID,  
 INC\_14999,  
 INC\_49999,  
 INC\_50000,  
 TOTPOP,  
 GQPOP,  
 HHPOP,  
 HH,  
 HH\_1,  
 HH\_2,  
 HH\_3,  
 HH\_4,  
 DU,  
 dplyr::everything()  
 )  
  
str(lscog\_se\_base)

sf [585 × 31] (S3: sf/tbl\_df/tbl/data.frame)  
 $ ID : int [1:585] 3010700 3010701 3010702 3010703 3010704 3010705 3010706 3010707 3010708 3010709 ...  
 $ Area : num [1:585] 0.846 1.098 0.395 0.338 0.381 ...  
 $ TAZ\_ID : int [1:585] 3010700 3010701 3010702 3010703 3010704 3010705 3010706 3010707 3010708 3010709 ...  
 $ COUNTY : chr [1:585] "Aiken SC" "Aiken SC" "Aiken SC" "Aiken SC" ...  
 $ AREA\_TYPE : chr [1:585] "SUBURBAN" "URBAN" "URBAN" "URBAN" ...  
 $ COUNTYID : int [1:585] 45003 45003 45003 45003 45003 45003 45003 45003 45003 45003 ...  
 $ INC\_14999 : num [1:585] 0 12 5 7 49 3 1 0 0 0 ...  
 $ INC\_49999 : num [1:585] 4 85 88 86 48 69 23 99 35 36 ...  
 $ INC\_50000 : num [1:585] 16 385 175 144 164 117 38 95 32 35 ...  
 $ TOTPOP : num [1:585] 66 1299 657 593 462 ...  
 $ GQPOP : num [1:585] 0 0 0 1 0 0 0 0 0 0 ...  
 $ HHPOP : num [1:585] 66 1299 657 592 462 ...  
 $ HH : num [1:585] 20 482 268 237 261 189 62 194 67 71 ...  
 $ HH\_1 : num [1:585] 2 84 46 67 152 64 5 50 19 30 ...  
 $ HH\_2 : num [1:585] 0 211 130 82 65 61 16 92 22 25 ...  
 $ HH\_3 : num [1:585] 5 97 49 21 18 29 17 18 16 14 ...  
 $ HH\_4 : num [1:585] 13 90 43 67 26 35 24 34 10 2 ...  
 $ DU : num [1:585] 22 512 274 257 279 193 71 196 79 74 ...  
 $ TOTAL\_EMP : num [1:585] 21 13 58 13 43 54 101 17 0 0 ...  
 $ AGR\_FOR\_FI : num [1:585] 0 0 0 0 0 0 0 0 0 0 ...  
 $ MINING : num [1:585] 0 0 0 0 0 0 0 0 0 0 ...  
 $ CONSTRUCTI : num [1:585] 0 3 2 1 0 0 0 3 0 0 ...  
 $ MANUFACTUR : num [1:585] 0 0 0 0 0 0 0 0 0 0 ...  
 $ TRANSP\_COM : num [1:585] 0 0 1 0 15 0 0 0 0 0 ...  
 $ WHOLESALE : num [1:585] 0 0 0 0 0 0 0 0 0 0 ...  
 $ RETAIL : num [1:585] 19 0 0 0 15 8 9 0 0 0 ...  
 $ FIRE : num [1:585] 0 5 0 0 13 5 17 2 0 0 ...  
 $ SERVICES : num [1:585] 2 5 55 12 0 41 75 12 0 0 ...  
 $ PUBLIC\_ADM : num [1:585] 0 0 0 0 0 0 0 0 0 0 ...  
 $ SHAPE :sfc\_MULTIPOLYGON of length 585; first list element: List of 1  
 ..$ :List of 1  
 .. ..$ : num [1:100, 1:2] 1699181 1699124 1699097 1699072 1699041 ...  
 ..- attr(\*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"  
 $ STUDENT\_COUNT: num [1:585] 0 0 0 0 0 717 0 0 0 0 ...  
 - attr(\*, "sf\_column")= chr "SHAPE"  
 - attr(\*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA ...  
 ..- attr(\*, "names")= chr [1:30] "ID" "Area" "TAZ\_ID" "COUNTY" ...

#### Python

# Combine population, household, employments, and school enrollment  
lscog\_se\_base = lscog\_taz\_pop.merge(  
 lscog\_taz\_enroll,  
 on=['ID', 'Area', 'TAZ\_ID', 'COUNTY', 'AREA\_TYPE', 'COUNTYID'],  
 how='left'  
)[[  
 'ID', 'Area', 'TAZ\_ID', 'COUNTY', 'AREA\_TYPE', 'COUNTYID',  
 'INC\_14999', 'INC\_49999', 'INC\_50000',  
 'TOTPOP', 'GQPOP', 'HHPOP',  
 'HH', 'HH\_1', 'HH\_2', 'HH\_3', 'HH\_4', 'DU'  
] + list(lscog\_taz\_pop.columns.difference(['ID', 'Area', 'TAZ\_ID',  
 'COUNTY', 'AREA\_TYPE',  
 'COUNTYID']))]  
   
lscog\_se\_base.info()

# 8. Data checks and validation

The validation process ensures data integrity and consistency across all socioeconomic variables through systematic checks of categorical totals. These validation steps identify any discrepancies between aggregate totals and component categories that may have occurred during the interpolation or aggregation processes.

## 8.1 Household size total

This validation confirms that the total household count matches the sum of all household size categories for each TAZ. Any discrepancies indicate potential issues in the household size distribution that require investigation and correction before model implementation.

#### R

# check the sum of household by household size  
lscog\_se\_base |>  
 dplyr::filter(HH != (HH\_1 + HH\_2 + HH\_3 + HH\_4)) |>  
 nrow()

[1] 0

#### Python

# check the sum of household by household size  
lscog\_se\_base[  
 lscog\_se\_base['HH'] != (lscog\_se\_base['HH\_1'] + lscog\_se\_base['HH\_2'] +  
 lscog\_se\_base['HH\_3'] + lscog\_se\_base['HH\_4'])  
].shape[0]

## 8.2 Household income total

The income category validation verifies that household totals equal the sum of all three income brackets across all zones. This check ensures the integrity of the income distribution data following the proportional allocation methodology applied during the ACS interpolation process.

#### R

# check the sum of household by income level  
lscog\_se\_base |>  
 dplyr::filter(HH != (INC\_14999 + INC\_49999 + INC\_50000)) |>  
 nrow()

[1] 0

#### Python

# check the sum of household by income level  
lsccog\_se\_base[  
 lscog\_se\_base['HH'] != (lscog\_se\_base['INC\_14999'] + lscog\_se\_base['INC\_49999'] +  
 lscog\_se\_base['INC\_50000'])  
].shape[0]

## 8.3 Employment categories

The employment validation confirms that total employment equals the sum of all industry sector categories for each TAZ. This comprehensive check validates the LEHD data integration and ensures that no employment is lost or double-counted during the sectoral disaggregation process.RetryClaude can make mistakes. Please double-check responses.

#### R

# check the sum of employment by categories  
lscog\_se\_base |>  
 dplyr::filter(  
 TOTAL\_EMP !=  
 (AGR\_FOR\_FI +  
 MINING +  
 CONSTRUCTI +  
 MANUFACTUR +  
 TRANSP\_COM +  
 WHOLESALE +  
 RETAIL +  
 FIRE +  
 SERVICES +  
 PUBLIC\_ADM)  
 ) |>  
 nrow()

[1] 0

#### Python

# check the sum of employment by categories  
lscog\_se\_base[  
 lscog\_se\_base['TOTAL\_EMP'] != (  
 lscog\_se\_base['AGR\_FOR\_FI'] +  
 lscog\_se\_base['MINING'] +  
 lscog\_se\_base['CONSTRUCTI'] +  
 lscog\_se\_base['MANUFACTUR'] +  
 lscog\_se\_base['TRANSP\_COM'] +  
 lscog\_se\_base['WHOLESALE'] +  
 lscog\_se\_base['RETAIL'] +  
 lscog\_se\_base['FIRE'] +  
 lscog\_se\_base['SERVICES'] +  
 lscog\_se\_base['PUBLIC\_ADM']  
 )  
].shape[0]

# 9. Export the final data

The final export creates the complete TAZ-level socioeconomic dataset in both tabular and spatial formats for direct integration into the travel demand model. This comprehensive dataset serves as the primary input for trip generation, providing all necessary demographic, economic, and educational variables organized by traffic analysis zone for the LSCOG regional modeling system.

**Export as CSV flat file**

#### R

# Export as CSV  
lscog\_se\_base |>  
 sf::st\_drop\_geometry() |>  
 readr::write\_csv(  
 file.path(root, "Task 1 TDM Development/Base Year/\_raw/lscog\_se\_taz.csv"),  
 append = FALSE  
 )

#### Python

# Export as CSV  
lscog\_se\_base |>  
 gpd.GeoDataFrame.to\_csv(  
 Path(root) / "Task 1 TDM Development" / "Base Year" / "\_raw" / "lscog\_se\_taz.csv",  
 index=False  
 )

**Export as Geodatabase layer**

#### R

# Export as GDB  
lscog\_se\_base |>  
 sf::write\_sf(  
 file.path(  
 root,  
 "Task 1 TDM Development/Base Year/\_gis/LSCOG\_2020Base\_SE.gdb"  
 ),  
 layer = "lscog\_se\_base",  
 append = FALSE  
 )

#### Python

# Export as GDB  
lscog\_se\_base |>  
 gpd.GeoDataFrame.to\_file(  
 Path(root) / "Task 1 TDM Development" / "Base Year" / "\_gis" / "LSCOG\_2020Base\_SE.gdb"  
 layer='lscog\_se\_base',  
 driver='FileGDB'  
 )

<!--  
quarto::quarto\_render("SE\_2020\_Download\_SC.Qmd", output\_format = "all") -->