LSCOG 2050 LRTP - Base Year SE Data Development

Pukar Bhandari
Metro Analytics
pbhandari@metroanalytics.com

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.

1.1.-.a R

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

```
Loading required package: pacman
```

```r
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
)
```

1.1.-.b 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

1.2.-.a 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".
```

```r
# set project crs
project_crs <- "EPSG:3361"
```

1.2.-.b Python

```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

1.3.-.a R

```r
# Set your API key into environment
tidycensus::census_api_key("your_api_key_here", install = TRUE)
```

1.3.-.b Python

```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.

1.4.-.a R

```r
# Set your main data folder
root <- "M:/MA_Project/SC_LSCOG LRTP"
```

1.4.-.b Python

```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.

2.1.-.a R

```r
# Define state abbreviation and county names
state_abb <- "SC"
county_names <- c(
  "Aiken",
  "Allendale",
  "Bamberg",
  "Barnwell",
  "Calhoun",
  "Orangeburg"
)
```

2.1.-.b Python

```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.

2.1.-.c R

```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"
```

2.1.-.d 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.

2.2.-.a 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" ...
```

2.2.-.b 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.

2.2.-.c R

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

2.2.-.d 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.

3.1.-.a 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-confidenti
ality-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" ...
```

3.1.-.b 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.

3.2.-.a 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" ...
```

3.2.-.b 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.

3.3.-.a R

```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 ...
```

3.3.-.b Python

```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'
```

```python
# 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.

3.4.-.a 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" ...
```

3.4.-.b Python

```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.

3.4.-.c 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" ...
```

3.4.-.d 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.

3.4.-.e 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" ...
```

3.4.-.f Python

```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.

4.1.-.a R

```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" ...
```

4.1.-.b 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.

4.1.-.c 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)
```

4.1.-.d 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.

4.2.-.a 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" ...
```

4.2.-.b 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.

4.2.-.c 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" ...
```

4.2.-.d 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.

4.3.-.a 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" ...
```

4.3.-.b Python

```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

5.1.-.a R

```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
  )
```

5.1.-.b Python

```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

5.1.-.c R

```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
  )
```

5.1.-.d Python

```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

5.2.-.a R

```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
  )
```

5.2.-.b Python

```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

5.3.-.a R

```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
  )
```

5.3.-.b 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

5.3.-.c 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
  )
```

5.3.-.d 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

5.4.-.a R

```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
  )
```

5.4.-.b Python

```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

5.4.-.c R

```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
  )
```

5.4.-.d Python

```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

5.5.-.a R

```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
  )
```

5.5.-.b Python

```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

5.5.-.c R

```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
  )
```

5.5.-.d 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

5.6.-.a 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
  )
```

5.6.-.b 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

5.6.-.c 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
  )
```

5.6.-.d 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

5.7.-.a 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
  )
```

5.7.-.b 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

5.7.-.c 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
  )
```

5.7.-.d 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.

6.1.-.a R

```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" ...
```

6.1.-.b Python

```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.

6.2.-.a 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" ...
```

6.2.-.b 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.

6.2.-.c 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" ...
```

6.2.-.d 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.

7.0.-.a 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()
  )
```

```r
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" ...
```

7.0.-.b Python

```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.

8.1.-.a R

```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
```

8.1.-.b Python

```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.

8.2.-.a R

```
# check the sum of household by income level
lscog_se_base |>
  dplyr::filter(HH != (INC_14999 + INC_49999 + INC_50000)) |>
  nrow()
```

```
[1] 0
```

8.2.-.b 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.

8.3.-.a 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
```

8.3.-.b 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

9.0.-.a 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
  )
```

9.0.-.b 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

9.0.-.c 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
  )
```

9.0.-.d 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") -->
```