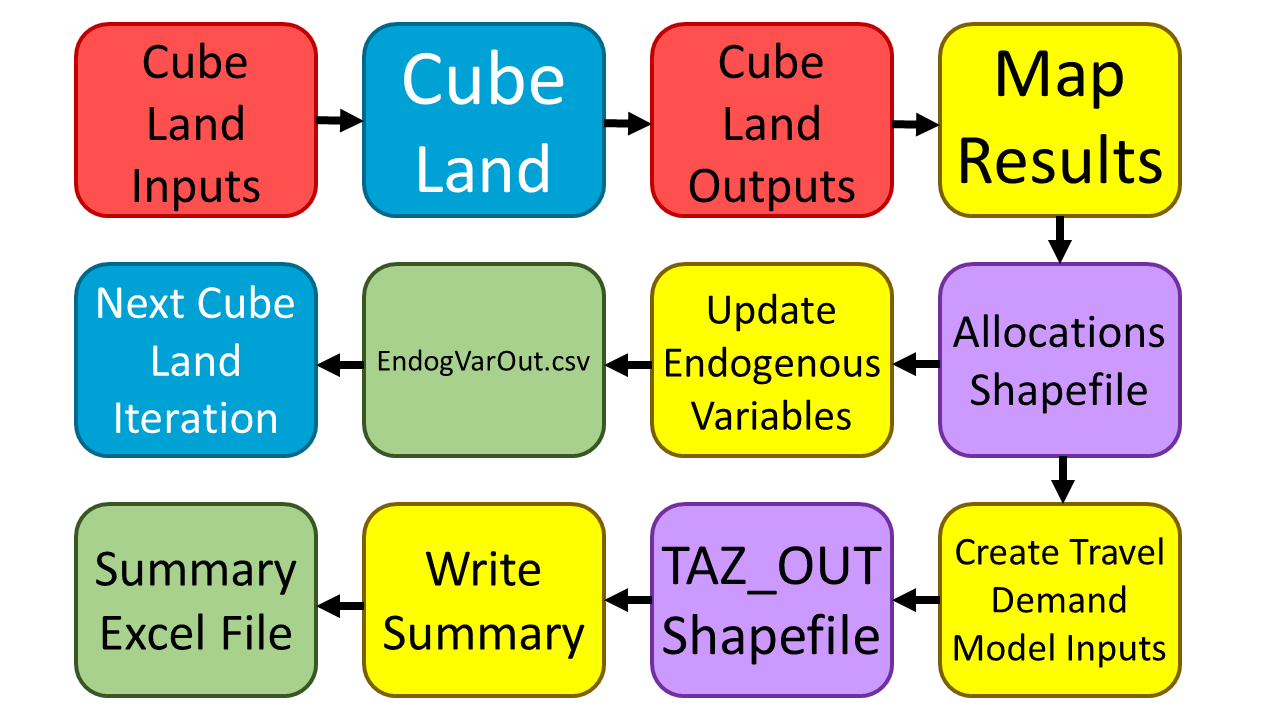
CILUM User’s Guide

# Data Structure

The data flow in CILUM is as follows:



Red: Cube land input/output tables  
Blue: Cube land runs  
Yellow: Python scripts  
Purple: ESRI Shapefiles  
Green: Files viewed in Microsoft Excel

First, Cube Land is run. Then, the script MapResults.py places the output files (supply, locations, and rent) into the file ALLOCATIONS.shp, creating a combined pivot table for the three outputs. This shapefile is used as an input to two different scripts, UpdateEndogVar.py, which updates the endogenous variables in the model, and lu2tdm.py, which creates travel demand model inputs.

When both the endogenous variables and travel demand model inputs are updated, attributes for each zone are calculated with matrix multiplications between subsets of the allocation data and subsets of columns in the agent table. For example, let’s say that 3 zones have the following number of modeled households within 4 agent categories:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Zone** | **Category 1** | **Category 2** | **Category 3** | **Category 4** |
| **1** | 12 | 5 | 3 | 0 |
| **2** | 1 | 18 | 9 | 11 |
| **3** | 0 | 0 | 15 | 20 |

Then, let’s say that agent category 1 has an average household size of 1.5, category 2 has an average size of 2, category 3 has an average size of 2.5, and category 4 has an average size of 3. The modeled population for each zone would be calculated in the following way:

So zone 1 would be modeled to have 32.5 people, zone 2 would have 93 people, and zone 3 would have 97.5 people. Other variables, such as average income, number of jobs, and percentage of low-income households are calculated in a similar way.

The file TAZ\_OUT.dbf, which contains the inputs to the travel demand model, is also the primary input file to the script WriteSummary.py, which writes a summary of the run, aggregating data by county, city/town, and by township.

# User Interface

The CILUM user interface is fairly simple. It has a box for entering a scenario, along with four buttons, “Select Scenario,” “Create Scenario,” “Run Model,” and “Run Post-Processing.”

## Select Scenario

By clicking “select scenario,” one can select a scenario that will be run when the model is run. Pressing this button will place a scenario directory in the “Current Scenario” box. If nothing is in the box, the model won’t run.

## Create Scenario

Clicking on this button takes one through the process of creating a scenario. The first thing that happens upon clicking is a window appearing and asking the user if the new scenario is a fixed-supply scenario. If the user selects “Yes,” then a fixed-supply scenario will be created. If the user selects “No,” then a variable-supply scenario will be created.

After the type of scenario is selected, another window appears asking the user the year that the new scenario takes place in. This is saved in the file FILES\YEAR.txt within the new scenario directory. The year is important as the summary script uses years when comparing annual growth rates for scenarios who’s parent scenarios take place in an earlier year.

After entering the year, a window appears asking the user to provide a description of the scenario. This is saved in the file scenario.txt in the new scenario directory. This file is used during the summary-writing process when determining which superdirectories of the current scenario directory are also scenario directories, and thus can be compared to the current scenario.

The scenario creation window then opens. There are three boxes for text, a checkbox, and three buttons. The first text box indicates the base scenario. All of the new scenario’s files, including the model inputs are copied from this directory. Clicking on the button “Select Base Scenario” will open up a dialog box that will allow the user to select a directory. The second text box indicates the directory in which the new scenario will be created. The “Select Scenario Button” allows the user to select a directory. Additionally, there is a checkbox that indicates if the new scenario being created is a child of another scenario. If this is selected, then both the Base Scenario and New Scenario boxes will be filled when the user selects a base scenario. The Scenario Name box is for the user to type in a name of the new scenario. When the “Create Scenario” button is clicked, the following happens:

1. The directory {New Location}/{Scenario Name} is created.
2. The contents of {Base Scenario} are copied into {New Location}/{Scenario Name}.
3. The files {New Location}/{Scenario Name}/scenario.txt and {New Location}/{Scenario Name}/FILES/YEAR.txt are created.

The user can then edit the model inputs as they wish. The scenario will be ready to run.

## Run Model

Clicking on this button runs the model in the Current Scenario box along with the post-processing.

## Run Post-Processing

Clicking on this button opens the post-processing window. At the top of the window is a box for entering the scenario that the user will run the post-processing on. This can be filled by pressing the “Select Scenario” button at the bottom of the window. Doing so opens a dialog box in which the user can select a scenario directory. There are checkboxes for each of the post-processing scripts. Clicking on the button “Run Post-Processing” then runs all of the selected post-processing scripts for the selected scenario.

# The Scenario Directory

* scenario.txt—This file provides a brief description of the scenario. It is needed to identify scenarios for comparison when writing summaries.
* vs\_template.ctl—This is a control file for running Cube Land.
* FILES
  + agent2naics.csv—Matrix that maps number of jobs by agent category to number of jobs by NAICS code
  + ALLOCATIONS.shp—Shapefile containing the number of units, agents, and rent for each zone
  + TAZ\_OUT.shp—Shapefile with travel demand model inputs
  + WorkersPerHH.csv—Shapefile with the number of workers per household for each agent category
  + YEAR.txt—File indicating the scenario’s year
* MODEL
  + IndyLUM\_VS.app—Cube application file
  + 01CTL00A.CTL—Control data
  + 01LUM00A.TXT—Context file that defines table attributes
  + 01LUM00A.INI—Parameters (such as number of zones, agent types, and real estate types)
  + 01LUM00A.DEC—Definition of endogenous variables
  + 01LND00A.PRN—Print file that shows model progress
  + Inputs
    - MGROW.dbf—Control totals
    - ACCATT.dbf—Accessibility and attractiveness values (This file is not used. Information is in the files ZONES.dbf)
    - SUBSIDIES.dbf—File for setting subsidies and taxes
    - COMPETITION.dbf—Defines which agents can bid on each land use type in each zone
    - RHO.dbf—Prohibitions file, which defines which real estate types can be built in each zone
    - FIXEDSUPPLY.dbf—Fixed supply. Only necessary if running model in fixed supply mode.
    - ZONES.dbf—Zonal attributes
    - REST.dbf—Real estate attributes
    - AGENTS.dbf—Agent attributes
    - BIDF.DBF—Bid function parameters
    - ZNAMES.dbf—Table mapping Cube zones to TransCAD zones
    - RESTDESC.dbf—Names of real estate types
    - MARKETS.dbf—Names of markets
    - AGENTDESC.dbf—Names of agent types
    - BIDADJUST.dbf—Bid adjustment factors
    - RENTADJUST.dbf—Rent adjustment factors
    - ADJ\_COST.dbf—Cost adjustment factors
    - RESTRLOC.dbf—Location restrictions
    - RESTRSUPPLY.dbf—Supply restrictions
    - COSTF.DBF—Cost function parameters
    - RENTF.DBF—Rent function parameters
  + Outputs
    - LOCATIONS.dbf—Allocated agent locations
    - RENT.dbf—Modeled rents
    - ZONES\_OUT.dbf—Endogenous variables
    - BIDS.dbf—Modeled bids
    - SUPPLY.dbf—Modeled supply