APPENDIX F

Python Module and UPlan Implementation

TOPICS Include:

1. Overview
2. Appropriate Use Statement
3. Python Module
4. UPlan Implementation
5. Validation Test Plan and Results
6. Ancillary Products
# Table of Contents

1. Overview 3
   A. Programming Terms Used 4

2. Appropriate Use Statement 6

3. Licensing 7

4. Python Module 9
   A. User Guide 9
   B. Code Documentation 10

5. UPlan Implementation 13
   A. Code Documentation 13
   B. Users Guide 23

6. Validation Test Plan and Results 31
   A. Test Plan 31
   B. Python Module Results 32
   C. UPlan Implementation Results 32

7. Ancillary Products
   A. Place Type Translator 46
1. Overview

Appendix “F” describes and documents three products of the “Improved Data and Tools for Integrated Land Use-Transportation Planning in California” funded by the California Department of Transportation (Caltrans). The broader goals and objectives of this study are described in the final report Overview.

One product described in this Appendix, the “python” module, is a building block component for using the study’s findings. (Note: “python” is a type of GIS programming code – see definitions below.) At its core, the python module is a specific-purpose calculator that utilizes the “Ds Analysis Modules” findings (which are described in Appendix “D” of the final report) and enables their application in GIS tools. The python module is not a standalone product and is not intended to exist without the support of other “helper” tools that prepare the inputs and record the outputs from it. In many ways, these helper functions are the more complex component of the tool kit and must be created and tested with care.

The second product described in this Appendix is the “UPlan Implementation.” (note: “UPlan” is a GIS scenario-planning tool that was developed by the University of California at Davis, which is used in a number of small and rural areas of California. Additional information about UPlan is provided in Section 5.) This process starts with the python module and then builds the “helper” functions that are needed to: prepare a baseline land use dataset, merge a UPlan growth projection with it, prepare all of the inputs to the python module, and record its outputs for further use and visualization.

Finally, an ancillary product is a prototype “placetype translation engine.” Written entirely in python, this translation module provides a structured “least distance” matching algorithm for converting one placetype with defined characteristics into the closest matching member of another placetype system based on matching characteristics. Like the python module, this component requires that “helper” functions be created to feed information into it and accept the outputs from it. But given those limits, it should be broadly usable in a wide range of other toolkits.
A. Programming Terms Used

**Python:** a free, open source, programming language in frequent use. ESRI has adopted python as the default scripting language for much of its business model. Python code can be edited by any text editor, though frequently an integrated development environment is used to speed the development process.

**Integrated Development Environment (IDE):** an application to assist with authoring programming code. These frequently include the automatic completion of commands and build in error and syntax checking. The code provided here was developed using the Pydev extension for Eclipse, a free and open source IDE.

**Module:** a unit of programming code distributed in a single python file. A module can take many forms, but in this case the module contains the classes and the code that enables them to be used. eg. “VMTEngine.py” is a module that contains the “main” classes VMT_2Step and VMT_3Step which make use of classes also included in VMTEngine that handle the binomial logistic and linear regression operations.

**Class:** the programming description of the properties and methods of an object intended to complete a task.

**Function:** a component of a class that defines how some task should be completed.

**Object:** an instance of a class. An object is a working copy of the “blueprint” laid out by the class that describes it. For example a simple class might describe an object that takes two numbers, adds them together and returns the result. An object created (formally “instantiated”) from the class would be handed the two numbers by the program which would also receive the result.

In an OOP programming language all variables are objects. i.e. a string is an object as is a number either integer or floating point.

**Object-oriented programming (OOP):** A method of programming in which the applications are built by combining objects (described by classes) to complete a programming task. The advantages of are that each individual class can be self contained and can be built by linking other classes together. In OOP, the fundamental idea is that each individual class serves as a building block that can be reused, extended, tested, and debugged individually, and that if changes are needed they can be made only to the class, and those fixes will propagate through all instances of its use. There can also be multiple objects created from the same class simultaneously which allows, if the classes are defined to support it, for what is known as multi-threading or multi-processing allowing calculations to happen in parallel on modern computers. (note: The objects provided by ESRI for assembling GIS tools do not handle these multi-processing tasks well.)

**Reference Model:** A working example of an algorithm or programming task with both the input and output values available that can be used as a model for subsequent
development. In this case, a spreadsheet assembled by Fehr & Peers for this project that contained the needed algorithms and an application to data was used as the "reference model" for developing the python module. This also provided the initial accuracy testing for the python module as it replicated the results of the spreadsheet for identical input data.

**List:** an ordered set of objects (frequently an ordered set of text or number objects). You retrieve the desired object from a list by its position. i.e. the 3rd item in the list.

**Dictionary:** a key indexed set of objects. i.e. a set in which you find the item you're looking for by using a name for it rather than an order.

**Comment:** A comment is an inactive line or section of code that does not get executed. It is intended as a comment on what the code is intended to do or as a note to other users. In python, a “#” symbol indicates that the remainder of the programing line should be considered a comment.

**Doc String:** A specialized form of comment. Doc Strings are a unified block of comments, generally, at the beginning of a module, class, or function that describes the purpose and usage of that section of code. In python, a block of comments such as a Doc String is enclosed in a set of three sequential single or double quotes like:

```
""
Comment here.
""
```
2. Appropriate Use Statement

The following statement is appended to the beginning of all of the modules prepared as a result of the “Improved Data and Tools for Integrated Land Use-Transportation Planning in California”. It describes the intended use and discourages uses outside of its intended scope.

This is a land use-transportation scenario comparison tool. It conducts a "sketch" level analysis of land use and transportation relationships using "7 Ds": density (or intensity), diversity (mixture of uses), distance to transit, design (# of street intersections), destination (location within a metropolitan area), demographics, and development scale.

It assesses how a proposed scenario differs from an established or "base case" scenario, comparing them on built environment metrics such as the number and types of jobs and households, densities, and the mix of land uses. The tool also compares scenarios regarding travel behavior, and - depending on the scale of the project - can provide estimates of vehicle miles traveled (VMT).

This tool can be used at a jurisdictional or regional level for comparing different scenarios and evaluating consistency with large-scale plans (e.g., Sustainable Communities Strategies, General Plans, or large specific plans). It can also be used to compare alternatives for large-scale land use-transportation plans and projects.

These tools should not be used as a replacement for calibrated and validated travel demand models or for regulatory compliance, such as regarding air quality requirements, CEQA, or NEPA.
3. Licensing

The following license is being applied to all modules distributed under this section of the project. The intent is that the tools be freely available to all who wish to use them without restrictions that would pose difficulties for including them within either other commercial or open source tools. As a result, the Apache 2.0 license has been selected. The following licensing statement will be included in the header of each module distributed as part of this toolkit:

Copyright 2012 Sacramento Area Council of Governments and University of California

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

   http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Contacts: University of California, Nathaniel Roth, neroth@ucdavis.edu
Sacramento Area Council of Governments, Raef Porter, reporter@sacog.org

The full legal details of the Apache 2 license may be viewed at:

http://www.apache.org/licenses/LICENSE-2.0.html

The products of this project licensed under the Apache 2.0 license are subject to the following restrictions (as provided by the license):

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.


Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only
to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

Redistribution:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License. The Apache 2.0 license further limits the liability attached to the use of these toolkits under section eight of the license:

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
4. Python Module

This section will describe the components that, taken as a whole, form the “python module” products of the “Improved Data and Tools for Integrated Land Use-Transportation Planning in California” project.

The python module is available for download from:
http://downloads.ice.ucdavis.edu/ultrans/statewidetools/Data_handler.zip

Any programmer who wishes to have “commit access” to the materials on this website may contact Nathaniel (Nate) Roth at neroth@ucdavis.edu to discuss their addition as a committing member of the project.

A user’s guide that provides a discussion of the requirements for implementing the code will presented along with an example “helper” module that demonstrates feeding information to the tools and accepting results.

This is followed by a brief section documenting the code used will follow. The majority of the detailed documentation of the code is included in the code and that should be considered the authoritative source of documentation on the code.

A. User Guide

All uses of python module require the construction of a helper function that prepares the data, hands it to the VMT Engine and accepts the responses.

It should be noted that anyone planning to make direct use of the python module must be comfortable working with python prior to trying to do so. This document is not intended to provide sufficient background for a novice programmer to implement this toolkit.

Please see http://downloads.ice.ucdavis.edu/ultrans/statewidetools/Data_handler.zip for an example of a helper function that hands rows from a table into the VMT Engine and then accepts the response. This is the simplest case, one in which the GIS work has already been completed to create a single table that has the total number of housing units, the proportions of housing units in each household type, all of the parameters summarized by their half mile buffers, and any other calculations such as the entropy index already calculated.

The example helper function is contained within a module called Data_handler.py.
Within the helper function there are distinct sections:

a) A class called DataHandler which is instantiated into and object with a list “cats” that has the list of household categories, and “classmat” which is a dictionary that has the household type parameters for each of the households categories listed in cats.
   i) The Handler function: Loops through all of the rows in the input table, and assembles them into the dictionary that gets handed to the VMTEngine
   ii) And accepts the return values and prints them to the screen.

b) A section at the bottom that starts with “if __name__ = '__main__':
   i) This section is used when running a module as a stand alone process. In this case, it contains the information to create the “cats” and “classmat” values that the Handler needs. It also reads the preprepared table into memory and does final preparation.
   ii) It then instantiates the DataHandler and hands to it each of the rows from the data table and prints the results to the screen

c) VMTEngine_2stage.py the module containing the 2 step VMTEngine that the datahandler uses to perform the calculations.

For a more complex example of a data handler, please see the discussion in the UPlan Implementation section. This will describe a method for preparing the GIS data for use in the model, handing it to the VMTEngine and recording the results.

B. Code Documentation

A spreadsheet provided by Fehr & Peers (http://downloads.ice.ucdavis.edu/ultrans/statewidetools/_TxD_CLEAN_R egressionBuilding.zip) served as the reference model for the python module. This spreadsheet provided a working example of the functions in use, their application to data, and the results produced.

The VMT Engine is the core calculation component of the python module. It is a self-contained python module that provides python classes called VMTEngine_2step and VMTEngine_3step. The intended use of this module is that a programmer can import it and then instantiate objects of type VMTEngine_2step or VMTEngine_3step as appropriate to their use (please refer to Appendices A-D for descriptions of the differences between the 2 and 3 step versions of the module). Our expectation is that the 2 step version will be the far more frequently used class. It is also the only version that has been applied and tested/validated as of this writing.

Two VMT Engine modules will be provided. The first, VMTEngine.py includes both the VMT_2Stage and VMT_3Stage classes and the classes that contribute to them (linear regression, binomial logistic, and some other helper classes). This module is intended for use in more complex modeling systems where either the two or three stage formulations of the tools may be called upon.

The second module contains only the 2 stage VMT Engine class and needed helper
classes. This reduced set is not strictly speaking needed, but simplifies the process of using it in a modeling framework that will only use the 2 stage version.

Each of the two and three stage modules has the list of parameters needed and the coefficients that need to be applied included within the class as defaults. While a computer programmer can override these values, we do not anticipate that this will be done with any frequency unless these tools are extended to new areas or updated with revised research.

Upon instantiation of an object of either the two step or three step variety, the programmer must immediately provide that object with an identification of the geographic area the object will be used on. This brings the needed parameter list and coefficients to the forefront for immediate application to data fed into the object. Once the object has been instantiated, the helper functions need only hand it a “dictionary” containing the input values, appropriately named and receive the output.

The intended use case requires that the engine be handed the following sets of information:

2) The household type: which includes information about the households size, number of drivers, available vehicles, presence of children, and income. Both this and point 2 are combined into a parameterized dictionary of the inputs which is read in by the VMT_Engine Classes.

3) Geographic context: all of the variables either summarized by the half-mile buffer around the household's location (i.e. number of jobs, acres of retail, etc), direct measurements from the location (i.e. distance to rail station), or calculated values (i.e. jobs/hh, or an land use mixture measure such as the entropy index)

4) The number of households of the type described in point 1.

The results are then handed back in the form of a list, in which the first element is the total VMT, the second is the binary probability (0 or 1) that a household of the type passed in will make one or more trips on that day, and the last item is the VMT that will be generated by a single household of that type.

The 2 step parameters for small MPOs are presented here. The parameters for small mpo three step, or 2 or 3 step formulations for SACOG or SANDAG implementations should reference the appropriate appendices or the code.

There are several distinct form of parameters. Some parameters are Boolean meaning that they’re true or false these are identified with a # sign. Parameters flagged with a * are used in determining the properties of the household. All of these will be identical for the run on similar households. Unmarked parameters are considered to be continuous. Any reference to a buffered area is an area with a half mile (805m) radius (or 503 acres total area) from the center of the geometry being analyzed.
### Table 1: Small MPO 2 Step Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>Constant: 0.6565 (step 1), 1.2171 (step2)</td>
</tr>
<tr>
<td>HHVEH_CAT_0</td>
<td>Household has 0 vehicles #*</td>
</tr>
<tr>
<td>HHSIZE_CAT_1</td>
<td>Household has 1 member #*</td>
</tr>
<tr>
<td>HHSIZE_CAT_2</td>
<td>Household has 2 members #*</td>
</tr>
<tr>
<td>HH_w0015</td>
<td>HH with children 0-15 years old # *</td>
</tr>
<tr>
<td>HH_w1621</td>
<td>HH with children 16-21 years old # *</td>
</tr>
<tr>
<td>WRKCOUNT</td>
<td>Workers in household</td>
</tr>
<tr>
<td>DRVRCNT</td>
<td>Number of Drivers in household</td>
</tr>
<tr>
<td>Income10K</td>
<td>Average HH income by $10k increments</td>
</tr>
<tr>
<td>DEN_NetRes</td>
<td>Household population/residential acres within buffer</td>
</tr>
<tr>
<td>DEN_DevAcres</td>
<td>Sum of Developed area within half mile</td>
</tr>
<tr>
<td>DIV_JobsHH</td>
<td>Number of jobs/hh within buffer</td>
</tr>
<tr>
<td>DIST_RRStop</td>
<td>Distance to nearest railroad station in miles</td>
</tr>
<tr>
<td>DEN_EmpAcre</td>
<td>Jobs per acre within the 1/2 mile buffer (total employment / 503 acres)</td>
</tr>
<tr>
<td>HHVEHCNT</td>
<td>Average Household Vehicle Count</td>
</tr>
<tr>
<td>DEN_inUrban</td>
<td>Record in urban area based on presence of the household in an Urban Area under US Census Definitions (This should be adjusted for future years) #</td>
</tr>
<tr>
<td>mi_acres</td>
<td>Minor commercial acres within buffer</td>
</tr>
<tr>
<td>DIV_LUEntropy</td>
<td>Land use mix within buffer (entropy index) see forumula below</td>
</tr>
<tr>
<td>DSG_Street</td>
<td>Number of Road miles within the half mile buffer</td>
</tr>
</tbody>
</table>

The land use entropy index is calculated as follows:

Employment Entropy = -$E/(\ln(N))$

Where:

$E=(b1/e)\ln(b1/e) + (b2/e)\ln(b2/e) + (b3/e)\ln(b3/e) + (b4/e)\ln(b4/e) + (b5/e)\ln(b5/e)$

$e = \text{total active employment for five active employment categories present in buffer}$

$b1 = \text{office employment}$
5. UPlan Implementation

UPlan is a simple urban growth model that predicts where new growth will occur using a simple set of user defined rules to allocate projected urban growth to available space. UPlan has been used at one point in all of the small MPOs, and many smaller counties in California, and has some users across the rest of the United States, and several international users in Asia and Africa.

A common use case for UPlan is the examination of a set of scenarios for future urban growth and the evaluation of the likely consequences of that growth on infrastructure and the environment. In the past, UPlan results have been converted into inputs for traditional travel demand models for the analysis of VMT and infrastructure.

The products of the “Improved Data and Tools for Integrated Land Use-Transportation Planning in California” project open another alternative that will allow the relatively rapid analysis of VMT consequences in a scenario planning environment.

These tools are provided in a toolbox intended for use with the UPlan 2.6 family in ArcGIS10 (the most frequently used version of UPlan). This ArcGIS toolbox contains the UPlan_VMTTools.tbx, and the needed python modules to run the full suite of tools.

A. User Guide

This section will provide basic instructions on the use of the UPlan_VMTTools Toolbox. This toolbox is intended for use only in UPlan 2.6x versions being run in ArcGIS10. This is by far the most frequent user case, though all efforts will be made to keep the toolkit current as UPlan and ArcGIS are updated.
The VMTTools toolbox will look much like Figure 1 though there are many ways to access the toolboxes. It will have a toolbox file (UPlan_VMTTools.tbx) which contains linkages to the python modules. There will also be a scripts folder that contains all of the python files used. The script folder and the toolbox (.tbx) must be kept in the same folder. The toolbox (.tbx) will be looking for a scripts folder immediately adjacent to itself in the directory structure that contains all of the python files.

Before attempting to use these tools, make sure that you have updated the numpy and scipy installations for the version of python installed with ArcGIS. This is a simple process if you have a simple installation of ArcGIS and only a single python installation on your computer. If you have multiple versions of python installed, you will need to make sure that numpy and scipy are installing for the ArcGIS installation of python.

**Base Data Preparation:**
To begin the base data preparation step, collect your existing land use, analysis boundary, roads, roads intersection, and railroad stop data in ArcMap (Figure 2). Ensure that all of the data is in the same projection as is used by your UPlan model.

Prior to beginning to run the base data preparation, please review the configuration settings embedded in VMTUPlan_Basedata.py for needed updates to reflect your base dataset.

Adjustments must be made to the __init__(self) section of the module using either a text editor or a python IDE.

```python
self.lufield = "lu f"
sel.reslutypes = ['RM', 'RS', 'M'] # All land use types that are considered residential
self.sfdlutypes = ['RS'] # All land use types that are considered SFD
self.mfdlutypes = ['RM', 'M'] # All land use types that are considered MFD
```
self.emlutypes = ['I', 'O', 'R', 'M', 'ID', 'AP', 'RR', 'SC', 'ML', 'P', 'U', 'AP', 'MA', 'MI']  # All land use types that are considered employment
self.retlutypes = ['R', 'RR', 'MI', 'MA']  # All land use types that are considered retail
self.lretlutypes = ['R', 'MI']  # All land use types that are considered local serving retail
self.serlutypes = ['S']  # All land use types that are considered service
self.oflutypes = ['O']  # All land use types that are considered office
self.indlutypes = ['ID', 'U', 'AP']  # All land use types that are considered industrial
self.publutypes = ['I', 'ML', 'P']  # All land use types that are considered public (non-school)
self.edulutypes = ['SC']  # All land use types that are considered educational
self.poptypes = ['pop']  # All fields with totals that should be considered population
self.sfdtypes = ['sfd']  # All fields with totals that should be considered SFD housing units
self.mfdtypes = ['mfd']  # All fields with totals that should be considered MFD housing units
self.rettyes = ['ret']  # All fields with totals that should be considered retail employees
self.sertypes = ['ser']  # All fields with totals that should be considered service employees
self.offtypes = ['office']  # All fields with totals that should be considered office employees
self.indotypes = ['ind']  # All fields with totals that should be considered industrial employees
self.pubtypes = ['inst']  # All fields with totals that should be considered public (non-school) employees
self.edutypes = ['school']  # All fields with totals that should be considered education employees

self.gridsize = 10  # grid size in acres

self.lufield = "lu_f" should be updated so that "lu_f" is replace with the name of the field containing the land use code.
self.reslutypes through self.edulutypes should be updated so that the contents of the "[R', 'S']" (a python list constructor) contain the list of all land use codes that are considered SFD, MFD, Retail, Local Serving Retail and so on.

self.poptypes, self.sfdtypes… And following entries, need to be edited to include the list of all fields containing the total number of population, housing units, and employees of the respective indicated types.

If desired self.gridsize = 10 # grid size in acres may be edited to replace the numeric value with an alternate grid size. This will change the spatial resolution of the analysis. The default 10 acre size appears to be a reasonable compromise in computational speed and spatial resolution. There is an exponential relationship between processing time, storage space, and cell size. If you cut the cell size in half, you multiply the time and storage requirements by four.
Figure 2: Initial data collection
On opening the Base Data Preparation tool from the toolbox you will be presented with a dialog box asking for the following items: (All feature classes should match the projection of the UPlan system).

1. A working director to place the results in. A file geodatabase called baselu.gdb will be created in this directory.

2. A polygon feature class that identifies the boundary of the analysis area

3. A polygon feature class with the existing land use. This dataset must have the following
   a. A field with the land use name
   b. A set of fields with the total number of population, sfd housing units, mfd housing units, retail employees, service employees, office employees, public (non-school) employees, education employees, industrial employees.

4. A line feature class with the road network for the area

5. A point feature class with the road intersections for the area
6. A point feature class with the railroad station locations

Click OK to begin the run. The run will probably take several hours to complete.

**VMTUPlan Calibration:**

Double clicking on the VMT Calibration Tool in the UPlan_VMTTools toolbox will open the dialog for running the calibration section of the tool (Figure 4). This will allow you to run the VMT tools on the existing land use to provide a base line and the ability to compare the results from scenarios to the present, and the present condition output to other sources of data.

![VMT Calibration tool dialog](image)

**Figure 4: VMT Calibration tool dialog**

You will be asked for the following inputs:

- **Base Data Source:** The file geodatabase created by a successful Base Data step
- **Household Details:** A polygon feature class with fields that conform to the naming conventions below. Each of these fields will contain the proportion of all households in that polygon that fall into the designated household type.

   - `BaseModelHV_3p_NoKids` # base = 3+ person household with a vehicle and no kids
   - `BaseModel_HV_3p_w0015` # base + child 0-15
   - `BaseModel_HV_3p_w11621` # base + child 16-21
   - `HHVEH_CAT_0_NoKids` # base + no vehicle
   - `HHSIZE_CAT_1_NoKids` # base + hhsize = 1
   - `HHSIZE_CAT_2_NoKids` # base + hhsize = 2
   - `HHVEH_CAT_0_w0015` # base + no vehicle and child 0-15
   - `HHSIZE_CAT_1_w0015` # base + hhsize = 1 and child 0-15
   - `HHSIZE_CAT_2_w0015` # base + hhsize = 2 and child 0-15
Urban Areas: A polygon feature class that contains the urban areas. In the base year the 2010 US Census urbanized area definition was used as an example. The fields in the feature class do not matter. This is used only as a presence/absence test.

When you click OK, the process will run, and will take a minimum of 3 hours to complete.

On completion there will be three important tables in vmt.gdb

Vmt_by_uid: this contains multiple rows for each grid cell. One row will exist for each household type present in that grid cell. This table cannot be joined directly to the grid feature class also found in the vmt.gdb. However, it can be connected to the grid using the “relate” function. In this way you can select the grids you are interested in through the relationship established see the VMT generation by each household of each type in that grid cell.

To establish a relate, add both the grid and the table to a map document. Right click on the grid feature class, and select Joins and Relates→Relate…. Then select the “uid” field from the grid feature class (step 1), the name of the table to relate to the grid (step 2), and then uid in step 3. The default name in 4 can be used.

To use the relate, open the attribute table of the grid, select the grid cells of interest, and then click on the table options icon at the top left of the attribute table, choose relates, and then the relate that you are interested in. The vmt_by_uid table will open and the rows connected to that grid cell will be selected.

Vmt_by_uid2: This table can be joined as normal with the join established between the uid in the grid feature class and the uid in vmt_by_uid2. This table contains the total VMT generated by all households in the grid cell, the number of households and the average VMT per household for the grid cell. After the join has been established, you can symbolize the results as you would any other data layer.

Combined: This table can also be joined to the grid feature class. It contains the grid level summaries of all variables fed into the VMTEngine for that grid cell and is useful for reviewing the geographic context variables that apply.

Cfg_VMTUPlan:
Before running either the Scenario or Calibration tools please review \texttt{cfg\_VMTUPlan.py} to ensure that all UPlan land uses are included and have appropriate proportions specified.

\textbf{Lulist} (a python list) must be updated so that it contains all of the land use names (short version) in use by the UPlan configuration (aka the land uses in the UPlan Variant).

Then \textbf{luconfig} (a python dictionary) must have an entry added for each UPlan land use that conforms to the following structure:

```python
# Template for UPlan land use conversion
luc = {}  # Establish an empty dictionary for the uplan land use
luc["sfd\_ac"] = 0.0  # the number of acres of SFD for each acre of the uplan land use
luc["mfd\_ac"] = 0.0  # the number of acres of MFD for each acre of the uplan land use
luc["ret\_ac"] = 0.2  # the number of acres of retail for each acre of the uplan land use
luc["lret\_ac"] = 0.05  # the number of acres of local serving retail for each acre of the uplan land use
luc["ser\_ac"] = 0.1  # the number of acres of service for each acre of the uplan land use
luc["off\_ac"] = 0.3  # the number of acres of office for each acre of the uplan land use
luc["pub\_ac"] = 0.05  # the number of acres of public/institutional (non-school) for each acre of the uplan land use
luc["edu\_ac"] = 0.05  # the number of acres of educational space for each acre of the uplan land use
luc["ind\_ac"] = 0.0  # the number of acres of industrial for each acre of the uplan land use
luc["sfd\_u"] = 0.0  # the number of SFD units for each residential unit of the uplan land use
luc["mfd\_u"] = 1.0  # the number of MFD units for each residential unit of the uplan land use
luc["ret\_u"] = 0.2  # the number of retail employees for each employee of the uplan land use
luc["lret\_u"] = 0.05  # the number of local serving retail employees for each employee of the uplan land use
luc["ser\_u"] = 0.25  # the number of service employees for each employee of the uplan land use
luc["off\_u"] = 0.45  # the number of office employees for each employee of the uplan land use
luc["pub\_u"] = 0.05  # the number of public (non-school) employees for each employee of the uplan land use
luc["edu\_u"] = 0.05  # the number of education employees for each employee of the uplan land use
luc["ind\_u"] = 0.0  # the number of industrial employees for each employee of the uplan land use
luc["du\_or"] = 0  # NOT IMPLEMENTED allows the user to override the default UPlan densities
luc["emp\_or"] = 0  # NOT IMPLEMENTED allows the user to override the default UPlan densities
luc["road\_ints"] = 0.4  # the number road intersections per acre
luc["road\_mi"] = 0.4  # the number of road miles per acre
luconfig["lutype"] = luc  # the addition of the configuration dictionary to the main lookup dictionary
```

\textbf{VMTUPlan Scenario}: 

Improved Data and Tools for Integrated Land Use-Transportation Planning in California

Appendix F – GIS Tools: Python Module and UPlan Implementation
Double clicking on the VMT Scenario Tool in the UPlan_VMTTools toolbox will open the dialog for running the Scenario section of the tool (Figure 5). This will allow you to run the VMT tools on the combination of the existing land use and the UPlan scenario.

![VMT Scenario Tool dialog](image)

**Figure 5: VMT Scenarios tool dialog**

You will be asked for the following inputs:

**Final Allocation Raster:** This is the raster called “finalalloc” for the UPlan run of interest.

**Base Data Source:** The file geodatabase created by a successful Base Data step

**Household Details:** A polygon feature class with fields that conform to the naming conventions below. Each of these fields will contain the proportion of all households in that polygon that fall into the designated household type.

- `BaseModelHV_3p_NoKids` # base = 3+ person household with a vehicle and no kids
- `BaseModel_HV_3p_w0015` # base + child 0-15
- `BaseModel_HV_3p_w11621` # base + child 16-21
- `HHVEH_CAT_0_NoKids` # base + no vehicle
- `HHSIZE_CAT_1_NoKids` # base + hhsizesize = 1
- `HHSIZE_CAT_2_NoKids` # base + hhsizesize = 2
- `HHVEH_CAT_0_w0015` # base + no vehicle and child 0-15
- `HHSIZE_CAT_1_w0015` # base + hhsizesize = 1 and child 0-15
- `HHSIZE_CAT_2_w0015` # base + hhsizesize = 2 and child 0-15
- `HHVEH_CAT_0_w1621` # base + no vehicle and child 16-21
- `HHSIZE_CAT_1_w1621` # base + hhsizesize = 1 and child 16-21
Urban Areas: A polygon feature class that contains the urban areas. In the base year the 2010 US Census urbanized area definition was used as an example. The fields in the feature class do not matter. This is used only as a presence/absence test.

When you click OK, the process will run, and will take a minimum of 3 hours to complete.

On completion there will be three important tables in vmt.gdb

Vmt_by_uid: this contains multiple rows for each grid cell. One row will exist for each household type present in that grid cell. This table cannot be joined directly to the grid feature class also found in the vmt.gdb. However, it can be connected to the grid using the “relate” function. In this way you can select the grids you are interested in through the relationship established see the VMT generation by each household of each type in that grid cell.

To establish a relate, add both the grid and the table to a map document. Right click on the grid feature class, and select Joins and Relates→Relate…. Then select the “uid” field from the grid feature class (step 1), the name of the table to relate to the grid (step 2), and then uid in step 3. The default name in 4 can be used.

To use the relate, open the attribute table of the grid, select the grid cells of interest, and then click on the table options icon at the top left of the attribute table, choose relates, and then the relate that you are interested in. The vmt_by_uid table will open and the rows connected to that grid cell will be selected.

Vmt_by_uid2: This table can be joined as normal with the join established between the uid in the grid feature class and the uid in vmt_by_uid2. This table contains the total VMT generated by all households in the grid cell, the number of households and the average VMT per household for the grid cell. After the join has been established, you can symbolize the results as you would any other data layer.

Combined: This table can also be joined to the grid feature class. It contains the grid level summaries of all variables fed into the VMTEngine for that grid cell and is useful for reviewing the geographic context variables that apply.
B. Code Documentation

Like the python module all of the underlying code is programmed in python, though in this case almost all of the GIS functions are handled through calls to the “arcpy” module provided by ESRI as part of the standard ArcGIS 10.0 installation. Through arcpy, a programmer can make calls to almost all of the ESRI GIS tools and data manipulation methods in addition to those built into python and its extensions.

**Numpy and Scipy**

To use these tools with ArcGIS you **must upgrade** the Numpy and Scipy installations that are installed with the default ArcGIS installation.

Numpy and Scipy are open source extensions to python that support higher level numeric processing and some spatial analysis. In particular the newer versions of Scipy include the KDTree indexing algorithm that allows for dramatically improved calculations on the nearest neighbors between spatial features. In this case, the KDTree functions are used to build the lists of neighbors within the half mile buffer used for the toolkit. Through the use of KDTree, the processing time on these buffering operations is reduced from times measured in hours, to times measured in a few minutes or seconds depending on the geographic area being calculated.

The current versions of these can be downloaded from:

Numpy: [https://sourceforge.net/projects/numpy/files/](https://sourceforge.net/projects/numpy/files/)

Scipy: [https://sourceforge.net/projects/scipy/files/](https://sourceforge.net/projects/scipy/files/)

Make sure that you download the installation files for python 2.6 and 32 bit windows. Eg. numpy-1.6.2-win32-superpack-python2.6.exe

**General Notes**

Like the python module, a great deal of the documentation on what is going on within the code is contained within the code itself. The full details will not be repeated here, because doing so would effectively require the inclusion of all code to provide suitable context.

Instead, the general programming flow of each of the three main scripts will be described.

**VMTUPlan_Basedata.py**

This module contains the class BaseDataPrep which handles the preparation of all base data needed to build the “existing” conditions dataset used by the VMT calculator both as the calibration dataset and the base to which UPlan’s projected land use change is added.

This class is instantiated by the concluding block of code:

```python
if __name__ == "__main__":
    print "Starting Process"
    bd = BaseDataPrep()
```

---

Improved Data and Tools for Integrated Land Use-Transportation Planning in California

Appendix F – GIS Tools: Python Module and UPlan Implementation
This code block instantiates an object of type BaseDataPrep called “bd” assigns the parameters passed in by Arcgis to the needed properties of the object and then calls the MainProc() function to run the entire process

The following parameters are required. All spatial layers must be in the same projection as is used by UPlan.

**Wdir:** the path for the working directory that you want to use. There are no formal requirements of this except that it be a folder that you have permission to create files in. However, it is recommended that you keep it close to the scenarios that you will be testing. The basedata.gdb geodatabase created by the BaseDataPrep process will be created in this folder, as will vmt.gdb for the calibration run.

**Ext:** A polygon feature class (shapefile or other feature class) that describes the boundary of the analysis. This will frequently be the full county.

**Eluds:** A polygon feature class containing the existing land use data. This has been modeled around the parcel level data prepared for this project and described in Appendix C(http://downloads.ice.ucdavis.edu/ultrans/stadwidetools/Appendix_C_Data_Collection.pdf... If the format changes from the default described in Section... Adjustments must be made to the __init__(self) section of the module using either a text editor or a python IDE.

```python
bd.bdddebug = True
bd.wdir = arcpy.GetParameterAsText(0)  # folder
bd.ext = arcpy.GetParameterAsText(1)  # boundary feature class
bd.eluds = arcpy.GetParameterAsText(2)  # existing land use dataset
bd.roads = arcpy.GetParameterAsText(3)  # roads dataset
bd.road_ints = arcpy.GetParameterAsText(4)  # road intersections with 3 or more legs
bd.rrstation = arcpy.GetParameterAsText(5)  # point locations of railroad stations

bd.MainProc()

print "Process Finished"
```
self.offlutypes = ['O'] # All land use types that are considered office
self.indlutypes = ['ID', 'U', 'AP'] # All land use types that are considered industrial
self.publutypes = ['I', 'ML', 'P'] # All land use types that are considered public (non-school)
self.edulutypes = ['SC'] # All land use types that are considered educational
self.poatypes = ['pop'] # All fields with totals that should be considered population
self.sfdtypes = ['sfd'] # All fields with totals that should be considered SFD housing units
self.mfdtypes = ['mfd'] # All fields with totals that should be considered MFD housing units
self.rettypes = ['ret'] # All fields with totals that should be considered retail employees
self.sertypes = ['ser'] # All fields with totals that should be considered service employees
self.offtypes = ['office'] # All fields with totals that should be considered office employees
self.indtypes = ['ind'] # All fields with totals that should be considered industrial employees
self.pubtypes = ['inst'] # All fields with totals that should be considered public (non-school) employees
self.edutypes = ['school'] # All fields with totals that should be considered education employees

self.gridsize = 10 # grid size in acres

These lines will need to be updated as follows.

self.lufield = "lu_f" should be updated so that “lu_f" is replace with the name of the field containing the land use code.
self.reslutypes through self.edulutypes should be updated so that the contents of the “[‘R’,‘S’]” (a python list constructor) contain the list of all land use codes that are considered SFD, MFD, Retail, Local Serving Retail and so on.

self.poatypes, self.sfdtypes... And following entries, need to be edited to include the list of all fields containing the total number of population, housing units, and employees of the respective indicated types.

If desired self.gridsize = 10 # grid size in acres may be edited to replace the numeric value with an alternate grid size. This will change the spatial resolution of the analysis. The default 10 acre size appears to be a reasonable compromise in computational speed and spatial resolution. There is an exponential relationship between processing time, storage space, and cell size. If you cut the cell size in half, you multiply the time and storage requirements by four.

Roads: A line feature class with all roads to be included for the road measurements.
Road_ints: A point feature class with all road intersections with 3 or more legs
Rrstation: A point feature class with point locations for all rail road stations.
The internal processing steps are as follows:
1. A file geodatabase called baselu.gdb is created to hold the modules products
2. The base grid is created to cover the entire extent with grid cells of the specified size (default = 10 acres)
3. A table containing the list of all grids with centers within a half mile of each cell’s center is created (using Scipy)
4. The input land use feature class is prepared for intersection with the grid
5. The land use is intersected with the grid. The number of road miles and road intersections in each grid cell is calculated
6. The distance from each grid cell to the closes rail road station is calculated
7. The totals of land use (acres by type and housing units and employment by type) are calculated for each grid cell.

**Cfg_VMTUPlan.py**

This is a configuration file used to define the conversion of the UPlan land use categories into those needed by the VMTTools. The configurations provided here specify the mixture of SFD and MFD housing, and employment by type as understood by the VMTTools in each of the UPlan land uses. It also asserts a new density of roads and road intersections for each UPlan land use that will be used in the toolkit.

This file will need to be edited to support the UPlan configuration used.

Specifically:

**Lulist** (a python list) must be updated so that it contains all of the land use names (short version) in use by the UPlan configuration (aka the land uses in the UPlan Variant).

Then **luconfig** (a python dictionary) must have an entry added for each UPlan land use that conforms to the following structure:

```python
# Template for UPlan land use conversion
luc = {}  # Establish an empty dictionary for the uplan land use
luc["sfd_ac"] = 0.0  # the number of acres of SFD for each acre of the uplan land use
luc["mfd_ac"] = 0.0  # the number of acres of MFD for each acre of the uplan land use
luc["ret_ac"] = 0.2  # the number of acres of retail for each acre of the uplan land use
luc["lret_ac"] = 0.05  # the number of acres of local serving retail for each acre of the uplan land use
luc["ser_ac"] = 0.1  # the number of acres of service for each acre of the uplan land use
luc["off_ac"] = 0.3  # the number of acres of office for each acre of the uplan land use
luc["pub_ac"] = 0.05  # the number of acres of public/institutional (non-school) for each acre of the uplan land use
luc["edu_ac"] = 0.05  # the number of acres of educational space for each acre of the uplan land use
luc["ind_ac"] = 0.0  # the number of acres of industrial for each acre of the uplan land use
luc["sfd_u"] = 0.0  # the number of SFD units for each residential unit of the uplan land use
```
luc[“mfd_u”] = 1.0 # the number of MFD units for each residential unit of the uplan land use
luc[“ret_u”] = 0.2 # the number of retail employees for each employee of the uplan land use
luc[“lret_u”] = 0.05 # the number of local serving retail employees for each employee of the uplan land use
luc[“ser_u”] = 0.25 # the number of service employees for each employee of the uplan land use
luc[“off_u”] = 0.45 # the number of office employees for each employee of the uplan land use
luc[“pub_u”] = 0.05 # the number of public (non-school) employees for each employee of the uplan land use
luc[“edu_u”] = 0.05 # the number of education employees for each employee of the uplan land use
luc[“ind_u”] = 0.0 # the number of industrial employees for each employee of the uplan land use
luc[“du_or”] = 0 # NOT IMPLEMENTED allows the user to override the default UPlan densities
luc[“emp_or”] = 0 # NOT IMPLEMENTED allows the user to override the default UPlan densities
luc[“road_ints”] = 0.4 # the number road intersections per acre
luc[“road_mi”] = 0.4 # the number of road miles per acre
luconfig[“lutype”] = luc # the addition of the configuration dictionary to the main lookup dictionary

The conclusion of the module contains some configuration information for the model overall, these include a list of household types used by the VMTEngine and a list of parameters that define each of those household types. These setting should not be edited except following a discussion with Nathaniel Roth (neroth@ucdavis.edu)

VMTUPlan_Calibration.py

The calibration module applies the VMTEngine to the base dataset generated by VMTUPlan_Basedata.py. This is intended to present a current conditions comparison to other sources. This enables some level of validation of the results against external real-world values.

There is a single class in the module called VMTUPlan. This is the equivalent of the Handler in the example Data_handler.py with the addition that it prepares all of the buffered values in GIS for handoff to the VMTEngine.

The object of type VMTUPlan is instantiated by the following code block at the bottom of the module:

```python
if __name__ == "__main__":

    """
    Arcgis hooks in here and passes in the needed parameters before executing the full model
    """

    basefgdb = arcpy.GetParameterAsText(0)
    hhdetails = arcpy.GetParameterAsText(1)
```
urbarea = arcpy.GetParameterAsText(2)
vmtup = VMTUPlan(basefgdb)
vmtup.hhdetails = hhdetails
vmtup.urbarea = urbarea
vmtup.MainProc()

The following parameters are passed into the module by ArcGIS:

**Basefgdb:** the path to the baselu.gdb prepared in by the BaseData step.

**Hhdetails:** A polygon feature class that contains the proportions of each of the household types. The fields for each of the proportions need to be consistent with the following list:

- **BaseModelHV_3p_NoKids** # base = 3+ person household with a vehicle and no kids
- **BaseModel_HV_3p_w0015** # base + child 0-15
- **BaseModel_HV_3p_w11621** # base + child 16-21
- **HHVEH_CAT_0_NoKids** # base + no vehicle
- **HHSIZE_CAT_1_NoKids** # base + hhsize = 1
- **HHSIZE_CAT_2_NoKids** # base + hhsize = 2
- **HHVEH_CAT_0_w0015** # base + no vehicle and child 0-15
- **HHSIZE_CAT_1_w0015** # base + hhsize = 1 and child 0-15
- **HHSIZE_CAT_2_w0015** # base + hhsize = 2 and child 0-15
- **HHVEH_CAT_0_w1621** # base + no vehicle and child 16-21
- **HHSIZE_CAT_1_w1621** # base + hhsize = 1 and child 16-21
- **HHSIZE_CAT_2_w1621** # base + hhsize = 2 and child 16-21

**Urbarea:** A polygon feature class that contains a boundary of what is considered urban. For current conditions, the US Census definition of urban area has been used.

The processing steps used by this module are as follows:

1. Creation of a local file geodatabase (vmt.gdb) in the same folder with the baselu.gdb
2. Copy data from baselu.gdb to vmt.gdb for processing and to preserve provenance of the data
3. Prepare the household details dataset by intersecting it with the centroids of each grid cell to assign proportions of housing units in that grid cell to each household type.
4. Summarize the variable by half mile buffers, using the “Near Table” from baselu.gdb
5. Combine all needed components from the household details, summarized land use and railroad distance for each grid cell.
6. Loop through all household combinations for each grid cell and record the results
7. Aggregate results to from all households to create a household grid cell total VMT and average VMT per household

**VMTUPlan Scenario.py**

The Scenario module applies the VMTEngine to a dataset created by merging the the base dataset generated by VMTUPlan_Basedata.py with outputs from a UPlan run.

There is a single class in the module called VMTUPlan. This is the equivalent of the Handler in the example Data_handler.py with the addition that it prepares all of the buffered values in GIS for handoff to the VMTEngine.

The object of type VMTUPlan is instantiated by the following code block at the bottom of the module:

```python
if __name__ == "__main__":
    
    # Arcgis hooks in here and passes in the needed parameters before executing the full model
    
    far = arcpy.GetParameterAsText(0)
    basefgdb = arcpy.GetParameterAsText(1)
    hhdetails = arcpy.GetParameterAsText(2)
    urbarea = arcpy.GetParameterAsText(3)
    vmtup = VMTUPlan(far, basefgdb)
    vmtup.hhdetails = hhdetails
    vmtup.urbarea = urbarea
    vmtup.MainProc()
```

The following parameters are passed into the module by ArcGIS:

**Far:** the final allocation raster for the UPlan run to be tested. This must be from a full UPlan setup including the uplan.mdb for the run and the relative path to the initialization.mdb database in the “ini” folder for the UPlan installation. These databases are used in the conversion of the final allocation raster into the scenario tested for the VMT

**Basefgdb:** the path to the baselu.gdb prepared in by the BaseData step.

**Hhdetails:** A polygon feature class that contains the proportions of each of the household types. The fields for each of the proportions need to be consistent with the following list:

- *BaseModelHV_3p_NoKids* # base = 3+ person household with a vehicle and no kids
- *BaseModel_HV_3p_w0015* # base + child 0-15
- *BaseModel_HV_3p_w11621* # base + child 16-21
- *HHVEH_CAT_0_NoKids* # base + no vehicle
- *HHSIZE_CAT_1_NoKids* # base + hhsize = 1
- *HHSIZE_CAT_2_NoKids* # base + hhsize = 2
- *HHVEH_CAT_0_w0015* # base + no vehicle and child 0-15
- *HHSIZE_CAT_1_w0015* # base + hhsize = 1 and child 0-15
Urbarea: A polygon feature class that contains a boundary of what is considered urban. For current conditions, the US Census definition of urban area has been used.

The processing steps used by this module are as follows:

1. Creation of a local file geodatabase (vmt.gdb) in the same folder with the baselu.gdb
2. Copy data from baselu.gdb to vmt.gdb for processing and to preserve provenance of the data
3. Covert the final allocation raster to a vector dataset with density figures for each land use. This utilizes the FAConverter.py module and inputs pulled from the run’s uplan.mdb and the UPlan setup’s initialization.mdb databases.
4. Intersect the final allocation vector dataset with the grid
5. Convert the densities from the final allocation raster back to acres and to housing or employee counts.
6. Aggregate the converted UPlan results to grid level totals
7. Prepare the household details dataset by intersecting it with the centroids of each grid cell to assign proportions of housing units in that grid cell to each household type.
8. Assemble the Base land use and UPlan outputs to a single dataset. This is done as a simple addition of the UPlan land use over and above the existing, both with respect to acrages and housing unit and employment totals.
9. Summarize the variable by half mile buffers, using the “Near Table” from baselu.gdb
10. Combine all needed components from the household details, summarized land use and railroad distance for each grid cell.
11. Loop through all household combinations for each grid cell and record the results
12. Aggregate results to from all households to create a household grid cell total VMT and average VMT per household
6. Validation Testing and Results

This section outlines the performance measures and comparisons of model results to externally available sources of data. A true validation of a model’s outputs requires a data set that is either independent from or built from randomly selected data withheld from the original model estimation. This provides one of the limiting factors on a formal validation of these tools, particularly in the case of a predictive model where there is no real-world data to compare the results against.

This leaves us with a limited set of options. We can compare the results to those of an alternate model. We can compare the results to expected behavior (e.g., elasticities derived from the literature.) None of these are wholly satisfactory when one of the reasons for developing the tool kit is to answer concerns with validity of alternate models and methods.

We can, however compare a current conditions dataset to real world sources if quantified datasets exist. In this case, there is no clearly superior real-world dataset to which the numbers may be compared.

A. Testing Plan

The testing plan is as follows. Our demonstration area is Tulare County and all figures represent values for Tulare County or a subarea of it.

Python Module

The python modules will be compared against their reference model (which is the spreadsheet version of the Ds analysis modules provided by Fehr & Peers) to ensure that the same inputs produce effectively identical results. We mention “effectively identical” meaning that in a programming system different methods for storing the same numbers, specifically floating point decimal numbers, have a small but potentially noticeable set of rounding errors inherent in the calculations. In this case, we will consider results identical if the difference in outputs are less than one one hundredth of a percent (0.0001) of the expected value provided by our reference model. If the python module operating on an identical input dataset to the spreadsheet produces results that satisfy those requirements, both at the aggregate (full county) and in a selection of distinct subareas, the computational fidelity of the python module will be assumed.

UPlan Implementation

The results of the base case UPlan process (results from the VMTUPlan_Calibration.py) will be compared to the results of the reference model, an expansion of the 2009 National
Household Travel Survey, the regional travel demand model, and the HPMS values. These comparisons will be at the regional/county level with the exception of comparisons to the regional travel demand model which can be accomplished at the TAZ level.

The land use projections will be examined in comparison to expected values based on published elasticities. Regional comparisons of these elasticities are inappropriate given the scale at which the tool performs so these comparisons will be conducted at a TAZ level.

**B. Python Module**

The Reference Model spreadsheet produces a total of 6,476,062 household-generated VMT per day. The Python Module running against the identical input dataset and with the coefficients rounded identically produces 6,476,063 VMT per day. This is a difference of 0.0000003 (0.00003%) which meets our threshold.

Testing a selection of seven TAZs, chosen to represent a range from central urban to very rural areas, we see the following comparisons:

**Table 2: Python Module compared to the reference model.** (Numbers have been rounded.)

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Reference Model (VMT)</th>
<th>Python Module (VMT)</th>
<th>Difference</th>
<th>Percent Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>5.2</td>
<td>5.2</td>
<td>0.0000024</td>
<td>0.000237</td>
</tr>
<tr>
<td>813</td>
<td>369.9</td>
<td>369.9</td>
<td>-0.0000172</td>
<td>-0.001725</td>
</tr>
<tr>
<td>1060</td>
<td>3930.5</td>
<td>3930.6</td>
<td>0.0000120</td>
<td>0.001197</td>
</tr>
<tr>
<td>1146</td>
<td>1413.3</td>
<td>1413.3</td>
<td>0.0000011</td>
<td>0.000114</td>
</tr>
<tr>
<td>1293</td>
<td>15624.1</td>
<td>15623.8</td>
<td>-0.0000160</td>
<td>-0.001601</td>
</tr>
<tr>
<td>1350</td>
<td>6575.3</td>
<td>6575.3</td>
<td>0.0000001</td>
<td>0.000014</td>
</tr>
<tr>
<td>1875</td>
<td>7203.2</td>
<td>7203.2</td>
<td>0.0000001</td>
<td>0.000007</td>
</tr>
</tbody>
</table>

**C. UPlan Implementation**

**Current conditions comparisons:**

There are no good direct comparisons between the household VMT generated in the toolkit and sources of data for formal validations. Instead we will present the values that are available for comparison and a brief discussion of why the differences may be expected.
First, in the aggregate comparison of the current conditions “Calibration” run of the existing land use through the same functions that will operate for the combined base conditions and UPlan projection run.

Reference Model Results: 6,476,062 household VMT

VMT Tools Result: 7,117,036 household VMT

Difference: 9.9%

Given the difference in geographic scales being analyzed, we feel that this is a remarkably similar result. The input metrics used within the reference model were compiled based on land use summaries of the half-mile radius surrounding the centroid of each transportation analysis zone (TAZ) and assumed that each TAZ had a homogeneous land use and land cover mixture. The VMT Tools worked off of a grid of 10-acre cells, each of which was populated with land use information from a parcel level analysis. The half-mile radius around each cell then contains approximately 50 others of these cells. The fact that we can aggregate the results of some 311 thousand cells worth of calculations and get a result that is within 10% of that provided by a TAZ level analysis is interesting on several levels. First, it demonstrates that the assumption of homogeneity within a TAZ holds up well under examination. Second, it calls into question the need for parcel or fine geographic scale analyses in scenario tools, particularly in cases where a rapid response with a relatively accurate result is more important than a high precision answer.

Other County Level Comparisons:

Table 3: Comparison of aggregate totals

<table>
<thead>
<tr>
<th></th>
<th>Millions of VMT</th>
<th>Percent Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current Conditions (VMT Tools)</td>
<td>7.117</td>
<td>0.0</td>
</tr>
<tr>
<td>Reference Model</td>
<td>6.476</td>
<td>-9.0</td>
</tr>
<tr>
<td>HPMS</td>
<td>9.965</td>
<td>40</td>
</tr>
<tr>
<td>2009 NHTS</td>
<td>4.724</td>
<td>-33.6</td>
</tr>
<tr>
<td>San Joaquin Valley Interregional Travel Model (SJVITM)</td>
<td>9.57</td>
<td>34.5</td>
</tr>
<tr>
<td>Tulare MIP Model</td>
<td>10.537</td>
<td>48.1</td>
</tr>
<tr>
<td>Tulare MIP Model (removing NHB trips)</td>
<td>5.907</td>
<td>-17.0</td>
</tr>
</tbody>
</table>

Many of the comparisons presented here are false comparisons. They are not directly equivalent. For example, the HPMS values contain through trips, commercial vehicles miles, and non-resident commuter trips as sources of VMT that are not accounted for in
the VMT Tools figures. Tulare County, being on Highway 99 and in an area with extensive inter-county commuting, becomes a “poster child” for these issues. The same issues exist in the San Joaquin Valley Interregional Travel Model (SJVTM). These figures are derived from vehicle loadings on network links and include all modes and all purposes.

An expansion of the 2009 NHTS to create a daily VMT estimate is the lowest daily VMT total by a significant margin. Taken in isolation, this would be very concerning given that this study’s parameters and coefficients were estimated from the 2009 NHTS. However, the 2009 NHTS has demonstrated that it does not agree well with other applicable data sources, such as the HPMS. Joe Castiglione (RSG, Inc.) has noted that several travel demand model formulations (e.g., “activity based” and “4 step” models) have experienced difficulty with calibrating to both the NHTS and observed counts. In addition, Tulare County had a total sample size in the 2009 NHTS of only 173 households, which places a great deal of weight on each household sampled, whereby small discrepancies or biases in the sampling could result in substantial effects on the estimated VMT.

We also tested a suite of very simple UPlan scenarios using these tools. The “current conditions” test (VMTUPlan_Calibration outputs) will be used as a frame of reference. All of the tested scenarios assume the same total growth in households and employment and an end year of 2050. The three alternate scenarios are: A “base case” scenario, where growth continues much as it has in the recent past; a “smart growth” scenario with only modest infill but with modest density increases; and an aggressive “infill” and densification scenario.

Comparisons to most four step travel models on a TAZ level are of limited use because the VMT produced by households is not explicitly tracked throughout the day. This makes linking VMT to the home TAZ for the trip makers possible only through an estimation process.

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Current Conditions</th>
<th>VMT/HH (HB only)</th>
<th>VMT/HH (All)</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>41.4</td>
<td>34.5</td>
<td>39.1</td>
</tr>
<tr>
<td>813</td>
<td>71.3</td>
<td>149</td>
<td>216.9</td>
</tr>
<tr>
<td>1060</td>
<td>17.0</td>
<td>16.5</td>
<td>24.3</td>
</tr>
<tr>
<td>1146</td>
<td>76.9</td>
<td>24.3</td>
<td>28.1</td>
</tr>
<tr>
<td>1293</td>
<td>74.5</td>
<td>17.6</td>
<td>21.2</td>
</tr>
<tr>
<td>1350</td>
<td>18.9</td>
<td>16.8</td>
<td>24.7</td>
</tr>
<tr>
<td>1875</td>
<td>73.0</td>
<td>21.2</td>
<td>33.4</td>
</tr>
</tbody>
</table>

Table 4: Comparison of UPlan Current Conditions Calculation to Local Travel Model. VMT/HH
Figure 6: VMT per Per Household by Scenario
## Table 5: Aggregate Scenarios Comparison

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Vehicle Miles Traveled</th>
<th>Household Count</th>
<th>Average daily VMT per Household</th>
<th>Percent difference from current conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>7,117,036</td>
<td>125,997</td>
<td>56.5</td>
<td>0</td>
</tr>
<tr>
<td>Base case</td>
<td>16,738,000</td>
<td>313,622</td>
<td>53.4</td>
<td>-5.5</td>
</tr>
<tr>
<td>Smart Growth</td>
<td>16,583,883</td>
<td>313,892</td>
<td>52.8</td>
<td>-6.5</td>
</tr>
<tr>
<td>Intensive Infill</td>
<td>12,776,831</td>
<td>313,829</td>
<td>40.7</td>
<td>-27.9</td>
</tr>
</tbody>
</table>

Several interesting features become apparent in the aggregate comparison. The most striking is that the intensive infill scenario produces significantly fewer VMT per household than the current conditions. This is caused largely because some of the core urban areas in this scenario have sufficiently high net residential densities, large enough developed acreages, and a sufficient number of jobs per household that the binary logistic equation to significantly reduce average daily rates of VMT per household. Effectively, the households in core urban areas are not making any privately owned vehicle trip on an average day; their needs are met through transit or non-motorized travel. Each future scenario has small areas that do this, but this is more prevalent in the intensive infill scenario. All of the future scenarios predict an urban mixture that benefits from reduced VMT per household. The “smart” growth scenario only benefits slightly more than the baseline growth, while the intensive infill scenario obtains roughly twice the benefit.

## Table 6: Sample TAZ Reference

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Description of Current Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>Agricultural, immediately adjacent to a small town.</td>
</tr>
<tr>
<td>813</td>
<td>Industrial with large undeveloped parcels. On the edges of a medium sized city.</td>
</tr>
<tr>
<td>1060</td>
<td>A largely residential area, with some undeveloped space and modest commercial, adjacent to the downtown of a large city for the county.</td>
</tr>
<tr>
<td>1146</td>
<td>Agricultural with limited agricultural industrial development (dairy) and scattered rural residential. Approximately equidistant from two growing cities.</td>
</tr>
<tr>
<td>1293</td>
<td>Residential development near the center of the region’s largest city. Relatively low density, with a golf course and some multi-family housing. Adjacent areas have significant office space and modest mixed retail/commercial.</td>
</tr>
</tbody>
</table>
Table 6 provides a brief description of the character of the example TAZs.

Table 7: VMT per HH Comparison by TAZ across scenarios

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Current VMT/HH</th>
<th>Baseline Scenario</th>
<th>% Change</th>
<th>Smart Scenario</th>
<th>% Change</th>
<th>Infill Scenario</th>
<th>% Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>41.4</td>
<td>35.6</td>
<td>-13.9</td>
<td>35.4</td>
<td>-14.5</td>
<td>28.0</td>
<td>-32.3</td>
</tr>
<tr>
<td>813</td>
<td>71.3</td>
<td>48.6</td>
<td>-31.9</td>
<td>50.0</td>
<td>-29.9</td>
<td>31.2</td>
<td>-56.2</td>
</tr>
<tr>
<td>1060</td>
<td>17.0</td>
<td>17.0</td>
<td>0.0</td>
<td>17.0</td>
<td>0.0</td>
<td>14.8</td>
<td>-12.9</td>
</tr>
<tr>
<td>1146</td>
<td>76.9</td>
<td>74.3</td>
<td>-3.4</td>
<td>76.0</td>
<td>-1.2</td>
<td>76.9</td>
<td>0.0</td>
</tr>
<tr>
<td>1293</td>
<td>74.5</td>
<td>74.5</td>
<td>0.0</td>
<td>74.5</td>
<td>0.0</td>
<td>24.5</td>
<td>-67.0</td>
</tr>
<tr>
<td>1350</td>
<td>18.9</td>
<td>19.0</td>
<td>0.4</td>
<td>19.4</td>
<td>2.5</td>
<td>3.2</td>
<td>-83.1</td>
</tr>
<tr>
<td>1875</td>
<td>73.0</td>
<td>54.2</td>
<td>-25.7</td>
<td>65.6</td>
<td>-10.2</td>
<td>72.1</td>
<td>-1.3</td>
</tr>
</tbody>
</table>

Within these scenario tests, access to rail transit and demographic mixtures were kept constant within each analysis unit. Considering demographics and distance to transit are fixed variables for each area, the other applicable “D” factors (density, diversity, destination accessibility, design and development scale) increase the total to seven “Ds” represented within the model.

The following tables 8-15 provide the average values assigned to each grid cell with a centroid in each TAZ for the current conditions dataset and the three scenarios.

Table 8: Total # of Households in the Half-Mile Buffer (Factor: Density)

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Current</th>
<th>Baseline Scenario</th>
<th>% Change</th>
<th>Smart Scenario</th>
<th>% Change</th>
<th>Infill Scenario</th>
<th>% Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>81</td>
<td>726</td>
<td>794.6</td>
<td>828</td>
<td>921.2</td>
<td>253</td>
<td>212.2</td>
</tr>
<tr>
<td>813</td>
<td>23</td>
<td>37</td>
<td>59.9</td>
<td>38</td>
<td>62.7</td>
<td>45</td>
<td>93.3</td>
</tr>
<tr>
<td>1060</td>
<td>293</td>
<td>293</td>
<td>0.0</td>
<td>293</td>
<td>0.0</td>
<td>483</td>
<td>65.0</td>
</tr>
<tr>
<td>1146</td>
<td>16</td>
<td>16</td>
<td>0.0</td>
<td>16</td>
<td>0.0</td>
<td>16</td>
<td>0.0</td>
</tr>
<tr>
<td>1293</td>
<td>249</td>
<td>249</td>
<td>0.0</td>
<td>249</td>
<td>0.0</td>
<td>757</td>
<td>203.5</td>
</tr>
<tr>
<td>1350</td>
<td>446</td>
<td>458</td>
<td>2.8</td>
<td>495</td>
<td>11.1</td>
<td>531</td>
<td>19.0</td>
</tr>
<tr>
<td>1875</td>
<td>78</td>
<td>441</td>
<td>466.1</td>
<td>114</td>
<td>46.0</td>
<td>87</td>
<td>11.8</td>
</tr>
</tbody>
</table>
Table 9: Net Residential Density in the Half-Mile Buffer (Factor: Density)

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Current</th>
<th>Baseline Scenario</th>
<th>% Change</th>
<th>Smart Scenario</th>
<th>% Change</th>
<th>Infill Scenario</th>
<th>% Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>3.3</td>
<td>4.0</td>
<td>19.4</td>
<td>4.6</td>
<td>37.1</td>
<td>4.8</td>
<td>42.8</td>
</tr>
<tr>
<td>813</td>
<td>3.3</td>
<td>3.6</td>
<td>10.7</td>
<td>3.0</td>
<td>-8.7</td>
<td>2.6</td>
<td>-21.5</td>
</tr>
<tr>
<td>1060</td>
<td>6.9</td>
<td>6.9</td>
<td>0.0</td>
<td>6.9</td>
<td>0.0</td>
<td>5.7</td>
<td>-17.7</td>
</tr>
<tr>
<td>1146</td>
<td>2.3</td>
<td>2.6</td>
<td>14.6</td>
<td>2.2</td>
<td>-5.7</td>
<td>2.3</td>
<td>1.4</td>
</tr>
<tr>
<td>1293</td>
<td>3.6</td>
<td>3.6</td>
<td>0.7</td>
<td>3.6</td>
<td>1.3</td>
<td>8.1</td>
<td>124.7</td>
</tr>
<tr>
<td>1350</td>
<td>5.8</td>
<td>5.8</td>
<td>0.0</td>
<td>6.0</td>
<td>3.7</td>
<td>7.1</td>
<td>22.0</td>
</tr>
<tr>
<td>1875</td>
<td>0.8</td>
<td>0.6</td>
<td>-18.4</td>
<td>0.5</td>
<td>-36.4</td>
<td>0.6</td>
<td>-29.9</td>
</tr>
</tbody>
</table>

Table 10: Employees per Acre in the Half-Mile Buffer (Factor: Density)

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Current</th>
<th>Baseline Scenario</th>
<th>% Change</th>
<th>Smart Scenario</th>
<th>% Change</th>
<th>Infill Scenario</th>
<th>% Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>0.24</td>
<td>0.94</td>
<td>292.2</td>
<td>0.95</td>
<td>296.1</td>
<td>1.39</td>
<td>477.3</td>
</tr>
<tr>
<td>813</td>
<td>1.22</td>
<td>1.75</td>
<td>43.0</td>
<td>1.76</td>
<td>44.4</td>
<td>8.96</td>
<td>633.9</td>
</tr>
<tr>
<td>1060</td>
<td>3.65</td>
<td>3.65</td>
<td>0.0</td>
<td>3.65</td>
<td>0.0</td>
<td>3.65</td>
<td>0.0</td>
</tr>
<tr>
<td>1146</td>
<td>0.05</td>
<td>0.06</td>
<td>1.3</td>
<td>0.06</td>
<td>3.7</td>
<td>0.05</td>
<td>0.0</td>
</tr>
<tr>
<td>1293</td>
<td>3.65</td>
<td>3.65</td>
<td>0.0</td>
<td>3.65</td>
<td>0.0</td>
<td>12.29</td>
<td>237.1</td>
</tr>
<tr>
<td>1350</td>
<td>5.20</td>
<td>5.32</td>
<td>2.3</td>
<td>5.32</td>
<td>2.3</td>
<td>18.28</td>
<td>251.3</td>
</tr>
<tr>
<td>1875</td>
<td>0.05</td>
<td>0.30</td>
<td>537.1</td>
<td>0.22</td>
<td>369.0</td>
<td>0.06</td>
<td>22.6</td>
</tr>
</tbody>
</table>

Table 11: Local Serving Retail in the Half-Mile Buffer (Factor: Destinations)

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Current</th>
<th>Baseline Scenario</th>
<th>% Change</th>
<th>Smart Scenario</th>
<th>% Change</th>
<th>Infill Scenario</th>
<th>% Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>6.5</td>
<td>22.1</td>
<td>240.0</td>
<td>22.1</td>
<td>240.1</td>
<td>16.5</td>
<td>154.7</td>
</tr>
<tr>
<td>813</td>
<td>10.6</td>
<td>21.2</td>
<td>99.6</td>
<td>21.7</td>
<td>104.3</td>
<td>83.7</td>
<td>687.9</td>
</tr>
<tr>
<td>1060</td>
<td>32.7</td>
<td>32.7</td>
<td>0.0</td>
<td>32.7</td>
<td>0.0</td>
<td>34.7</td>
<td>6.2</td>
</tr>
<tr>
<td>1146</td>
<td>0.0</td>
<td>0.1</td>
<td>NA</td>
<td>0.1</td>
<td>NA</td>
<td>0.0</td>
<td>NA</td>
</tr>
<tr>
<td>1293</td>
<td>4.4</td>
<td>4.4</td>
<td>0.1</td>
<td>4.4</td>
<td>0.1</td>
<td>81.9</td>
<td>1753.0</td>
</tr>
<tr>
<td>1350</td>
<td>34.0</td>
<td>34.1</td>
<td>0.4</td>
<td>34.1</td>
<td>0.4</td>
<td>136.4</td>
<td>301.2</td>
</tr>
<tr>
<td>1875</td>
<td>0.2</td>
<td>5.2</td>
<td>2334.2</td>
<td>3.5</td>
<td>1544.3</td>
<td>0.3</td>
<td>29.4</td>
</tr>
</tbody>
</table>
Table 12: Jobs per Household in the Half-Mile Buffer (Diversity)

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Current</th>
<th>Baseline Scenario</th>
<th>% Change</th>
<th>Smart Scenario</th>
<th>% Change</th>
<th>Infill Scenario</th>
<th>% Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>0.50</td>
<td>0.50</td>
<td>-1.1</td>
<td>0.48</td>
<td>-5.4</td>
<td>0.87</td>
<td>73.9</td>
</tr>
<tr>
<td>813</td>
<td>2.84</td>
<td>2.89</td>
<td>2.0</td>
<td>3.30</td>
<td>16.4</td>
<td>12.42</td>
<td>337.5</td>
</tr>
<tr>
<td>1060</td>
<td>1.71</td>
<td>1.71</td>
<td>0.0</td>
<td>1.71</td>
<td>0.0</td>
<td>0.98</td>
<td>-42.4</td>
</tr>
<tr>
<td>1146</td>
<td>3.88</td>
<td>1.74</td>
<td>-55.1</td>
<td>2.28</td>
<td>-41.3</td>
<td>2.94</td>
<td>-24.2</td>
</tr>
<tr>
<td>1293</td>
<td>3.31</td>
<td>3.28</td>
<td>-0.9</td>
<td>3.26</td>
<td>-1.4</td>
<td>3.17</td>
<td>-4.1</td>
</tr>
<tr>
<td>1350</td>
<td>2.60</td>
<td>2.64</td>
<td>1.4</td>
<td>2.54</td>
<td>-2.3</td>
<td>5.95</td>
<td>128.8</td>
</tr>
<tr>
<td>1875</td>
<td>16.63</td>
<td>8.67</td>
<td>-47.9</td>
<td>9.81</td>
<td>-41.0</td>
<td>11.70</td>
<td>-29.6</td>
</tr>
</tbody>
</table>

Table 13: Entropy Index in the Half-Mile Buffer (Factor: Diversity)

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Current</th>
<th>Baseline Scenario</th>
<th>% Change</th>
<th>Smart Scenario</th>
<th>% Change</th>
<th>Infill Scenario</th>
<th>% Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>0.395</td>
<td>0.589</td>
<td>49.3</td>
<td>0.603</td>
<td>52.8</td>
<td>0.538</td>
<td>36.3</td>
</tr>
<tr>
<td>813</td>
<td>0.629</td>
<td>0.695</td>
<td>10.5</td>
<td>0.664</td>
<td>5.6</td>
<td>0.794</td>
<td>26.3</td>
</tr>
<tr>
<td>1060</td>
<td>0.733</td>
<td>0.733</td>
<td>0.0</td>
<td>0.733</td>
<td>0.0</td>
<td>0.645</td>
<td>-12.0</td>
</tr>
<tr>
<td>1146</td>
<td>0.000</td>
<td>0.141</td>
<td>NA</td>
<td>0.066</td>
<td>NA</td>
<td>0.018</td>
<td>NA</td>
</tr>
<tr>
<td>1293</td>
<td>0.581</td>
<td>0.582</td>
<td>0.2</td>
<td>0.582</td>
<td>0.3</td>
<td>0.866</td>
<td>49.0</td>
</tr>
<tr>
<td>1350</td>
<td>0.672</td>
<td>0.659</td>
<td>-1.9</td>
<td>0.664</td>
<td>-1.1</td>
<td>0.845</td>
<td>25.8</td>
</tr>
<tr>
<td>1875</td>
<td>0.053</td>
<td>0.324</td>
<td>505.5</td>
<td>0.289</td>
<td>439.8</td>
<td>0.222</td>
<td>315.0</td>
</tr>
</tbody>
</table>

Table 14: Street Miles in the Half Mile Buffer (Factor: Design)

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Current</th>
<th>Baseline Scenario</th>
<th>% Change</th>
<th>Smart Scenario</th>
<th>% Change</th>
<th>Infill Scenario</th>
<th>% Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>4.8</td>
<td>9.4</td>
<td>97.1</td>
<td>9.1</td>
<td>90.6</td>
<td>6.6</td>
<td>38.6</td>
</tr>
<tr>
<td>813</td>
<td>5.0</td>
<td>6.0</td>
<td>21.1</td>
<td>6.3</td>
<td>25.7</td>
<td>11.1</td>
<td>122.7</td>
</tr>
<tr>
<td>1060</td>
<td>18.3</td>
<td>18.3</td>
<td>0.0</td>
<td>18.3</td>
<td>0.0</td>
<td>22.9</td>
<td>25.3</td>
</tr>
<tr>
<td>1146</td>
<td>1.7</td>
<td>1.9</td>
<td>13.3</td>
<td>1.7</td>
<td>3.2</td>
<td>1.7</td>
<td>0.1</td>
</tr>
<tr>
<td>1293</td>
<td>12.1</td>
<td>12.2</td>
<td>0.1</td>
<td>12.2</td>
<td>0.1</td>
<td>19.7</td>
<td>61.9</td>
</tr>
<tr>
<td>1350</td>
<td>14.9</td>
<td>15.0</td>
<td>0.8</td>
<td>15.0</td>
<td>0.9</td>
<td>23.5</td>
<td>57.9</td>
</tr>
<tr>
<td>1875</td>
<td>1.2</td>
<td>1.7</td>
<td>45.2</td>
<td>1.5</td>
<td>25.5</td>
<td>1.2</td>
<td>2.6</td>
</tr>
</tbody>
</table>
Table 15: Developed Acres in the Half-Mile Buffer (Factor: Development Scale)

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Current</th>
<th>Baseline Scenario</th>
<th>% Change</th>
<th>Smart Scenario</th>
<th>% Change</th>
<th>Infill Scenario</th>
<th>% Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>116.3</td>
<td>327.4</td>
<td>181.6</td>
<td>307.9</td>
<td>164.8</td>
<td>202.3</td>
<td>74.0</td>
</tr>
<tr>
<td>813</td>
<td>297.5</td>
<td>347.6</td>
<td>16.8</td>
<td>366.1</td>
<td>23.1</td>
<td>586.1</td>
<td>97.0</td>
</tr>
<tr>
<td>1060</td>
<td>267.7</td>
<td>267.7</td>
<td>0.0</td>
<td>267.7</td>
<td>0.0</td>
<td>474.2</td>
<td>77.1</td>
</tr>
<tr>
<td>1146</td>
<td>32.6</td>
<td>44.1</td>
<td>35.1</td>
<td>37.2</td>
<td>14.1</td>
<td>32.8</td>
<td>0.6</td>
</tr>
<tr>
<td>1293</td>
<td>266.5</td>
<td>266.9</td>
<td>0.2</td>
<td>266.9</td>
<td>0.2</td>
<td>588.8</td>
<td>120.9</td>
</tr>
<tr>
<td>1350</td>
<td>310.2</td>
<td>314.0</td>
<td>1.2</td>
<td>313.8</td>
<td>1.2</td>
<td>645.1</td>
<td>108.0</td>
</tr>
<tr>
<td>1875</td>
<td>76.0</td>
<td>122.9</td>
<td>61.8</td>
<td>102.4</td>
<td>34.7</td>
<td>83.0</td>
<td>9.2</td>
</tr>
</tbody>
</table>

Elasticity Comparisons

The comparison these results to elasticities is problematic for several reasons: 1) the interrelationship of independently estimated elasticities cannot be assumed to be linear, and 2) appropriate bounding limits on elasticities need to be defined because otherwise the elasticities have the potential to over-represent changes that may occur in scenarios with proportionally large changes.

Table 16(below) presents a comparison of changes in VMT predicted by the UPlan implementation of the Ds tools and the range of values that might be expected. The effects of applying elasticities to the measured average values for each of the TAZs in each of the three scenarios has also been calculated. The following assumptions were made in applying the elasticities (note: these guidelines and elasticities are adopted from the CAPCOA report.)

1. The percent change in a measure would be capped at 500%
2. The maximum allowed percent change in the VMT from any particular measure would be 30%

The following elasticities were assumed.

- Density: 0.07
- Destination Accessibility: 0.2
- Diversity: 0.09
- Design: 0.12

---

Some of these elasticities are similarly challenging to apply to the measures used by this toolkit. Notably, Destination Accessibility is intended to measure the distance to desired destinations. In this case, the effects were calculated on the amount of local serving retail in a half-mile buffer.

Table 16: Percent Change from the UPlan Implantation, Mean, Minimum, Maximum and Summed Percent Changes by Elasticities in VMT by Scenario.

<table>
<thead>
<tr>
<th>TAZ</th>
<th>% Change</th>
<th>Mean</th>
<th>Min</th>
<th>Max</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>13.9</td>
<td>14.0</td>
<td>-0.1</td>
<td>30.0</td>
<td>97.8</td>
</tr>
<tr>
<td>813</td>
<td>31.9</td>
<td>4.5</td>
<td>0.2</td>
<td>19.9</td>
<td>31.5</td>
</tr>
<tr>
<td>1060</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1146</td>
<td>3.4</td>
<td>4.6</td>
<td>-5.0</td>
<td>30.0</td>
<td>27.7</td>
</tr>
<tr>
<td>1293</td>
<td>0.0</td>
<td>0.0</td>
<td>-0.1</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>1350</td>
<td>-0.4</td>
<td>0.1</td>
<td>-0.2</td>
<td>0.2</td>
<td>0.5</td>
</tr>
<tr>
<td>1875</td>
<td>25.7</td>
<td>17.1</td>
<td>-4.3</td>
<td>30.0</td>
<td>119.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TAZ</th>
<th>% Change</th>
<th>Mean</th>
<th>Min</th>
<th>Max</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>14.5</td>
<td>14.1</td>
<td>-0.5</td>
<td>30.0</td>
<td>98.5</td>
</tr>
<tr>
<td>813</td>
<td>29.9</td>
<td>4.7</td>
<td>-0.6</td>
<td>20.9</td>
<td>32.8</td>
</tr>
<tr>
<td>1060</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1146</td>
<td>1.2</td>
<td>8.1</td>
<td>-3.7</td>
<td>30.0</td>
<td>56.5</td>
</tr>
<tr>
<td>1293</td>
<td>0.0</td>
<td>0.0</td>
<td>-0.1</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>1350</td>
<td>-2.5</td>
<td>0.2</td>
<td>-0.2</td>
<td>0.8</td>
<td>1.1</td>
</tr>
<tr>
<td>1875</td>
<td>10.2</td>
<td>12.3</td>
<td>-3.7</td>
<td>30.0</td>
<td>85.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TAZ</th>
<th>% Change</th>
<th>Mean</th>
<th>Min</th>
<th>Max</th>
<th>Sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>32.3</td>
<td>13.2</td>
<td>3.0</td>
<td>30.0</td>
<td>92.4</td>
</tr>
<tr>
<td>813</td>
<td>56.2</td>
<td>16.0</td>
<td>-1.5</td>
<td>30.0</td>
<td>112.1</td>
</tr>
<tr>
<td>1060</td>
<td>12.9</td>
<td>0.4</td>
<td>-3.8</td>
<td>4.5</td>
<td>2.7</td>
</tr>
<tr>
<td>1146</td>
<td>0.0</td>
<td>4.7</td>
<td>-2.2</td>
<td>30.0</td>
<td>27.9</td>
</tr>
<tr>
<td>1293</td>
<td>67.0</td>
<td>11.6</td>
<td>-0.4</td>
<td>30.0</td>
<td>81.0</td>
</tr>
<tr>
<td>1350</td>
<td>83.1</td>
<td>10.2</td>
<td>1.3</td>
<td>30.0</td>
<td>71.3</td>
</tr>
<tr>
<td>1875</td>
<td>1.3</td>
<td>4.6</td>
<td>-2.7</td>
<td>28.3</td>
<td>32.2</td>
</tr>
</tbody>
</table>

The mean elasticity based calculation is not a perfect measure. It is an unequal combination of several factors including: several density, two diversity, a design, and a
destination accessibility measure; but it is indicative of the measures being used in the tool, though of not their relative strengths.

The calculations presented are based on the percent change in the value measured and fed into the VMT calculator. A 100% change is a doubling of the value, 50% is an increase by half, and a -50% change would be cutting the value in half. The percent changes in per household VMT produced by the UPlan implementation fall within a reasonable range compared to the available literature.

It should be noted that the products of this project do not have the same maximum limitations imposed that are recommended by the CAPCOA report. With only three exceptions, the percent change presented by the UPlan implementation fall under the summed elasticity calculated value. That summed value would exceed the maximum change if all of the elasticities represented completely independent factors in VMT generation, which is not the case. In those cases where the % change exceeds the summed value, two are cases where the 30% maximum change is playing a role, and the last is an area of infill and redevelopment in a complex environment resulting in mixed elasticity based values that largely cancel each other’s effects out.

Tables 17-23 present the expected effects of the change in the parameter as a percentage reduction in per household VMT based on each parameter’s change. All values are calculated based on changes to the average value of the parameters within the TAZ, and each of those parameters is a summary of the land use within a half-mile.

**Table 17: Elasticity Effects Based on Total Housing Units**

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Baseline Scenario</th>
<th>Smart Growth Scenario</th>
<th>Intensive Infill Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>30.0</td>
<td>30.0</td>
<td>14.9</td>
</tr>
<tr>
<td>813</td>
<td>4.2</td>
<td>4.4</td>
<td>6.5</td>
</tr>
<tr>
<td>1060</td>
<td>0.0</td>
<td>0.0</td>
<td>4.5</td>
</tr>
<tr>
<td>1146</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1293</td>
<td>0.0</td>
<td>0.0</td>
<td>14.2</td>
</tr>
<tr>
<td>1350</td>
<td>0.2</td>
<td>0.8</td>
<td>1.3</td>
</tr>
<tr>
<td>1875</td>
<td>30.0</td>
<td>3.2</td>
<td>0.8</td>
</tr>
</tbody>
</table>

**Table 17: Elasticity Effects Based on Net Residential Density**

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Baseline Scenario</th>
<th>Smart Growth Scenario</th>
<th>Intensive Infill Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>1.4</td>
<td>2.6</td>
<td>3.0</td>
</tr>
<tr>
<td>813</td>
<td>0.8</td>
<td>-0.6</td>
<td>-1.5</td>
</tr>
<tr>
<td>1060</td>
<td>0.0</td>
<td>0.0</td>
<td>-1.2</td>
</tr>
</tbody>
</table>
### Table 18: Elasticity Effects Based on Employees Per Acre

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Baseline Scenario</th>
<th>Smart Growth Scenario</th>
<th>Intensive Infill Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>20.5</td>
<td>20.7</td>
<td>30</td>
</tr>
<tr>
<td>813</td>
<td>3.0</td>
<td>3.1</td>
<td>30.0</td>
</tr>
<tr>
<td>1060</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1146</td>
<td>0.1</td>
<td>0.3</td>
<td>0.0</td>
</tr>
<tr>
<td>1293</td>
<td>0.0</td>
<td>0.0</td>
<td>16.6</td>
</tr>
<tr>
<td>1350</td>
<td>0.2</td>
<td>0.2</td>
<td>17.6</td>
</tr>
<tr>
<td>1875</td>
<td>30.0</td>
<td>25.8</td>
<td>1.6</td>
</tr>
</tbody>
</table>

### Table 19: Elasticity Effects Based on Acres of Local Serving Retail

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Baseline Scenario</th>
<th>Smart Growth Scenario</th>
<th>Redevelopment</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>813</td>
<td>19.9</td>
<td>20.9</td>
<td>30.0</td>
</tr>
<tr>
<td>1060</td>
<td>0.0</td>
<td>0.0</td>
<td>1.2</td>
</tr>
<tr>
<td>1146</td>
<td>30.0</td>
<td>30.0</td>
<td>30.0</td>
</tr>
<tr>
<td>1293</td>
<td>0.0</td>
<td>0.0</td>
<td>30.0</td>
</tr>
<tr>
<td>1350</td>
<td>0.1</td>
<td>0.1</td>
<td>30.0</td>
</tr>
<tr>
<td>1875</td>
<td>30.0</td>
<td>30.0</td>
<td>5.9</td>
</tr>
</tbody>
</table>

### Table 20: Elasticity Effects Based on Jobs per Household

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Baseline Scenario</th>
<th>Smart Growth Scenario</th>
<th>Intensive Infill Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>-0.1</td>
<td>-0.5</td>
<td>6.6</td>
</tr>
<tr>
<td>813</td>
<td>0.2</td>
<td>1.5</td>
<td>30.0</td>
</tr>
<tr>
<td>1060</td>
<td>0.0</td>
<td>0.0</td>
<td>-3.8</td>
</tr>
<tr>
<td>1146</td>
<td>-5.0</td>
<td>-3.7</td>
<td>-2.2</td>
</tr>
<tr>
<td>1293</td>
<td>-0.1</td>
<td>-0.1</td>
<td>-0.4</td>
</tr>
<tr>
<td>1350</td>
<td>0.1</td>
<td>-0.2</td>
<td>11.6</td>
</tr>
<tr>
<td>1875</td>
<td>-4.3</td>
<td>-3.7</td>
<td>-2.7</td>
</tr>
</tbody>
</table>
Table 21: Elasticity Effects Based on Land use Entropy Index

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Baseline Scenario</th>
<th>Smart Growth Scenario</th>
<th>Intensive Infill Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>4.4</td>
<td>4.7</td>
<td>3.3</td>
</tr>
<tr>
<td>813</td>
<td>0.9</td>
<td>0.5</td>
<td>2.4</td>
</tr>
<tr>
<td>1060</td>
<td>0.0</td>
<td>0.0</td>
<td>-1.1</td>
</tr>
<tr>
<td>1146</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>1293</td>
<td>0.0</td>
<td>0.0</td>
<td>4.4</td>
</tr>
<tr>
<td>1350</td>
<td>-0.2</td>
<td>-0.1</td>
<td>2.3</td>
</tr>
<tr>
<td>1875</td>
<td>30.0</td>
<td>30.0</td>
<td>28.3</td>
</tr>
</tbody>
</table>

Table 22: Elasticity Effects Based on Miles of Street

<table>
<thead>
<tr>
<th>TAZ</th>
<th>Baseline</th>
<th>Smart Growth</th>
<th>Redevelopment</th>
</tr>
</thead>
<tbody>
<tr>
<td>392</td>
<td>11.6</td>
<td>10.9</td>
<td>4.6</td>
</tr>
<tr>
<td>813</td>
<td>2.5</td>
<td>3.1</td>
<td>14.7</td>
</tr>
<tr>
<td>1060</td>
<td>0.0</td>
<td>0.0</td>
<td>3.0</td>
</tr>
<tr>
<td>1146</td>
<td>1.6</td>
<td>0.4</td>
<td>0.0</td>
</tr>
<tr>
<td>1293</td>
<td>0.0</td>
<td>0.0</td>
<td>7.4</td>
</tr>
<tr>
<td>1350</td>
<td>0.1</td>
<td>0.1</td>
<td>6.9</td>
</tr>
<tr>
<td>1875</td>
<td>5.4</td>
<td>3.1</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Some interesting features arose that could be considered unintended consequences of the scenarios. In particular, the intensive infill scenario actually decreases the net residential density in many of the city edge TAZs because the rural residential areas surrounding the cities have not been diluted by the inclusion of large numbers higher density units. The core area TAZs do have density increases.

In summary, these tools work successfully and produce reasonable results given the inputs in the test area. However, it definitely would be preferable to conduct additional testing and deployment in other locations of California. Further, we do not recommend that absolute values of VMT be published from these tools because doing so is inconsistent with their intended application. Instead, users should consider the relative effects of each scenario, e.g. in TAZ 392, the baseline growth scenario reduced per household VMT by 13.9%, the smart growth by 14.5%, and the intensive infill scenario by 32.3%.
7. Ancillary Products

A. Place Type Translator

The project team determined that it was necessary to address the need for “place type translation” to provide a common set of terms and definitions for various local jurisdictions’ land use zoning classifications and definitions. The following formulation for place type conversions evolved to meet some of those needs.

The underlying idea is that for a particular set of place types for which there is specific descriptive information and an expected range of values, it is possible to calculate the range-normalized “n-space” distance between a description of a location and all of the place types in the set regarding any other “place” description. If the place type that is “closest” to that location, it will provide the closest match between that particular place and the entire place type system.

This method has significant theoretical similarities to multiple linear regression methods or other supervised classification algorithms.

A prototype of this algorithm has been assembled and is available for use, or for refinement under the Apache 2.0 license.

The code and demonstration can be downloaded from:

http://downloads.ice.ucdavis.edu/ultrans/statewidetools/PT_Translator.zip

Acknowledgements:

The Urban Land Use and Transportation Center would like to thank:

The California Department of Transportation for funding this project, and in particular thank Terry Parker, our esteemed colleague and tireless cheerleader, quarterback, and coach.

Raef Porter, Sacramento Area Council of Governments, who provided the needed administrative, moral, and technical support in the development of these tools.

Jerry Walters and the whole team at Fehr & Peers for doing much of the heavy lifting on the statistical analysis, model estimation, and all around guidance.

Michael McCoy for being the principal investigator at ULTRANS and letting me (Nathaniel) have fairly free reign with my time and effort on this project.

Laurel Torney and Kyle Shipley who assisted on this project in many way, most of them being tedious.