San Diego Association of Governments Subregional Allocation Model

Phase 1 Model: Final Report

Task Order 14

June 25, 2015

Table of Contents

[Introduction 2](#_Toc423003602)

[Section 1: Real Estate Price Model 3](#_Toc423003603)

[CODE AND OPERATION 3](#_Toc423003604)

[LINK WITH PECAS 5](#_Toc423003605)

[SPECIFICATION 6](#_Toc423003606)

[Section 2: Location Choice Model 8](#_Toc423003607)

[CODE AND OPERATION 9](#_Toc423003608)

[LINK WITH PECAS 12](#_Toc423003609)

[SPECIFICATION 12](#_Toc423003610)

[Section 3: Proforma Real Estate Developer Model 13](#_Toc423003611)

[CODE AND OPERATION 14](#_Toc423003612)

[LINK WITH PECAS 17](#_Toc423003613)

[VALIDATION 17](#_Toc423003614)

[Appendix 1: Real Estate Price Model Coefficients 19](#_Toc423003616)

[Appendix 2: Location Choice Model Coefficients 23](#_Toc423003617)

[Appendix 3: Installation Instructions for the sandag\_urbansim Model 32](#_Toc423003618)

Appendix 4: Memorandum 1 – Data…………………………………………………………………….……………………………….35

Appendix 5: Memorandum 2 – Model Development…………………………………………………………………………….47

# Introduction

This document aims to familiarize the reader with the current state of the proof-of-concept Subregional Allocation Model, describe code and key workflows, and report on estimation and calibration of the model system. It discusses the process of model estimation, calibration, aspects of data development, and how these processes can be replicated by running scripts. The content contained herein constitutes Technical Memorandums 3, 4, and 5. For an overview of and background on the proposed model system, the reader is encouraged to first read Memorandum 2. For a deeper dive into data requirements, please see Memorandum 1. Memorandums 1 and 2 are included in the appendix for reference.

This project to develop an UrbanSim-oriented model for the San Diego region was initiated by the San Diego Association of Governments (SANDAG). UrbanSim is a modeling system developed to support the need for analyzing the potential effects of land use policies and infrastructure investments on the development and character of cities and regions. The system has been developed using the Python programming language and supporting libraries, and is licensed as open sourced software. UrbanSim has been applied in a variety of metropolitan areas in the United States and abroad, including Albuquerque, Denver, Detroit, Eugene-Springfield, Honolulu, Houston, Paris, Phoenix, Salt Lake City, Seattle, the San Francisco Bay Area and Zürich.

This application of UrbanSim to the San Diego region has been developed in close collaboration with SANDAG staff, without which this project could not have been possible. The focus of the model development effort was on real estate supply-side predictions. Location choice models and real estate price models were implemented so as to provide data inputs for the UrbanSim proforma-based developer model. The SANDAG UrbanSim model development process began in mid-fall of 2014, and this is the first draft model, to be used for evaluating the system and input data. Future model development phases may revise certain data pieces or model design choices, re-estimate certain models, and conduct additional calibration, validation, and sensitivity testing.

Code referenced in this document can be found on GitHub: <https://github.com/synthicity/sandag_urbansim>

Section 1 of this document describes code and statistical estimation of the real estate price model. Section 2 covers the same topics for the location choice models. The final section discusses code, calibration, and initial validation of the real estate developer model. Results and background memos are contained in the appendix.

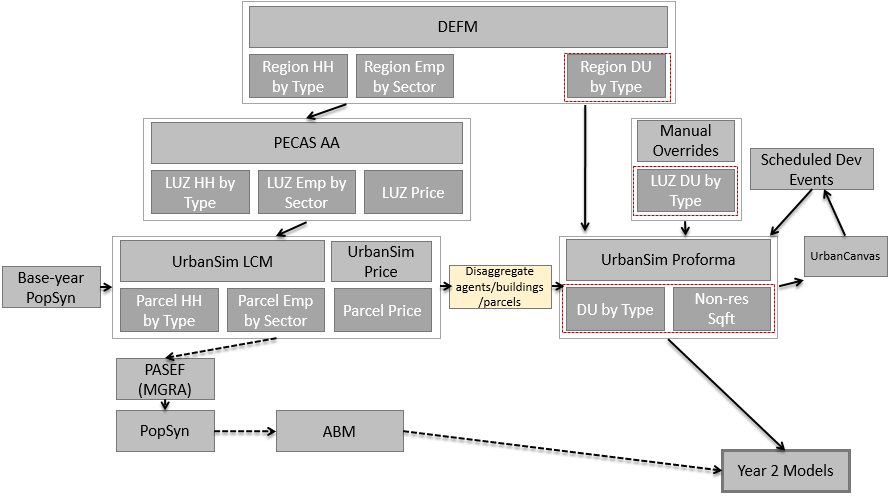


Figure 1: Model system diagram

# Section 1: Real Estate Price Model

The Real Estate Price Model (REPM) predicts the price per square foot of each building. For residential space, price is expressed in terms of sale price. For non-residential space, price is expressed in terms of rent. UrbanSim uses real estate prices as the indicator of the match between demand and supply of land at different locations and with different development types, and of the relative market valuations for attributes of housing, nonresidential space, and location. This role is important to the rationing of land and buildings to consumers based on preferences and ability to pay, as a reflection of the operation of actual real estate markets. Also, importantly, any adjustment in prices alters real estate developer preferences for the location, type, and density of new construction. Note that since PECAS handles region-to-LUZ (an intermediate modeling geography) allocation of households and employment, the building-level price model’s influence on the demand side location choice models is more limited in this model application than in the typical UrbanSim model: demand agents have limited spatial scope to respond to changes in predicted building-level prices (or must do so via PECAS) because of the LUZ controls. An adjustment in prices can alter sub-LUZ location preferences, but will not have an effect on regional patterns of households and jobs except via price’s influence on the location and level of new real estate construction. Additional overview of the REPM can be found in Memorandum 2 in the appendix.

The REPM is an Ordinary Least Squares (OLS) hedonic regression model that predicts the price of residential and non-residential square footage. The dependent variable is the natural log of price. Separate regressions are run for each development type. The REPM updates the price attributes in the building table, but instead of storing price as the natural log of price (this is what is actually estimated and predicted), the REPM automatically exponentiates the predicted value so that the per-square-foot prices stored in the tables are more human interpretable. Prices are updated annually in each simulation year. In the future, a price equilibration step can be considered (in conjunction with the location choice models) if LUZ controls are removed.

### CODE AND OPERATION

The entry-points for the REPM code lives in /sandag\_urbansim/models.py. Recall that each ‘model’ in UrbanSim is now a function registered with the simulation framework. In models.py see the functions named ‘rsh\_estimate’, ‘rsh\_simulate’, ‘nrh\_simulate’, ‘nrh\_estimate2’, and ‘nrh\_simulate2’. These are the functions called by the simulation framework when running price model estimation or price model simulation. The arguments to these functions specify the YAML configuration file to read from, the data tables involved, and the building table price output column name. In the functions listed above, the reader may wonder why ‘nrh\_estimate’ is not listed. This function lives in /urbansim\_defaults/models.py. Wherever possible, we import code from urbansim\_defaults so as to share code across UrbanSim implementations. When a function in sandag\_urbansim has the same name as a function in urbansim\_defaults, the function in sandag\_urbansim takes priority and is used in place of the urbansim\_defaults function.

As currently configured in the Simulation notebook (/sandag\_urbansim/notebooks/Simulation.ipynb), the price models run near the beginning of each simulation year. They are run prior to demand agent allocation and the real estate supply models, which they influence. The price models are influenced by the previous year’s demand agent allocation and the previous year’s real estate supply construction. The sequence of models to run each year can be reordered by editing the Simulation notebook.

The real estate price model is simulated on the UrbanSim buildings table. However, it is estimated off of separate datasets. The estimation datasets are: non-residential CoStar records with rent information (the ‘costar’ table), residential sales transaction records from the county assessor (the ‘assessor\_transactions’ table). This data originally comes from .csv files provided by SANDAG: price/costar2012.csv, price/priceDataSet.csv. These input .csv files are processed and loaded by /scripts/load.py (lines 70 – 74) and /scripts/process.py (lines 165 – 185 and lines 207 – 236). The PECAS LUZ price table is similarly loaded and processed.

As mentioned, the price model estimation datasets are currently the ‘costar’ table (for non-residential) and the ‘assessor\_transactions’ table (for residential). If the user desires to add another price model estimation dataset, add the new table as a function in /sandag\_urbansim/datasources.py. Make sure to decorate the table function with an @sim.table decorator (see the existing examples in datasources.py). Typically these table functions read from the HDF5 file (‘store’), but table functions can read from .csv or a database as well if that is more convenient (e.g. use pd.read\_csv within the function).

Re-estimating the residential price model regression equations is straight-forward. Open the IPython notebook at /sandag\_urbansim/notebooks/Estimation.ipynb and execute the appropriate cells in the annotated notebook. Prior to any estimation run, execute the top two cells of the Estimation notebook (imports and accessibility engine initialization). To re-estimate the residential hedonics, run the third code cell (rsh\_estimate) in the notebook. To re-estimate the non-residential hedonics, run the 5th and 7th code cells (nrh\_estimate, nrh2\_estimate). The price model specifications (and resulting estimated coefficents) are stored in YAML configuration files in the /sandag\_urbansim/configs directory: rsh.yaml, nrh.yaml, nrh2.yaml. To re-specify the models (e.g. add or drop explanatory variables, redesign segmentation, adjust filters), open these YAML files in a text editor and make desired changes. Each segment has a ‘model\_expression’ node where the explanatory variable set can be adjusted. For any segment that lacks a ‘model\_expression’ node, the specification defaults to the model\_expression shown in the ‘default\_config’ section. Model expressions are written in patsy syntax (see github.com/pydata/patsy). In each price model YAML file, the ‘fit\_filters’ setting shows the filter on observations in estimation (only observations that pass all filters get used in estimation). The ‘predict\_filters’ setting shows the filter on records in simulation (only building records that pass the filter get a value predicted). The ‘segmentation\_col’ setting specifies the variable (either a primary or computed attribute on the buildings table) that will be used to define segments. A separate regression equation is estimated for each segment. Here, the segments are based on development\_type\_id. Estimation results are stored in the YAML file after each time estimation is executed from the Estimation notebook.

The variable names in the model\_expression for each segment of the price model are typically defined in /sandag\_urbansim/variables.py (unless the variable is a network accessibility variable, in which case it will be defined in /sandag\_urbansim/configs/neighborhood\_vars.yaml). The variables.py file is where additional variables can be defined and then used by the model\_expression. Note that the urbansim\_defaults repository also contains a variables.py- the variables defined here can be referenced in the model\_expression as well if the variable name is not already used in sandag\_urbansim.

Explanatory variables in the price models such as distance to coast, distance to freeway/onramp, distance to school, distance to park, and distance to transit come from pre-calculated parcel-level distances provided by SANDAG. The file located at /space/local\_effect\_distances.csv is loaded and the distance fields are joined to the parcels tables. See lines 56-57 of /scripts/load.py and lines 31-40 of /scripts/process.py. In the future, it may be desirable to update these distance fields over the course of the simulation. For example, when a new school is inserted by scheduled\_development\_events (i.e. SiteSpec) for a future year, this new school can influence nearby prices if the ‘distance\_to\_school’ column in the ‘parcels’ table is updated.

Network accessibility variables to be used as explanatory variables in the price model should be defined in /sandag\_urbansim/configs/neighborhood\_vars.yaml. Note that the network that these variables are run on is contained in /sandag\_urbansim/data/osm\_sandag.h5. This HDF5 file contains a ‘nodes’ DataFrame and an ‘edges’ DataFrame, and currently these refer to the OpenStreetMap network. To swap in a different network, update the osm\_sandag.h5 file. See the pandana documentation for network table requirements.

The real estate price model serves an important purpose in the model system because prices are a key input to the proforma model of real estate development. As prices increase, so does the likelihood of new real estate development, subject to development constraints such as zoning. Prices mediate between demand and supply. Predictions from the real estate price model populate price columns on the building table, which the proforma model then summarizes and uses as input. The proforma model summarizes price model predictions as defined in /sandag\_urbansim/configs/price\_vars.yaml. This is where network accessibility queries utilized by the proforma model are defined. See for example, the variables in the price\_vars YAML file with name ‘residential’ and ‘office’.

### LINK WITH PECAS

The UrbanSim building-level price model is influenced by LUZ-level PECAS-generated prices via an explanatory variable in the model specification. See the ‘pecas\_price’ variable in rsh.yaml, nrh.yaml, and nrh2.yaml. By including the PECAS price (which can be understood as a measure of relative LUZ attractiveness), building level prices become sensitive to fluctuations in PECAS price predictions over the course of the simulation. See Figure 2. This provides some integration between the models in the system. As particular locations in the SANDAG region become more attractive in PECAS, UrbanSim prices will be influenced. For example, if PECAS increases prices in a given LUZ, building-level prices predicted by the UrbanSim hedonic will, other things equal, also increase, and this will result in a higher probability of real estate development in that LUZ. In the current model system, PECAS primarily drives the real estate demand-side and UrbanSim primarily handles the supply side. The presence of this pecas\_price explanatory variable is one avenue in the system by which the demand-side influences the supply-side. Beyond the ‘pecas\_price’ explanatory variable, other connections between the demand-side and supply-side come via explanatory variables in the price model specifications such as ‘jobs\_400m’ (number of jobs within 400 meters along the local street network) and occupancy rate variables (‘res\_occupancy\_3000m’, ‘nonres\_occupancy\_3000m’). These additional variables are the result of LUZ demand agent totals (from PECAS) being allocated to the building-level (by the UrbanSim location choice models).

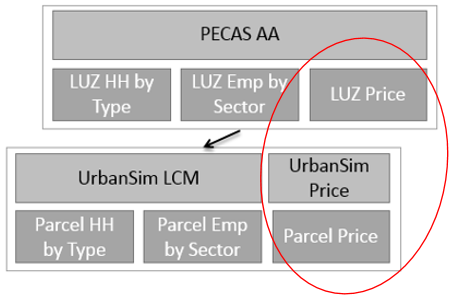


Figure 2: PECAS LUZ price influences the disaggregate UrbanSim price via the ‘pecas\_price’ explanatory variable

If it is desired to remove the link between PECAS prices and UrbanSim prices as represented by the ‘pecas\_price’ variable, simply open the price model YAML files in a text editor and delete ‘pecas\_price’ from the model expressions, then re-estimate from the Estimation notebook.

The code-based definition of the pecas\_price explanatory variable can be found in /sandag\_urbansim/variables.py. See lines 189 to 198. PECAS prices by LUZ and development type are disaggregated to the building level according to building location and building development type. Also see the ‘pecas\_price’ variables defined in variables.py for the CoStar dataset and the assessor residential transactions dataset. These are the estimation datasets for the real estate price model, and PECAS prices need to be associated with the estimation datasets in order to include the ‘pecas\_price’ explanatory variable in the specification.

The ‘pecas\_prices’ table referenced by the ‘pecas\_price’ variable definition is located in the HDF5 store (this originally comes from luz\_controls/pecas\_PriceAndSpaceQuantity.csv). It is loaded into the simulation by /sandag\_urbansim/datasources.py. See lines 112 through 116. It is currently a static table that gives PECAS prices by LUZ and development\_type\_id for every simulation year. The ‘pecas\_prices’ table should be updated over the course of the simulation during integrated PECAS-UrbanSim runs. This would be achieved by defining a simple model that reads the updated PECAS prices from whatever file format or database it is exported to by PECAS, ensuring that the the resulting DataFrame (e.g. new\_pecas\_prices) is formatted like the existing pecas\_prices table, and then running ‘sim.add\_table(“pecas\_prices”, new\_pecas\_prices)’.

### SPECIFICATION

The above paragraphs have noted some of the pecas\_price, demand agent, and occupancy variables in the price model specifications. Other explanatory variables in the specifications include distance to coast, distance to freeway/onramp, distance to school, distance to park, distance to transit, measures of year built, number of stories, average household income within some distance, and other demographic characteristics of each location. For example, see Figure 3 for explanatory variables (with a + or – denoting the sign of coefficient) in the non-residential price model segments. PECAS prices by development type provide a connection between PECAS and UrbanSim prices. The jobs\_within\_N\_meters variables capture existing density and existing demand. Prices are hypothesized to be positively associated with density, income, accessibility, stories, and PECAS price. Historic structures are also hypothesized to exhibit a positive relationship with price.



Figure 3- Coefficient sign by variable and non-residential price submodel

See a table of real estate price model estimated coefficients by segment in Appendix 1 at the end of this document. Choice of segments was determined by sample size in the estimation datasets. Because the hedonic regressions include explanatory variables that are maintained as part of the simulation system, these estimated regression equations are used to update relative prices during simulation.

Figure 4 shows average non-residential rent per square foot by LUZ as predicted by the price model for year 2013. Figure 5 shows average residential price per square foot by LUZ. Unadjusted price predictions were reviewed by SANDAG staff and deemed reasonable in overall pattern.

Figure 4: Average non-residential rent per square foot by LUZ, year 2013 (colors represent quantiles)

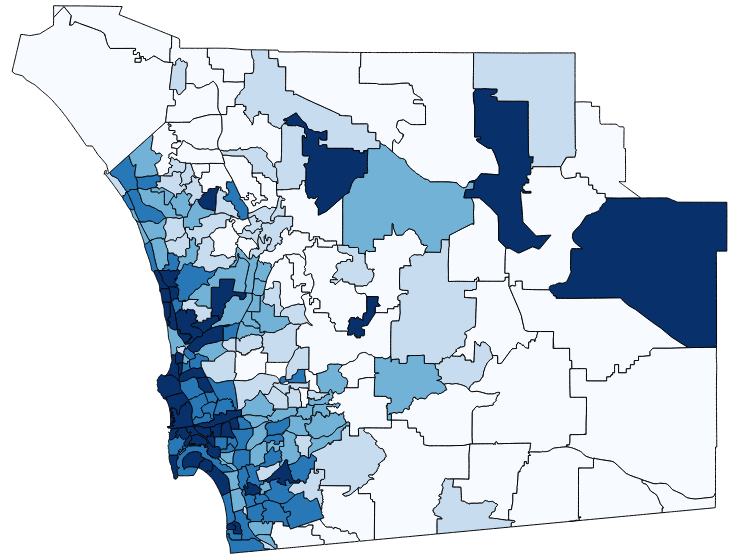
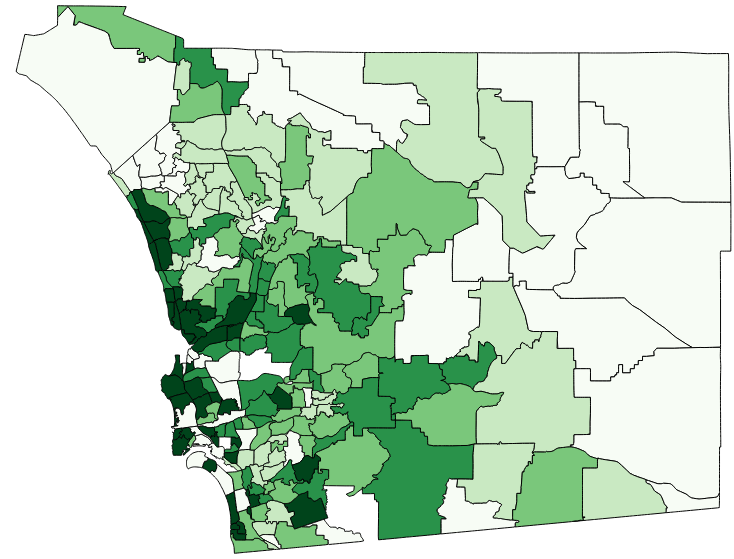


Figure 5: Average residential price per square foot by LUZ, year 2013 (colors represent quantiles)



Note that the real estate price model’s predictions may be modified by the proforma model of real estate supply. Any price adjustments are as perceived by developers-only (the building table is not updated). First, the predicted prices may be adjusted according to ‘price shifters’ that are set during calibration. See Table 1 for price shifter values that result from supply model calibration. Secondly, the proforma may scale prices upward (by a single-factor regionally, preserving subregional variation in price predictions) to ensure that there are enough profitable units in the region to match the overall DEFM control totals for new built space. Both these price adjustment behaviors can be turned off: for example, calibrated price shifters can all be set to 1.0 in /sandag\_urbansim/data/calibration/msa\_shifters.csv.

Table 1: Prices as perceived by developers are adjusted by calibrated price shifters

|  |  |  |
| --- | --- | --- |
| msa\_id | res\_price\_shifter | nonres\_price\_shifter |
| 0 | 0.28 | 0.46 |
| 1 | 0.25 | 0.45 |
| 2 | 0.5 | 0.64 |
| 3 | 2.15 | 2.3 |
| 4 | 1.73 | 0.96 |
| 5 | 1.63 | 2.26 |
| 6 | 2.15 | 1 |

# Section 2: Location Choice Models

The location choice models predict the probability that demand agents (households, jobs) will choose a particular building within each LUZ, then simulates the choice of building. The model is specified as a multinomial logit (MNL) model. The household location choice model (HLCM) allocates households, and the employment location choice model (ELCM) allocates jobs. See Memorandum 2 in the appendix for an additional overview of the location choice models.

In the SANDAG model system, the location choice models serve two primary purposes. First, the location choice model influences prices by affecting vacancy rates and the number of agents in each location. Through their influence on price, the location choice models have an impact proforma predictions. Secondly, the location choice models provide MGRA level household estimates that PASEF consumes to generate detailed demographic predictions.

Simulation of the demand-side UrbanSim location choice models involves no calibration in the SANDAG model system. This is because SANDAG’s calibrated PECAS model handles region-to-LUZ allocation. Broad demand-side spatial patterns are determined by PECAS, and UrbanSim’s influence is only at the sub-LUZ level. Calibrating the location choice models in their sub-LUZ allocation role would not offer much benefit.

The location choice models are estimated regionally, not separately by LUZ. Estimated coefficients reflect the location patterns of demand agents for the entire region. Location choice models are then simulated by LUZ based on these regional coefficients. For example, in the household location choice model (HLCM), which is constrained by LUZ, estimation takes place with every building in the region in the chooser alternative set, but simulation takes place with only LUZ buildings in the alternative set. LUZ-level estimation of the location choice models would not be feasible due to sample size (in many cases) and the difficulty of specifying separate models by 220+ LUZs.

Note that the model is currently set up to use static PECAS-generate household controls by year and there is no live feedback between the real estate supply model and PECAS until integrated runs are attempted. This means that, in the current model setup, the LUZ level household controls in future simulation years may become inconsistent with where new built-space is being predicted by UrbanSim. This problem will be avoided in integrated runs because there will be feedback between the models, and PECAS will respect UrbanSim real estate supply predictions (LUZ household totals will not exceed LUZ dwelling unit totals). Until this feedback is in place, households may go unplaced in some LUZs if not enough vacant units exist in the LUZ to accommodate them.

Household demand-allocation does not synthesize a future-year population, but instead transitions the base-year synthesized population (from SANDAG’s PopSyn2 implementation) forward. The population is transitioned forward in annual time steps to match the LUZ-level control totals for each year. And the location choice models then allocate from LUZ to building. The implemented household transition model works in the following way: The model compares the total number of households (currently by LUZ and activity\_id) in the households table at the beginning of a simulation year, to the total number of households (by LUZ and activity\_id) specified in the annual household control totals for that year. If the control total value is higher, the model adds the necessary number of households to the household table by sampling existing households (of the same type) and duplicating them. If the control totals indicate a declining household count (by type) then the appropriate number of households in the data are selected at random and removed. The role of this model is to keep the household data in the simulation synchronized with LUZ-level aggregate expectations of population and households received from PECAS. Note that the transition model can be configured by the choice of what columns to have in the annual control totals. For example, control totals include an activity\_id column, which captures household size and income, and this could be used to simulate a region-wide decline in average household size.

A couple additional notes related to demand agent allocation in the current simulation. The transition models are the primary determinant of how many unplaced demand agents there are. But another source of unplaced demand agents is building demolition. If the developer model demolishes structures that contain household or jobs, those agents must select a new building within the LUZ the following simulation year. Also note that whereas most UrbanSim model systems have agent relocation models, the SANDAG model does not, because relocation models do not make sense in the context of PECAS handling most of the demand-side (e.g. if a household relocated, they would currently always be placed back within the same LUZ so as not to violate PECAS LUZ controls).

### CODE AND OPERATION

The entry-point for the location choice model code is in /sandag\_urbansim/models.py and /urbansim\_defaults/models.py. See the ‘hlcm\_luz\_estimate’ and ‘hlcm\_luz\_simulate’ functions in /sandag\_urbansim/models.py, and the ‘elcm\_estimate’ and ‘elcm\_simulate’ functions in /urbansim\_defaults/models.py. Note that the hlcm\_luz\_simulate function contains code to loop over all LUZs and simulate location choices separately for each.

To re-estimate the location choice models, open the following IPython notebook: /sandag\_urbansim/notebooks/Estimation.ipynb. Execute the top two cells to import dependencies and instantiate the accessibility engine. Then, for HLCM re-estimation, navigate down to the section with the ‘HLCM Estimation’ header. Execute the first cell in this section, and HLCM estimation will run, with estimation results printed out. Note in this cell that ‘rsh\_simulate’ is run just prior to the HLCM estimation- this is to get residential prices in place so price can be used as an explanatory variable in the location choice model. For ELCM re-estimation, navigate to the notebook section with header ‘ELCM Estimation’, and execute the cell that runs “elcm\_estimate”.

To change the set of variables in the location choice model specifications, update the model\_expression arguments in /configs/hlcm\_luz.yaml and /configs/elcm.yaml. As with the real estate price model hedonic regressions described in the previous section, the model expressions in the location choice model YAML files should be written in patsy syntax. The ‘segmentation\_col’ argument specifies the agent column used to segment the model (separate coefficients estimated for each segment of observations). The ‘choosers\_fit\_filters’ argument specifies the filters on observations in estimation (what observations to exclude). The choosers\_predict\_filters specifies filters on households in simulation. The alts\_...\_filters specify filters on location alternatives. The ‘sample\_size’ argument specifies the number of location alternatives to sample during estimation. The ‘estimation\_sample\_size’ argument specifies the maximum number of observations to use in estimation (if number of observations in a segment exceeds this value, observations are sampled- this is to prevent excessive memory usage).

The source of the current household table used in the model is a .csv file representing formatted PopSyn2 output provided by SANDAG (see /population/household.csv). To swap in a new base-year synthetic population, replace this .csv file and re-run the data regeneration scripts, which will load the new csv and allocate household records from MGRA to building. Beyond the existing set of household columns, additional columns from PopSyn2 can be added to household.csv without problem if additional demographic detail is needed.

The source of the current jobs table is from LEHD (see /employment/jobs\_lehd\_raw.csv). This block-level employment data source is allocated to the building level in /scripts/process.py. To swap in SANDAG’s own employment data, simply replace the jobs table in the sandag.h5 HDF5 file, keeping the same format (one row per job, sector\_id and building\_id populated). If the sector typology is different in the new employment data, any variables in /sandag\_urbansim/variables.py and /configs/neighborhood\_vars.yaml that refer to employment sectors should be updated to refer to the appropriate new sector\_id.

The location choice model places jobs into ‘job\_spaces’. The location choice model will not allocate more jobs to buildings than there are job spaces. Just as residential units are a capacity constraint for the HLCM, job spaces are a capacity constraint for the ELCM. Job spaces are calculated by dividing the square footage in each building by the appropriate square-footage-per-job value. Square-foot-per-job by LUZ and development type is contained in the building\_sqft\_per\_job table. This is a processed version of /employment/ sqftPerEmpByDevType.csv. As an example, for a building in LUZ 1 and development\_type\_id 2, the appropriate square-footage-per-job value is looked up in this table, and this value is then used to calculate the number of job spaces in the building. The code that performs this calculation lives in the ‘job\_spaces’ function in /sandag\_urbansim/variables.py.

The control totals table can be altered in the future if the design of household allocation changes. For example, by removing the luz\_id column, the table could be used for region-to-building allocation (skipping the PECAS-controlled LUZ intermediate geography altogether). Or if more detailed household types are to be used, additional activity\_id’s can be introduced into the table (just make sure to also edit the activity\_id household variable definition in /sandag\_urbansim/variables.py). See the ‘household\_transition’ function in /sandag\_urbansim/models.py for the transition model code.

While the household-side of demand in the model is transitioned using a control table, the employment side is currently transitioned by applying a simple growth rate. This is because an LUZ employment control table was not provided. See the jobs\_transition function in /sandag\_urbansim/models.py: the number of jobs increases by 5% each year (adjust the .05 proportion to change this growth rate). New jobs are introduced based on this simple growth rate, then allocated from the region to buildings by the ELCM. Add code to the ‘jobs\_transition’ function to transition the employment side in a different way (i.e. using a transition model that utilizes an annual\_employment\_control\_totals table).

See Table 2 for a mock example of the annual\_household\_control\_totals table, for illustrative purposes only. See Table 3 for a mock example of the annual\_employment\_control\_totals table, once employment controls are ready.

Table 2- LUZ household controls from PECAS, the annual\_household\_control\_totals table

|  |  |  |  |
| --- | --- | --- | --- |
| year | base\_luz | activity\_id | total\_number\_of\_households |
| 2013 | 1 | 1 | 600 |
| 2013 | 1 | 2 | 650 |
| 2013 | 1 | 3 | 700 |
| 2013 | 1 | 4 | 750 |
| 2013 | 1 | 5 | 800 |
| 2013 | 2 | 1 | 400 |
| 2013 | 2 | 2 | 500 |
| 2013 | 2 | 3 | 600 |
| 2013 | 2 | 4 | 700 |
| 2013 | 2 | 5 | 800 |
| 2013 | 3 | 1 | 100 |
| 2013 | 3 | 2 | 200 |
| 2013 | 3 | 3 | 300 |
| 2013 | 3 | 4 | 400 |
| 2013 | 3 | 5 | 500 |

Table 3- LUZ employment controls from PECAS, the annual\_employment\_control\_totals table

|  |  |  |  |
| --- | --- | --- | --- |
| year | base\_luz | sector\_id | total\_number\_of\_jobs |
| 2013 | 1 | 1 | 1200 |
| 2013 | 1 | 2 | 1300 |
| 2013 | 1 | 3 | 1400 |
| 2013 | 1 | 4 | 1500 |
| 2013 | 1 | 5 | 1600 |
| 2013 | 2 | 1 | 800 |
| 2013 | 2 | 2 | 1000 |
| 2013 | 2 | 3 | 1200 |
| 2013 | 2 | 4 | 1400 |
| 2013 | 2 | 5 | 1600 |
| 2013 | 3 | 1 | 200 |
| 2013 | 3 | 2 | 400 |
| 2013 | 3 | 3 | 600 |
| 2013 | 3 | 4 | 800 |
| 2013 | 3 | 5 | 1000 |

After the HLCM has run and placed households into buildings in each simulation year, MGRA level output indicators are exported to PASEF (SANDAG’s small-area demographic model). Currently, MGRA level household totals by activity\_id are being exported to the ‘pasef’ directory in /sandag\_urbansim/data. The exported .csv files have the following naming scheme: mgra\_hh\_Y where Y is the simulation year for which the household indicators were exported. Each .csv has 3 columns: mgra\_id, activity\_id, number\_of\_households. The code that generates these export indicators can be found in /sandag\_urbansim/models.py: see the ‘model\_integration\_indicators’ function and the code comment for ‘Households by MGRA to PASEF’.

### LINK WITH PECAS

The UrbanSim location choice models allocate increases in demand-side agents (i.e. households and jobs) from the LUZ level to the building/parcel level. Region-to-LUZ allocation is carried out by PECAS. The link between PECAS and UrbanSim is an LUZ-level control totals table. See the following file for the household controls: /sandag\_urbansim/data/pecas\_hh\_controls.csv. This table contains PECAS predictions and gives the total number of households by year, LUZ, and household type. This .csv file is read by UrbanSim near the beginning of each simulation and formatted as an UrbanSim control table named ‘annual\_household\_control\_totals’. See the function with this name in datasources.py. This function, which runs the first time the annual\_household\_control\_totals table is needed, reads the csv, changes column names, removes unneeded columns, and integerizes the household total column. The resulting table is then used in each simulation year by the household transition model to evolve the base-year synthetic population to match prescribed LUZ-level controls. New households are introduced with a building\_id of -1. The location choice model then places these new households into buildings, and replaces the -1 building\_id with an actual building\_id. To run the model with a new static set of PECAS controls, simply swap out pecas\_hh\_controls.csv for an updated version.

When integrated runs with PECAS are ready to be tested, UrbanSim’s annual\_household\_control\_totals table should be updated in simulation based on the latest PECAS values. Each simulation year’s latest LUZ control values should be read from a file or database as a DataFrame (e.g. new\_pecas\_hh\_controls), and then used to update the UrbanSim table: sim.add\_table(“annual\_household\_control\_totals”, new\_pecas\_hh\_controls).

### SPECIFICATION

Since the location choice models handle LUZ to building allocation, simple specifications will serve well (with a focus on local/building characteristics rather than regional patterns). The HLCM is stratified by activity\_id, reflecting the hypothesis that households in each of the six household types contained in the current activity\_id typology have different location behaviors. Separate coefficients are estimated for each household activity\_id. The current set of explanatory variables in each segment are: sqft\_per\_unit, year\_built, residential\_units, jobs\_400m, and res\_price\_per\_sqft. Most of these are building-level characteristics, and the only exception (jobs\_400m) is highly local. While this HLCM specification is provisional and could be revised, it improves upon random allocation.

Similarly, the ELCM is stratified by employment sector\_id, and separate coefficients are estimated for each sector. Each sector is hypothesized to have different locational tendencies. The 20-sector NAICS-based typology is currently that used by LEHD, which is the source of the jobs table. Explanatory variables include: number of jobs of the same sector within 1200 meters (to capture sectoral clustering), jobs\_800m, ave\_parcel\_size, is\_office, and is\_retail.

See Appendix 2 for the estimated location choice model coefficients. These represent limited specifications, but the location choice models here play a more limited role than they do in other UrbanSim model systems (region to LUZ allocation is taken care of by PECAS).

# Section 3: Proforma Real Estate Developer Model

The Real Estate Developer Model simulates the location, type and density of real estate development at the level of specific parcels. Proforma-based profitability calculations for every parcel in the region are run each simulation year. The calculations account for variables such as price, costs by structure type, fees, and zoning. See Memorandum 2 in the appendix for additional overview-level information on the proforma model. A focus in the model design was meeting SANDAG’s operational needs.

This section describes calibration of the real estate supply predictions. The period 2013 to 2015 was repeatedly simulated, with adjustments made after each iteration to move the simulation in the direction of MSA calibration targets prepared from 2004 – 2010 observed data. To aid in calibration, this period was simulated without the scheduled\_development\_model (known pipeline projects), so that UrbanSim was responsible for all supply predictions.

The 2013-2015 period was iteratively simulated (the model base-year is 2012). After each iteration, the price shifter for each MSA was updated in the direction that would move the simulation towards the targets. For example, if an MSA proportion-of-growth target is undershot, the price shifter for that county is incremented upwards so that new real estate development becomes more attractive. If an MSA proportion-of-growth target is overshot, the dummy coefficient for that county is incremented downwards. If the simulated growth proportion for an MSA is within a narrow range of the target growth proportion for residential units or non-residential square-footage, the price shifter for that MSA is held constant. This process is carried out for each MSA and residential/non-residential separately after every calibration iteration.

It’s important to note that although the model was calibrated over a few simulation years, calibration does not pre-determine model outcomes: typical simulations will be run well beyond the calibration period to 2040 and beyond, and UrbanSim accounts for a wide variety of variables that will influence growth separate from the calibration process. For example, MSAs that have historically grown very rapidly can run out of zoned capacity halfway through the simulation, shifting growth to MSAs that historically experienced slower growth. Changing congestion effects and price effects can influence the spatial distribution of growth in the simulation. These are examples of complex feedbacks that UrbanSim is designed to represent.

Note that prior to calibration, the first step in evaluating proforma results was an eye-ball test. Patterns of LUZ-level new built space construction were reviewed and shared with SANDAG during weekly meetings. The development pattern mirrored the price predictions closely. An unrealistic amount of residential growth was simulated in certain localized areas with high prices, available zoned capacity, and strong NIMBY forces. This motivated the need to calibrate to observed patterns of growth and capture factors unaccounted for in the proforma’s rule-based profitability calculations.

In much of the discussion in the reports so far, we have discussed how other components of the model system (e.g. price models, location choice models) influence the supply predictions. But influence goes in the other direction as well. The location of new supply sets the upper bound on the number of new demand agents in each location, and the location and characteristics of new supply influences the probabilities of agents locating in certain areas. New supply also affects vacancy rates and density levels, which then influence prices. The model system consists of interconnected components, and influence is mutual and can be complex.

### CODE AND OPERATION

The model code for the real estate supply model lives mainly in four functions in /sandag\_urbansim/models.py: ‘feasibility’, ‘residential\_developer’, ‘non\_residential\_developer’, and ‘scheduled\_development\_events’. The ‘feasibility’ model calculates profitability of development on each parcel, and the ‘\_developer’ models pick development projects to match the DEFM target. The ‘scheduled\_development\_events’ model inserts building records based on known development projects in the pipeline (Sitespec).

For calibrating the proforma model, the first-stage calibration geography selected was MSA. There are 7 MSA’s within the SANDAG region. Built space calibration targets were prepared based on the proportion of growth captured by each MSA in the 2004 – 2010 period. The residential supply predictions were calibrated to the proportion residential unit growth captured by each MSA between 2004 - 2010, and non-residential supply predictions were calibrated to the proportion of non-residential square-footage growth captured by each MSA in the same time period. A calibration IPython notebook was prepared to iteratively run the model system, and after each iteration adjust MSA-level price shifters such that the simulation moved in the direction of the calibration targets. See Calibration.ipynb located in /sandag\_urbansim/notebooks. Calibration settings are configured in the second cell. Calibration settings include: MSA proportion growth targets by residential/non-residential, number of calibration iterations, step-size (how much to increment the price shifters after each iteration, if necessary), and the threshold (how many percentage points [expressed in proportion terms] difference is acceptable between target/simulated). The calibration targets are easy to change. After configuring these settings, run all three cells in the notebook to start a calibration run. Each calibration iteration involves running /sandag\_urbansim/calib\_simulation.py. To change the scenario being iteratively run during calibration, edit calib\_simulation.py. Here you can change the number of years to simulate and the set of models to run in each iteration.

Existing calibrated MSA price shifters are as shown in Table 1, in Section 1 of this document on prices.

After MSA level calibration, LUZ results were again reviewed. Certain high-priced LUZs (La Jolla, Mission Beach) with available zoned capacity were still receiving too much growth. A second round of calibration, targeting these specific LUZs, was undertaken. See the Calibration\_LUZ.ipynb IPython notebook located in /sandag\_urbansim/notebooks. This notebook can be used as a template for future LUZ-level calibration. The calibration settings in the second cell of this notebook are the same as in the MSA calibration notebook, with the exception that now the calibration target pertains only to proportion dwelling unit growth that went to the target LUZs (LUZ 69, 70, and 72). Run all cells to initiate the LUZ calibration. The scenario being iteratively run in this case is that specified in /sandag\_urbansim/calib\_luz\_simulation.py. This calibration scenario (number of years, models to run) can be tweaked as necessary.

Note that, whether running MSA calibration or the targeted LUZ calibration, after each iteration a .csv file is outputted to /sandag\_urbansim/calibration. The latest price shifters (msa\_shifters.csv, luz\_du\_shifter.csv) are stored in this directory. Also, the latest simulated proportions are stored (msa\_du\_simulated.csv, msa\_nrsf\_simulated.csv, luz\_dusimulated.csv). After each calibration iteration, these files get overwritten.

SANDAG requested a way to assert dwelling unit target values in certain zones, if sufficient profitable projects exist in the zone to meet the unit target. These LUZ-level manual overrides for dwelling units are specified via the ‘luz\_overrides.csv’ file located in /sandag\_urbansim/data/overrides. See Table 4 for an example of luz\_overrides.csv.

Table 4: Example of luz\_overrides.csv. Targets expressed in terms of total dwelling units.

|  |  |  |  |
| --- | --- | --- | --- |
| year | luz\_id | development\_type\_id | target |
| 2012 | 70 | 21 | 14000 |
| 2012 | 113 | 19 | 3000 |
| 2012 | 70 | 20 | 4000 |

For each combination of year, LUZ and residential development type, an override target can be specified. If the target exceeds the existing number of dwelling units in the LUZ and development type, the proforma developer model will try to meet the target. The target will be met as long as there are sufficient profitable projects and zoned capacity in the LUZ. Controlled LUZs (those with an override specified) are treated separately from uncontrolled LUZs. Controlled LUZs are simulated first, and any remaining unmet residential unit target value is transferred over to uncontrolled LUZs so that regional DEFM dwelling unit targets are met each year. LUZ overrides are useful for asserting residential unit values to either prevent excessive development or spur additional development. Note that if the target is less than the existing number of units in the LUZ, no change will occur as the LUZ override mechanism will not demolish buildings.

The code for LUZ dwelling unit overrides lives in /sandag/sandag\_urbansim/models.py. See lines 509 to 553 in the ‘residential\_developer’ function. The luz\_overrides.csv file is read, records associated with the current simulation year are kept, and then if overrides exist, the developer model tries to match the specified target in those LUZs.

The scheduled development events model (Sitespec) inserts known development projects into the simulation in the appropriate year. See the ‘scheduled\_development\_events’ function in /sandag\_urbansim/models.py for the model code. The list of scheduled development events is contained in /sandag\_urbansim/data/scheduled\_development\_events.csv. New building records are inserted based on the records in this .csv file. The ‘year\_built’ column specifies the simulation year when the buildings will be inserted. To update scheduled development events, either update the scheduled\_development\_events.csv file directly, or alternatively update site\_spec.shp in the ‘scheduled’ directory of the raw data input folder and then run the data regeneration scripts. The data regeneration scripts (load.py, process.py, and export\_to\_h5.py) load site\_spec.shp, tag Sitespec records with a parcel\_id, process it into the needed format (the scheduled\_development\_events table), and then write the formatted table as a .csv to the /sandag\_urbansim/data directory. The simulation then reads the outputted .csv when running the scheduled\_development\_events model. If Urban Canvas is used in the future to maintain development projects, then UrbanSim can be configured to read scheduled\_development\_event records from the Urban Canvas database.

Matching built space targets from DEFM is a priority in the model system. Code was added to the proforma model to address a corner case: when not enough profitable projects exist in the region to match the regional DEFM target. If this situation is encountered, prices are scaled up by a single factor regionally (preserving subregional price variation) such that enough profitable projects exist to meet the regional target (subject to zoning). The code for this lives in the ‘iter\_feasibility’ function at line 214 of /sandag\_urbansim/models.py. Given prices in the SANDAG region, lack of profitable projects is not realistic, so the code here ensures that simulated prices can always support matching the DEFM targets. Subregional variation in prices as predicted by the real estate price model is preserved.

Note that parcel filters can be applied in the proforma to filter out parcels that should not receive development. This is an optional feature. Currently, parcel development capacity is determined by zoning, allowed uses, and proportion\_undevelopable. If parcel filters are desired, uncomment lines 282 and 307 in /sandag\_urbansim/models.py, and then comment out the line immediately after each. Then populate the ‘parcel\_filter’ argument in /sandag\_urbansim/configs/settings.yaml (in the ‘feasibility’ section of the yaml file). Only parcels that pass the filters are run through the profitability calculations, so these optional filters can be used to filter out parcels from development consideration. But the same effect can be achieved via zoning and proportion\_undevelopable.

Zoning is a key input to the proforma model of real estate supply. In /sandag\_urbansim/datasources.py see the ‘zoning’ and ‘zoning\_allowed\_uses’ functions, which represent the zoning tables used by the simulation. These tables are used to calculate parcel-level variables such as ‘max\_dua’ and ‘max\_far’ (see the functions with these names in /sandag\_urbansim/variables.py), which are key input variables to the proforma. Also note the ‘max\_height’ variable, which is currently set at a very high value so that it does not represent a constraint. This is because we only observe ‘max\_dua’ and ‘max\_far’ in the zoning. If ‘max\_height’ becomes available in a future zoning table we can utilize this information in the ‘max\_height’ function. Also see the ‘parcel\_is\_allowed’ function in variables.py, which uses the zoning\_allowed\_uses table to indicate which parcels a given development\_type\_id is allowed on. Finally, see the ‘proportion\_developable’ variable, which is used to scale allowable densities to account for environmental constraints such as steep slopes and flood plains. Together these variables represent the development constraints respected by the proforma model.

To update the ‘zoning’ and ‘zoning\_allowed\_use’ tables, swap out the following files and then re-run the data scripts in /sandag\_urbansim/scripts: 'zoning/zoning.csv', 'zoning/zoning\_allowed\_uses.csv'. Ensure that the zoning\_id on parcels is correct if updating zoning.

Parcel-level fees are another input to the proforma calculations. To update the fees used in the model, swap out the following files and then re-run the data scripts in /sandag\_urbansim/scripts: 'proformaInputs/fees/fee\_schedule.csv', 'proformaInputs/fees/parcel\_fee\_schedule.csv'.

DEFM regional built space controls determine the amount of new space to construct in the simulation each simulation year. These controls are stored in the following .csv files: /sandag\_urbansim/data/ defm\_res\_unit\_controls.csv, /sandag\_urbansim/data/non\_res\_space\_control.csv. Swap out these .csv files if updated DEFM regional space controls are generated. These files are read and processed by the ‘residential\_space\_targets’ and ‘non\_residential\_space\_targets’ functions in /sandag\_urbansim/models.py. These functions are called by the developer model each year to obtain the appropriate targets.

Price predictions from the hedonics are a key input to the proforma model of real estate development. Real estate price model output should be evaluated closely. As described in the section on the real estate price model, prices are summarized for input to the proforma according to the network accessibility variable definitions found in: /sandag\_urbansim/configs/price\_vars.yaml

Buildings and parcels are the core tables operated on by the proforma model. New buildings are predicted (rows inserted into the buildings table) and each building is assigned a location (each building has a parcel\_id and must fit within the development constraints of that parcel). To update the building and parcel tables in the model, swap out the existing building and parcel shapefiles in the input data directories and re-run the data scripts in /sandag\_urbansim/scripts. The building and parcel shapefiles are located at: 'space/parcel.shp', 'space/building.shp'. See Memorandum 1 in the appendix for the building and parcel table schema, and a description of the role of each table column.

After running a simulation involving the real estate supply models, the user will typically generate indicators to summarize predictions. In the Simulation notebook, examples of this are provided as a template for future indicators. The second code cell of the notebook runs a simulation from year 2013 to 2040. After this simulation has completed, the third code cell can be executed, and this creates a DataFrame of LUZ indicators, including number of new residential units constructed in the forecast period, and amount of new non-residential square-footage constructed in the forecast period. Other indicators can be similarly arrived at using the pattern shown here. After a simulation has run, get the buildings table by executing ‘sim.get\_table('buildings').to\_frame()’ which returns a DataFrame of the buildings in the forecast year (e.g. all buildings in year 2040). Then this DataFrame can be sliced, diced, and summarized to arrive at the desired indicators. New building-level variables can be added to variables.py. Other simulated tables can be similarly accessed using the sim.get\_table() function.

### LINK WITH PECAS

After the proforma model has made real estate supply predictions, new built space totals are summarized at the LUZ level and passed to PECAS. This information is needed so that PECAS does not over-allocate demand agents to LUZs (e.g. number of households should not exceed number of dwelling units). For this purpose, at the end of each simulation year UrbanSim exports a .csv of LUZ built space totals by development type. The .csv file is outputted to the following directory: /sandag\_urbansim/data/pecas\_urbansim\_exchange. The ‘luz\_du\_Y.csv’ (where Y is the simulation year) gives, the number of residential units by LUZ and development type. This ‘luz\_nrsf\_Y.csv’ (where Y is the simulation year) gives, the amount of non-residential square-footage by LUZ and development type. The code that generates these output indicators can be found in the ‘model\_integration’ function in /sandag\_urbansim/models.py, just below the “Space by LUZ to PECAS” code comment. This code can be tweaked if the input file format needed by PECAS (i.e. floorspaceI.csv) is different.

### VALIDATION

After calibration of the developer model, the model system was used to simulate the 2012 – 2015 period and MSA comparisons were made between simulated growth proportions and observed post-2010 growth proportions. Recall that the model was calibrated to real estate supply growth between 2004 and 2010. Post 2010 data was saved for an initial validation exercise.

Residential validation data is from SANDAG residential unit estimates, provided for the 2010 to 2014 period. The MGRA level estimates file (‘estimates\_residential\_units\_households.csv’) contains residential unit estimates by year, and this file was processed to arrive at residential unit growth by MSA. This growth was then expressed in terms of proportion residential unit growth captured by each MSA in the 2010 to 2014 period. See Table 5, which compares simulated and observed residential growth proportions. The column on the left contains proportion residential unit growth captured by each MSA in the 2010 to 2014 period. The column on the right shows simulated residential growth proportions between 2012 (the model base-year) and 2015. There is a reasonable similarity between the two columns (correlation coefficient of .92).

Non-residential square-footage data for the 2010 to 2012 time period was obtained from the buildings table, and proportion of growth in non-residential square footage captured by each MSA between 2010 and 2012 was used for validation. See Table 6, which compares simulated and observed non-residential growth proportions. The column on the left contains proportion non-residential square-footage growth captured by each MSA in the 2010 to 2012 period. The column on the right shows simulated non-residential growth proportions between 2012 (the model base-year) and 2015. There is a positive correlation between the two columns (correlation coefficient of .37), though the similarity in variation across MSAs is not as high as on the residential side.

The calibration/validation approach taken with the Phase 1 model divided up recent observed data into a calibration set (2004 – 2010) and a validation set (2010 and beyond). The model was calibrated to growth proportions by MSA from the former, and validated to growth proportions by MSA from the latter. Because the base-year is 2012, and a relatively short time-series of observed data is available beyond that, true longitudinal validation is challenging. As additional observed data accumulates, further validation can be undertaken. Sensitivity testing may be a valuable validation exercise as well- validating the model with respect to expected policy and behavioral sensitivities. Reasonable use of the model system should be determined by a combination of validation, sensitivity testing, further model refinements, experience from use, and common sense. The process of model testing and adjustment helps to ensure reasonableness of simulation output and sensitivity to policies of interest, as well as providing guidance as to how to most appropriately use the model.

Table 5: Residential supply validation- MSA

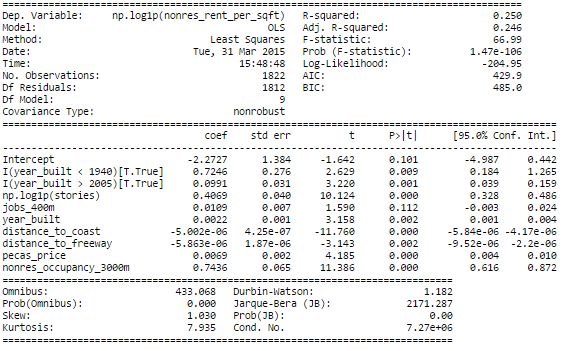
|  |  |  |
| --- | --- | --- |
| msa\_id | proportion\_du\_growth\_observed | proportion\_du\_growth\_simulated |
| 0 | 0.135098 | 0.228233 |
| 1 | 0.315154 | 0.308403 |
| 2 | 0.187753 | 0.153677 |
| 3 | 0.058635 | 0.024566 |
| 4 | 0.153369 | 0.148905 |
| 5 | 0.143292 | 0.136216 |
| 6 | 0.0067 | 0 |

Table 6: Non-residential supply validation- MSA

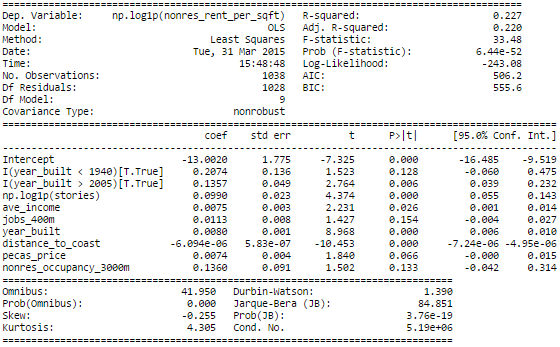
|  |  |  |
| --- | --- | --- |
| msa\_id | proportion\_ nrsf\_growth\_observed | proportion\_nrsf\_growth\_simulated |
| 0 | 0.343273 | 0.134149 |
| 1 | 0.115041 | 0.285717 |
| 2 | 0.030555 | 0.197081 |
| 3 | 0.020157 | 0.010204 |
| 4 | 0.19953 | 0.207715 |
| 5 | 0.291049 | 0.165135 |
| 6 | 0.000394 | 0 |

# Appendix 1: Real Estate Price Model Coefficients

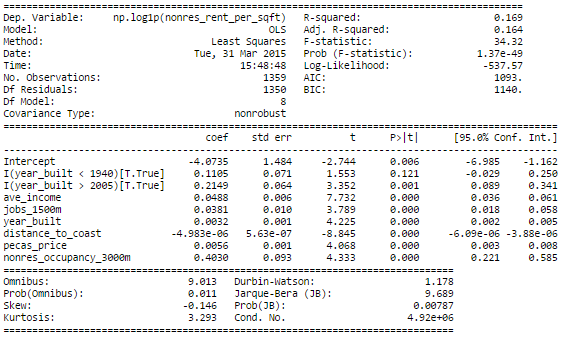
**Price Model: OLS Regression Results for Development Type ID 2**



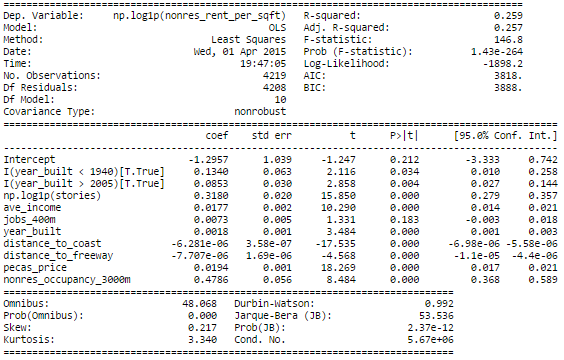
**Price Model: OLS Regression Results for Development Type ID 4**



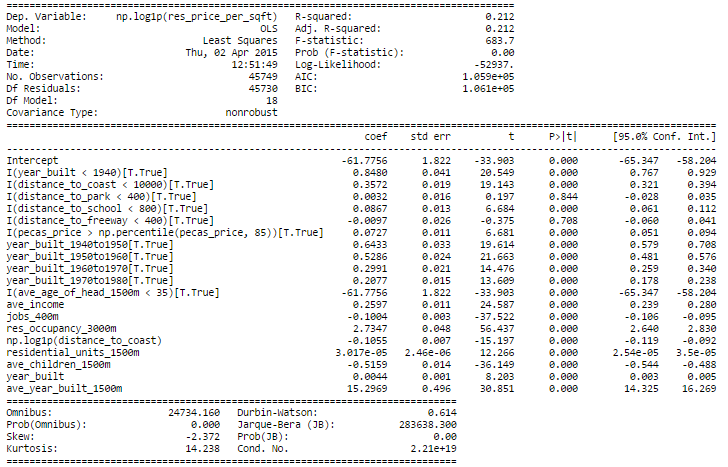
**Price Model: OLS Regression Results for Development Type ID 5**



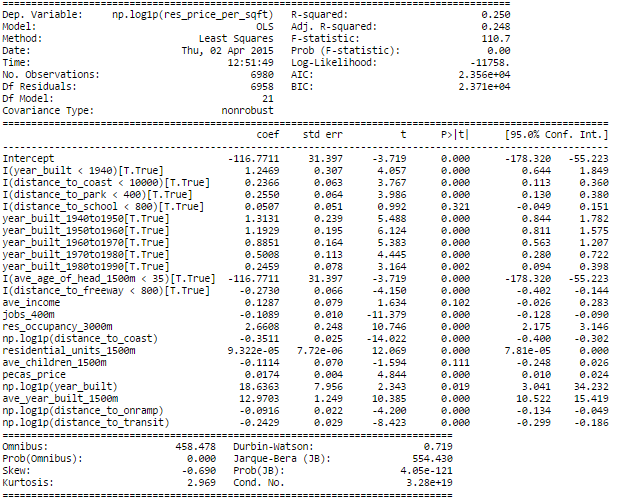
**Price Model: OLS Regression Results for Development Type Other-Non-residential**



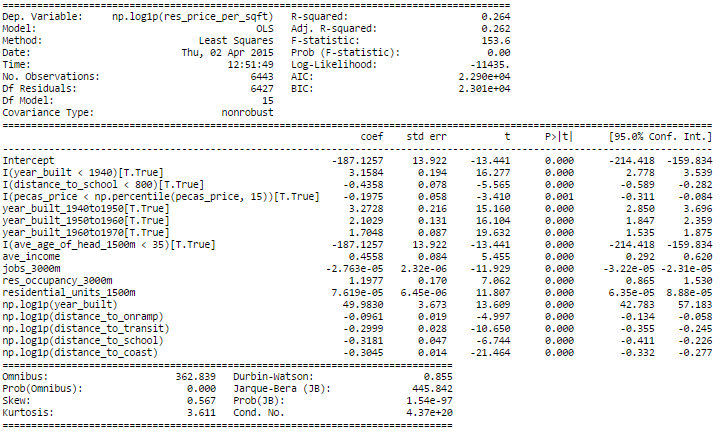
**Price Model: OLS Regression Results for Development Type ID 19**



**Price Model: OLS Regression Results for Development Type ID 20**

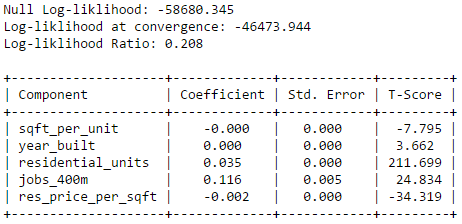


**Price Model: OLS Regression Results for Development Type ID 21**

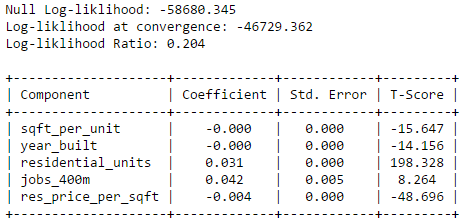


# Appendix 2: Location Choice Model Coefficients

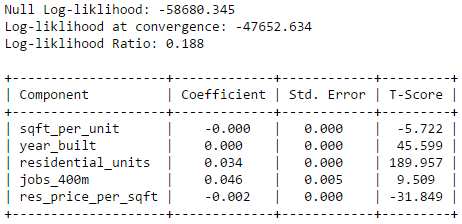
**Household Location Choice Model: Estimated Coefficients for Activity ID 38**



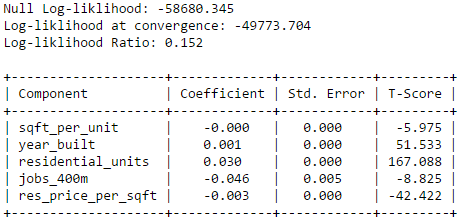
**Household Location Choice Model: Estimated Coefficients for Activity ID 39**



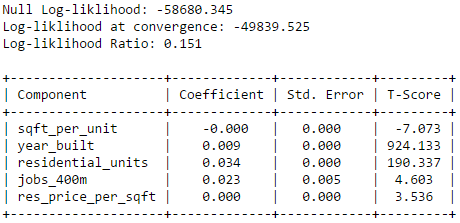
**Household Location Choice Mode: Estimated Coefficients for Activity ID 40**



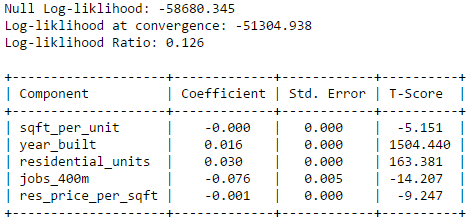
**Household Location Choice Model: Estimated Coefficients for Activity ID 41**



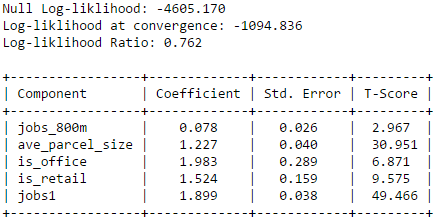
**Household Location Choice Model: Estimated Coefficients for Activity ID 42**



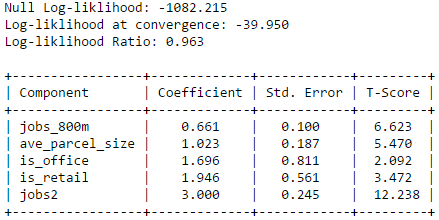
**Household Location Choice Model: Estimated Coefficients for Activity ID 43**



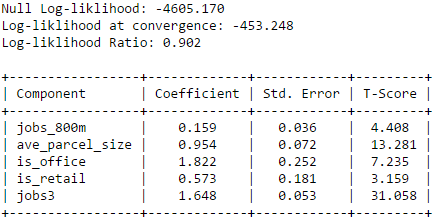
**Employment Location Choice Model: Estimated Coefficients for Sector ID 1**



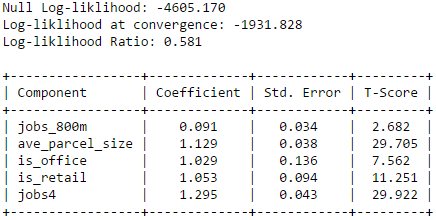
**Employment Location Choice Model: Estimated Coefficients for Sector ID 2**



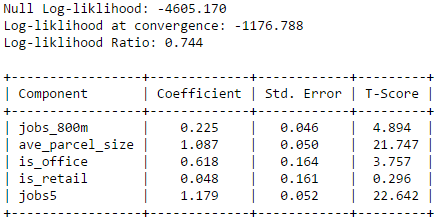
**Employment Location Choice Model: Estimated Coefficients for Sector ID 3**



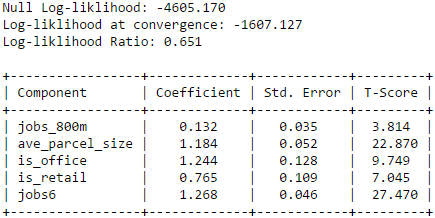
**Employment Location Choice Model: Estimated Coefficients for Sector ID 4**



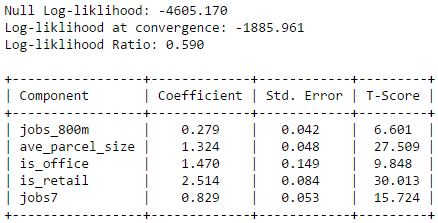
**Employment Location Choice Model: Estimated Coefficients for Sector ID 5**



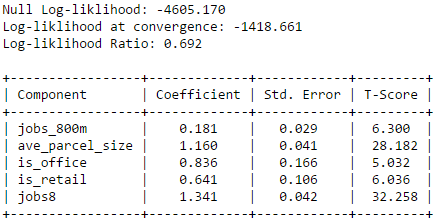
**Employment Location Choice Model: Estimated Coefficients for Sector ID 6**



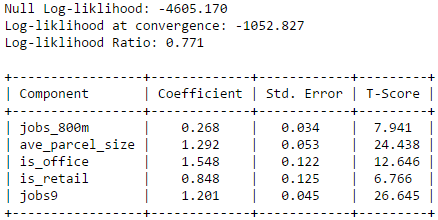
**Employment Location Choice Model: Estimated Coefficients for Sector ID 7**



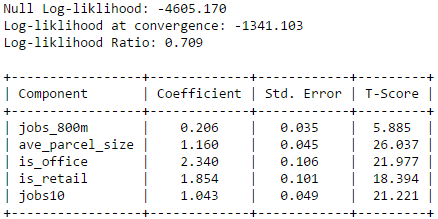
**Employment Location Choice Model: Estimated Coefficients for Sector ID 8**



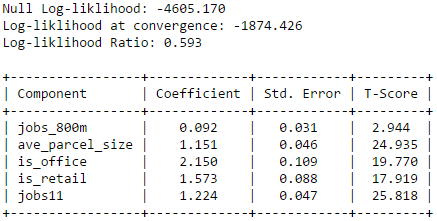
**Employment Location Choice Model: Estimated Coefficients for Sector ID 9**



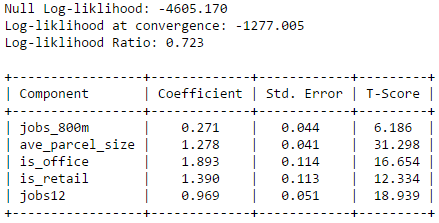
**Employment Location Choice Model: Estimated Coefficients for Sector ID 10**



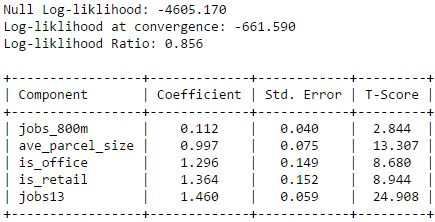
**Employment Location Choice Model: Estimated Coefficients for Sector ID 11**



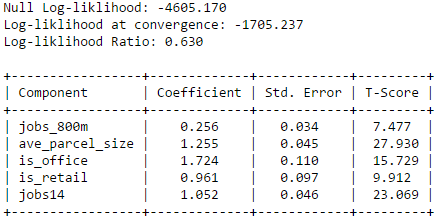
**Employment Location Choice Model: Estimated Coefficients for Sector ID 12**



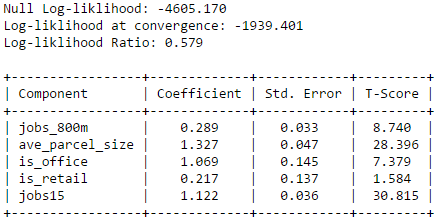
**Employment Location Choice Model: Estimated Coefficients for Sector ID 13**



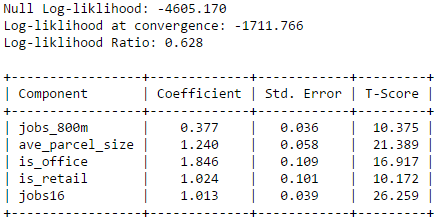
**Employment Location Choice Model: Estimated Coefficients for Sector ID 14**



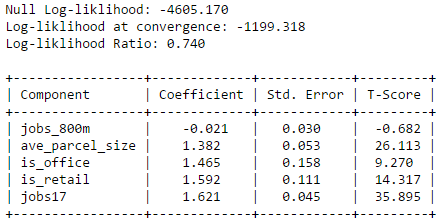
**Employment Location Choice Model: Estimated Coefficients for Sector ID 15**



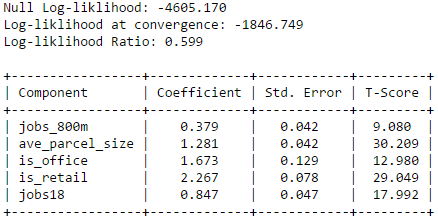
**Employment Location Choice Model: Estimated Coefficients for Sector ID 16**



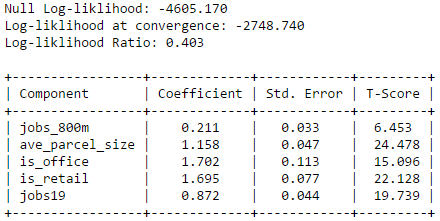
**Employment Location Choice Model: Estimated Coefficients for Sector ID 17**



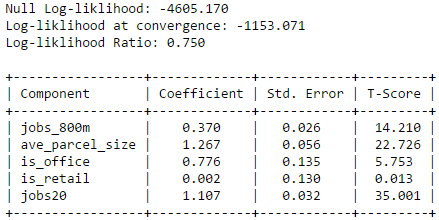
**Employment Location Choice Model: Estimated Coefficients for Sector ID 18**



**Employment Location Choice Model: Estimated Coefficients for Sector ID 19**



**Employment Location Choice Model: Estimated Coefficients for Sector ID 20**



# Appendix 3: Installation Instructions for the sandag\_urbansim Model

This document lists the Windows installation steps for the proof-of-concept Subregional Allocation Model. The code for this project lives on github at <https://github.com/synthicity/sandag_urbansim>. The input data can be downloaded from the SANDAG OneDrive directory.

For a more frequently updated version of these instructions, please see the following sandag\_urbansim GitHub wiki page:

https://github.com/synthicity/sandag\_urbansim/wiki/Windows-installation-instructions

**Suggested computer requirements**

The model runs best on a computer with a 64-bit operating system and at least 16GB of RAM.

**Steps**

1. Download and install Git
   1. <https://git-scm.com/downloads>
2. Download and install 64-bit Python 2.7 Anaconda installer
   1. <https://3230d63b5fc54e62148e-c95ac804525aac4b6dba79b00b39d1d3.ssl.cf1.rackcdn.com/Anaconda-2.2.0-Windows-x86_64.exe>
   2. Other versions available at: https://store.continuum.io/cshop/anaconda/
3. Download and install Postgres
   1. <http://get.enterprisedb.com/postgresql/postgresql-9.4.2-1-windows-x64.exe>
   2. After installation, use postgres' "Application Stack Builder" to install PostGIS
   3. Add C:\Program Files\PostgreSQL\9.4\bin to your PATH environmental variable if not already there (edit this if different postgres version used)
4. Install additional Python dependencies
   1. From command prompt type:
      1. conda install gdal
      2. conda install sqlalchemy=0.8.0
      3. pip install pandana GeoAlchemy2
   2. Download wheel for psycopg2
      1. Browse to <http://www.lfd.uci.edu/~gohlke/pythonlibs/#psycopg>
      2. Download a wheel for 64-bit python 2.7 (e.g. psycopg2-2.5.5-cp27-none-win\_amd64.whl)
      3. From command prompt type, navigate to directory containing downloaded wheel, then type “pip install psycopg2-2.5.5-cp27-none-win\_amd64.whl” to install the downloaded python wheel
5. Install UrbanSim. From command prompt type:
   1. git clone https://github.com/synthicity/urbansim -b sandag
   2. cd urbansim
   3. python setup.py develop
6. Install urbansim\_defaults. From command prompt type:
   1. git clone <https://github.com/synthicity/urbansim_defaults>
   2. cd urbansim\_defaults
   3. git reset --hard 993ac4d20f3c2b7b2b8b38844045a36b1adb1758
   4. python setup.py develop
7. Install spandex. From command prompt type:
   1. git clone <https://github.com/synthicity/spandex>
   2. cd spandex
   3. python setup.py develop
8. Get code for the sandag\_urbansim project
   1. git clone <https://github.com/synthicity/sandag_urbansim>
9. In your home directory (e.g. on windows C:/users/username), create a ".spandex" directory and in this directory create a text file named "user.cfg".
   1. Populate text file with postgres database settings.
   2. E.g.:

[database]

database = sandag

user = postgres

password = postgres

host = localhost

port = 5432

[data]

directory = C:\\sandag\\data

srid = 2230

1. If the postgres database specified in the config file doesn't already exist, create it (e.g. use PgAdmin).
2. If the input data directory specified in the config file doesn't already exist, create it and populate with the contents of OneDrive.
   1. In this data directory, also create an 'out' folder- this is where the data regeneration script output (e.g. the hdf5 file) will go.
3. If not using an existing sandag.h5 HDF5 file, generate it
   1. python /sandag\_urbansim/scripts/run.py
   2. If successfully runs, you will see a sandag.h5 exported to the ‘out’ folder within your data directory (i.e. the OneDrive directory that your spandex configuration file points to)
4. Run a simulation out to 2040
   1. In the /sandag\_urbansim/data directory, in addition to the hdf5 file, you will also need the files/folders contained in:
      1. <https://dl.dropboxusercontent.com/>u /69619688/for\_data\_dir\_sandag\_05282015.zip
      2. Unzip the contents of the above zip file and place in /sandag\_urbansim/data alongside sandag.h5 hdf5 file.
   2. Run the Simulation notebook
      1. cd /sandag\_urbansim/notebooks
      2. ipython notebook
      3. Open Simulation notebook, execute desired cells
5. Run calibration
   1. Ensure /sandag\_urbansim/data directory is populated per the previous step
   2. Run the Calibration notebook
      1. cd /sandag\_urbansim/notebooks
      2. ipython notebook
      3. Open Calibration notebook, edit settings as desired, execute cells
6. If planning to export data to the Urban Canvas database from this machine, configure pgpass.conf (credentials for exporting to the database)
   1. Pgpass.conf is typically located in: C:\Users\username\AppData\Roaming\postgresql
   2. Open in text editor, and add a line that says:
      1. urbancanvas.cp2xwchuariu.us-west-2.rds.amazonaws.com:\*:sandag:parcel22building

# Appendix 4: Memorandum 1

**SANDAG Subregional Allocation Model**

**Assessment of SANDAG Data Sets**

**Technical Memorandum 1**

This memorandum inventories SANDAG’s existing data sets, and describes needed data work. The data assessment and needs analysis presented here is in light of the proposed model system described in the Model Development Plan (see Memorandum 2). The Model Development Plan outlines how UrbanSim is being implemented within SANDAG’s existing modeling context, and how UrbanSim integrated with existing SANDAG tools will comprise a new proposed modeling system. With a focus on data, the present memorandum builds off of the Model Development Plan (please read that document first).

Typically, when a region embarks on implementing UrbanSim, large amounts of raw data must be processed into a form usable for model estimation and simulation (i.e. an ETL step). Data from many different sources are reconciled into one common framework, and the data is then subjected to cleaning and imputation. Because of SANDAG’s long history of cutting-edge modeling, including recent ABM and PECAS implementations, a significant portion of the data work needed for implementing UrbanSim has already been carried out. Based on discussions with SANDAG and a review of their existing databases, it is clear that SANDAG already has a wealth of urban modeling data that will expedite the model development process. In many cases, existing tables will just need to be formatted and reshaped into the UrbanSim schema.

SANDAG has expended substantial effort in developing a clean base-year dataset. For example, the core assessor’s parcel data has been supplemented with information from a variety of additional sources. Commercial building data has been incorporated from a proprietary vendor (CoStar). Because the assessor typically has poor information on government buildings, government data on publically-owned buildings was incorporated. The number of stories for many non-residential buildings was populated based on elevator permit data, which has information on number of floors. Spatial and geometric operations were applied to the parcels, for example to populate distance fields and to calculate proportion overlap between parcels and environmental features that limit real estate development. Base-year data has been validated against other observed data (e.g. census residential unit counts). Extensive efforts to collect data from jurisdictions has been undertaken- for example, zoning and land use regulations, development fees, and pipeline projects. SANDAG’s archive of historical data can be used to generate targets for calibration and validation. These are just some examples of SANDAG’s significant head start in acquiring data needed to implement the proposed modeling system.

This document describes the data required to implement the proposed modeling system, and the extent to which SANDAG’s existing data resources are useful for this effort. Additional dataset collection and data processing tasks are identified as needed. The next page provides a high-level tabular overview of needed data work. Then the ensuing pages discuss key UrbanSim data tables and how SANDAG’s existing data can be used to populate these tables.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Data Category** | **Task** | **Effort** | **Owner** | **Status** |
| *Parcels and Buildings* | -Determine development types | \* | SANDAG | Done |
|  | -Prepare base-year parcel table | \*\*\* | SANDAG | Done |
|  | -Prepare base-year buildilngs table | \*\*\* | SANDAG | In-progress |
|  | -Attach all relevant location\_id's to parcel table | \* | SANDAG | In-progress |
|  | -Populate proportion\_undevelopable field | \* | SANDAG |  |
|  | -Attach local distance fields to parcel table | \* | SANDAG |  |
| *Households* | -Generate base-year synthetic population using PopSyn | \*\* | SANDAG |  |
|  | -Format synthetic population for use in UrbanSim | \* | SANDAG |  |
|  | -Assign every base-year synthetic household record a building\_id | \*\* | Synthicity | |
|  | -If sufficient sample observed in travel survey, prepare households-for-estimation table | \*\* | SANDAG |  |
| *Employment* | -Prepare base-year employment by parcel (or MGRA) and sector table | \*\* | SANDAG |  |
|  | -Synthesize UrbanSim jobs table based on the employment data provided by SANDAG | \* | Synthicity | |
|  | -Assign every base-year synthetic job record a building\_id | \*\* | Synthicity | |
| *Other tables* | -Prepare building\_sqft\_per\_job table | \*\*\* | SANDAG |  |
|  | -Prepare local street network tables (initially OSM) | \*\* | Synthicity | |
|  | -Prepare base-year zone-to-zone skims (e.g. AM peak period auto travel times and costs) | \*\* | SANDAG |  |
|  | -Prepare zonal/MGRA table with travel model-generated logsums | \*\* | SANDAG |  |
|  | -Format zoning table and send along with allowable uses table | \*\* | SANDAG | In-progress |
|  | -Format parcel-level fees table | \* | SANDAG |  |
|  | -Prepare pipeline projects table for scheduled development event as csv and/or import into UrbanCanvas | \*\* | SANDAG/Synthicity | |
| *Model integration data exchange files* | -Define file for inputs to PASEF from UrbanSim | \* | SANDAG |  |
|  | -Define file for ABM inputs from UrbanSim (MGRA residential units, employment) | \* | SANDAG |  |
|  | -Define file for LUZ-level residential unit (MF/SF) controls and the LUZ residential unit override file | \* | Synthicity | |
|  | -Define file for proforma-generated inputs to PECAS AA in year 2: LUZ space quantities. Should match FloorspaceI.csv | \* | SANDAG |  |
|  | -Define file for accessibility variables generated from ABM so UrbanSim can use updated accessibilities after each travel model year | \* | Synthicity | |
|  | -Define file for LUZ employment by sector/year control table. | \* | Synthicity | |
|  | -Define file for LUZ households by type/year control table. | \* | Synthicity | |
|  | -Define file for LUZ price by year and space type table. | \* | Synthicity | |
| *Other data processing* | -Check that base-year vacancy rates implied by data are reasonable | \* | Synthicity | |
|  | -Load building/parcel/zoning tables into UrbanCanvas database | \*\* | Synthicity | |
|  | -Prepare calibration targets (observed growth in residential\_units and non-residential sqft over some representative time period) for sub-regional geography. Targets at the Major Statistical Area geography or similar | \*\*\* | SANDAG |  |

## PARCELS AND BUILDINGS

Parcels and buildings are the core tables in the present modeling effort, which is focused on the supply side of the real estate market. The proforma-based real estate development model will add (and sometimes delete) records from the buildings table. Parcels are the unit of land on which construction takes place and the unit of land for which built-space capacity is calculated (as determined by zoning and developable proportion).

Each parcel may be associated with 0, 1, or more buildings. The parcel table should have a location identiﬁer for every other level of geography that results need to be summarized at. If model results need to be summarized at new geographic levels, additional geographic identiﬁers can be added to the parcels table.

Two parcel fields are of particular importance to the proforma-based real estate developer model. The first is zoning\_id, which relates each parcel to the zoning table. Any zoning\_id with values not found in the zoning table will be assumed to be undevelopable (for example, values of -1, -2, and -3 were observed, but no zoning\_id matches these values). The second is proportion\_undevelopable, which is the proportion of the parcel’s land area that cannot be developed (e.g. if there is an environmental constraint).

Table 1 shows parcel table columns. The development\_type\_id to be used is similar to the SANDAG PECAS space types, with a few minor changes. The land\_value column may be left unpopulated because San Diego County assessor’s data may give one assessed value instead of differentiating land value from improvement value. The ‘mgra\_id’ and ‘luz\_id’ columns are SANDAG-specific location identifiers that will be important to include for modeling purposes. The proportion\_undevelopable column will be populated based on SANDAG GIS layers for land that is undevelopable due to environmental constraints. Military land will be treated off-model, so military parcels should have proportion\_undevelopable set to 1 (i.e. UrbanSim should not build there). The ‘tax\_exempt\_status’ and ‘apn’ columns will be directly from the San Diego County assessor’s data. SANDAG has a processed parcel data table that is used for PECAS, and this existing resource will be the basis for the UrbanSim parcel table.

Street geometries are included in the SANDAG parcel data. These may be filtered out in UrbanCanvas for visualization purposes (and set as undevelopable).

Table 1: Parcel table

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| parcel\_id | integer | Unique identifier |
| development\_type\_id | integer | ID of parcel development type |
| land\_value | float | Assessed value of the parcel land |
| parcel\_acres | float | Land area of parcel in acres |
| region\_id | integer | County ID of parcel (SD is 1 county) |
| mgra\_id | integer | MGRA ID of parcel |
| zoning\_id | integer | Zoning ID of parcel |
| luz\_id | integer | LUZ ID of parcel |
| msa\_id | integer | MSA ID of parcel |
| proportion\_undevelopable | float | Proportion of land undevelopable |
| tax\_exempt\_status | integer | 1 if tax exempt, else 0 |
| apn | string | Local identifier in assessor data |
| shape | geometry | Geometry of parcel |
| centroid | geometry | Geometry of parcel centroid |

SANDAG’s base-year buildings table will be derived from the parcel data. If the data supports it, there can be multiple buildings on a parcel (a many to one relationship between buildings and parcels). Each building links to a speciﬁc parcel id. The buildings table is modiﬁed by the proforma-based real estate development model (which introduces new buildings and in some cases demolishes old ones) and price models (which update the price ﬁelds). The scheduled development events model will also add records to this table. The buildings table represents the built space that agents in the simulation will occupy. Each household and job will be assigned to a speciﬁc building.

Table 2 shows building table columns. The improvement value may reflect total assessed value of the parcel because the San Diego County assessor does not break assessed value into improvement value and land value. For many commercial records, the non\_residential\_sqft column is imputed based on CoStar data. The price\_per\_sqft column may be left as null, to be filled in by hedonic model predictions at the beginning of the simulation. The stories column is supplemented by data from elevator permits. Data on government buildings is incorporated from a separate data source on publically-owned buildings because the county assessor does not always represent public properties in detail. The year\_built column may be an aggregation of multiple dates in the assessor’s data, for example in the case of a major remodel or expansion.

In additional to being used for non-residential square footage, CoStar data was also used for supplementing the following fields in some cases: year\_built, stories, improvement value (last sale price). If CoStar last sale price is used to populate improvement value, adding a last sale data column may be useful.

Table 2: Building table

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| building\_id | integer | Unique identifier |
| development\_type\_id | integer | ID of building development type |
| parcel\_id | integer | Parcel ID of building |
| improvement\_value | float | Assessed value of building |
| residential\_units | integer | Total residential units in building |
| residential\_sqft | integer | Residential square footage |
| non\_residential\_sqft | integer | Non-residential square footage |
| price\_per\_sqft | float | Market price per square foot |
| stories | integer | Number of stories in building |
| year\_built | integer | Year in which building was built |
| geometry | geometry | Geometry of building footprint |

**Parcel and building table tasks:**

* Determine development types
* Prepare base-year parcel table
* Prepare base-year building table
* Attach all relevant location\_id’s to parcel table
* Populate proportion\_undevelopable field on parcel table
* Attach local distance fields to parcel table

## HOUSEHOLDS

The household table contains synthesized 2012 household data for the SANDAG region, with each row pertaining to one synthesized household. This will be generated by PopSyn, the SANDAG population synthesizer. During simulation, new households will be introduced into this table based on LUZ-level household control totals provided by PECAS AA. The building\_id of households will be populated by UrbanSim’s Household Location Choice Model.

If the SANDAG travel survey contains information on recent-mover status of households, and a sufficient sample of recent-movers exists, a households-for-estimation table should be prepared. It will be used in estimation and should be formatted just like the synthetic households table. In the absence of observed recent-movers, it is recommended that the population be synthesized with recent-mover status (“when moved into current dwelling”) as a control variable.

Optionally, a persons table can be used as well. This is useful for calculating certain explanatory variables, and also useful if microsimulating demographic processes is planned in the future. The persons table contains one record for each synthetic person, linked to households via a household\_id field. Suggested person attributes include age, earnings, household\_id, person\_id, race\_id, relationship to householder, sex, student status, and working status.

Table 3 shows household table columns. Since SANDAG runs an in-house population synthesizer (PopSyn) it is anticipated that this will be a straightforward table to create.

Table 3: Household table

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| household\_id | integer | Unique identifier |
| building\_id | integer | Building household resides in |
| tenure | integer | own (1) vs rent (2) |
| persons | integer | Number of people in household |
| workers | integer | Number of workers in household |
| age\_of\_head | integer | Age of head of the household |
| income | integer | Income of household |
| children | integer | Number of children in household |
| race\_id | integer | Race of head of household |
| cars | integer | Number of cars in household |

**Household table tasks:**

* Generate base-year synthetic population using PopSyn
* Format synthetic population for use in UrbanSim
* Assign every base-year synthetic household record a building\_id
* If sufficient sample of recent-movers observed in travel survey, prepare households-for-estimation table.

## JOBS

The jobs table contains one record for each job in the region. This will be generated based on existing SANDAG employment data (at the parcel or MGRA level). During simulation, new jobs will be introduced into this table based on LUZ-level employment control totals provided by PECAS AA. The building\_id of jobs will be populated by UrbanSim’s Employment Location Choice Model.

The home\_based\_status field is optional, and indicates whether the job is based in a residential unit. Note that, just as with the households table, it is possible to have a jobs\_for\_estimation table if a specific sample of jobs should be used for estimating the employment location choice model (e.g. if data is available on recently locating jobs).

Table 4 shows the job table columns. Synthicity can generate this table using existing scripts if SANDAG provides aggregate employment totals by sector (and optionally home-based status) at the parcel or MGRA level.

Table 4: Job table

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| job\_id | integer | Unique identifier |
| building\_id | integer | Building that job is located in |
| sector\_id | integer | Employment sector ID |
| home\_based\_status | integer | 1 if home-based, else 0 |

**Job table tasks:**

* Prepare base-year employment by parcel (or MGRA) and sector table
* Synthesize UrbanSim jobs table based on the employment data provided by SANDAG
* Assign every base-year synthetic job record a building\_id

## BUILDING\_SQFT\_PER\_JOB

This table contains the non-residential square footage each job will occupy in each combination of development type and zone. This information is used to calculate the number of job spaces in each building and non-residential vacancy rates. The building square-foot per job assumptions should vary geographically and by development type, but if detailed data is not currently available, placeholder values can be used.

SANDAG has already collected similar job square footage data for use in PECAS. Table 5 shows the table columns.

Table 5: Building\_sqft\_per\_job table

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| zone\_id | integer | Location identifier (MGRA or LUZ) |
| development\_type\_id | integer | Development type ID of record |
| building\_sqft\_per\_job | integer | Square-footage occupied by job |

**Building\_sqft\_per\_job tasks:**

* Prepare building-sqft per job table

## NETWORK NODES/EDGES

UrbanSim’s internal network queries require a representation of the local street network. Initially, Synthicity will prepare these tables based on OpenStreetMap data. SANDAG can swap in tables based on their own network data at their convenience.

The network is comprised of a set of nodes and edges. The nodes table contains x and y coordinates, and is indexed by node\_id. The edges table contains a “from” column, a “to” column, and a weight column. The from and to fields should be the node\_id’s that compose the edge. The weight field should be a measure of the impedance of the edge, e.g. distance or travel time. See Tables 6 and 7.

Table 6: Nodes table

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| node\_id | integer | Unique identifier |
| x | float | X attribute of node (e.g. longitude) |
| y | float | Y attribute of node (e.g. latitude) |

Table 7: Edges table

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| from | integer | Node ID that begins the edge |
| to | integer | Node ID that ends the edge |
| weight | float | Network impedance |

**Network tables tasks**

* Prepare local street network tables (initially OSM)

## TRAVEL\_DATA AND ZONES

The travel\_data table contains skims- zone-to-zone travel times and costs. This table allows accessibility variables based on skims to be added to the specification during the model estimation process. It is used to calculate accessibility variables such as “employment within 20 minutes auto driving time”. The travel data table will be updated with the results of any in-simulation travel model run. This automated updating will be implemented as part of travel model integration tasks.

The zones table contains a list of trafﬁc analysis zones (MGRAs) used in the travel model. Characteristics of each zone are represented by columns in this table. This table is also where travel-model-generated logsums should be stored.

SANDAG will be able to generate these tables from travel model (ABM) output. It is preferred that this data comes from standard ABM output so that custom accessibility metrics do not need to be generated.

Table 8: Travel Data table

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| from\_zone\_id | integer | Origin zone (or MGRA) |
| to\_zone\_id | integer | Destination zone (or MGRA) |
| travel\_time | float | Zone-to-zone peak travel time |

Table 9: Zones table

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| zone\_id | integer | Zone ID (or MGRA) |
| logsum | float | Logsum value |

**Travel data and zones tasks:**

* Prepare base-year zone-to-zone skims (e.g. AM peak period auto travel times and costs)
* Prepare zonal/MGRA table with travel model-generated logsums

## PROFORMA INPUTS

Built space capacity at the parcel level is a key input to the proforma model. For each parcel, the proforma requires information on maximum floor-area-ratio (FAR), maximum dwelling units per acre (DUA), proportion of parcel that is undevelopable, and allowable development types under zoning. Additional inputs to the proforma include development fees and construction costs. If maximum building height and other zoning requirements are available, these can be used as well.

SANDAG has existing data on zoning, development fees, and construction costs from development of the PECAS SD module. A review of the existing zoning and fee tables revealed that the needed data was already in a form similar to what is needed for UrbanSim’s proforma model. See Table 10 for zoning table columns that can be utilized, depending on data availability.

Table 10: Zoning table

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| zoning\_id | integer | Unique identifier |
| jurisdict\_id | integer | Jurisdiction ID |
| zone | string | Name of zoning class |
| zonecode | varchar | Combination of jurisdiction\_id and zone |
| min\_far | float | Min floor area ratio |
| max\_far | float | Max floor area ratio |
| min\_front\_setback | float | Min front setback distance |
| max\_front\_setback | float | Max front setback distance |
| rear\_setback | float | Rear setback distance |
| side\_setback | float | Side setback distance |
| min\_dua | float | Min dwelling units per acre |
| max\_dua | float | Max dwelling units per acre |
| max\_building\_height | integer | Max height of buildings in feet |
| allowed\_uses | array/list | List of allowable development types. Alternatively, separate table with every combination of development type and zoning\_id |
| shape | geometry | Geometry of zoning polygon |

**Proforma data tasks:**

* Format zoning table
* Format parcel-level fees table

## Other Data Processing tasks

* Check that base-year vacancy rates implied by data are reasonable
* Load building/parcel/zoning tables into UrbanCanvas database
* Prepare pipeline projects table for scheduled development events and import into UrbanCanvas
* Prepare calibration targets (observed growth in residential\_units and non-residential sqft over some representative time period ) for sub-regional geography. Targets at the Major Statistical Area geography or similar

## Model Integration Data Formats

This section lists the intermediate simulation data that is needed for exchange of information between models in the proposed forecasting system. Example files will be stored as CSV’s on Pivotal Tracker. For additional information (and short examples), please see the Model Integration section of the Model Development Plan Memorandum. Also, please see the model sequence addendum to Memo 2, along with the schema tables below (Tables 11 – 15).

* Inputs to PASEF from UrbanSim
* ABM inputs from UrbanSim (MGRA residential units, employment)
* LUZ-level residential unit (MF/SF) controls from DEFM
* User-specified LUZ residential override file
* Proforma-generated inputs to PECAS AA in year 2: LUZ space quantities (FloorspaceI.csv)
* Accessibility variables generated from ABM so UrbanSim can use updated accessibilities after each travel model year
* LUZ employment by sector/year control table.
* LUZ households by type/year control table.
* LUZ price by year and development type table.

Table 11: Residential unit controls from DEFM (annual\_residential\_unit\_control\_totals.csv)

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| year | Integer | Simulation year |
| unit\_type | Integer | Single-family (1) vs Multifamily (2) |
| total\_number\_of\_residential\_units | Integer | Control total for residential units |

Table 12: User-specified LUZ residential unit overrides (luz\_residential\_unit\_overrides.csv)

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| year | Integer | Simulation year |
| luz\_id | Integer | ID of the LUZ |
| unit\_type | Integer | Single-family (1) vs Multifamily (2) |
| total\_number\_of\_residential\_units | Integer | Control total for residential units |

Table 13: LUZ household controls from PECAS (households\_by\_luz.csv’)

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| luz\_id | Integer | ID of the LUZ |
| hh\_type | Integer | Household type ID, defined in pecas\_household\_types.yaml |
| total\_number\_of\_households | Integer | Control total for LUZ households |

Table 14: LUZ employment controls from PECAS (employment\_by\_luz.csv’)

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| luz\_id | Integer | ID of the LUZ |
| sector\_id | Integer | ID of the industrials sector |
| total\_number\_of\_jobs | Integer | Control total for LUZ employment |

Table 15: LUZ price from PECAS (price\_by\_luz.csv’)

|  |  |  |
| --- | --- | --- |
| Column Name | Data Type | Description |
| luz\_id | Integer | ID of the LUZ |
| space\_type\_id | Integer | ID of the PECAS space type |
| price\_per\_sqft | Float | Price per square-foot value |

## Typology Tables

Table 16: Development Types

|  |  |
| --- | --- |
| development\_type\_id | name |
| 1 | Agriculture and Mining |
| 2 | Light Industrial |
| 3 | Heavy Industry |
| 4 | Office |
| 5 | Retail |
| 6 | Depot Space |
| 7 | Hotel/Motel |
| 8 | Primary Schools |
| 9 | Secondary Schools |
| 10 | Post-Secondary Institution |
| 11 | College Dormitory |
| 12 | Health Care |
| 13 | Religious |
| 14 | Recreation |
| 15 | Active Park |
| 16 | Government Operations |
| 17 | Dump Space |
| 18 | Mixed Use |
| 19 | Single Family Detached Residential |
| 20 | Single Family Attached Residential |
| 21 | Multi-Family Residential |
| 22 | Mobile Home |
| 23 | Military Residential (Non GQ) |
| 24 | Transportation Right-of-way |
| 25 | Parking Lot |
| 26 | Undeveloped open space |
| 27 | Beach |
| 28 | Water |
| 29 | Military Reservation |
| 30 | Indian Reservation |
| 31 | Vacant Developable Land |

# Appendix 5: Memorandum 2

**SANDAG Subregional Allocation Model**

**Model Development Plan**

**Technical Memorandum 2**

Contents

[Introduction 48](#_Toc405964287)

[High-level Model System Overview 49](#_Toc405964288)

[URBANSIM COMPONENTS 51](#_Toc405964289)

[Demand-side Allocation 51](#_Toc405964290)

[Price Model 52](#_Toc405964291)

[Supply-side Predictions 53](#_Toc405964292)

[Model Integration 56](#_Toc405964293)

[Network-based Accessibility Variables 63](#_Toc405964294)

[Model Estimation Approach 64](#_Toc405964295)

[Phase 1 Calibration Strategy 65](#_Toc405964296)

[Roadmap and Responsibilities 66](#_Toc405964297)

[MODEL OVERVIEW TABLES 69](#_Toc405964298)

# Introduction

This document seeks to formalize the design of the Phase 1 SANDAG Subregional Allocation Model, a proposed forecasting system for making spatially-detailed built space predictions in the San Diego region, along with small-area household and employment allocation. After an overview of the model system, this document describes each UrbanSim model component to be implemented. Then, a roadmap of tasks is laid out, with time estimates and responsibilities. Referenced tables can be found at the end of this document.

The main focus of this proposal is on the supply-side of the real estate market. Given regional capacity constraints, redevelopment and infill are becoming increasingly important to explicitly represent in supply-side models. UrbanSim’s proforma-based model of real estate development is well-suited to capture these types of development.

The model system proposal is based on extensive discussion with SANDAG about modeling needs, requirements of the forecasting process, and the most appropriate role for UrbanSim in the context of SANDAG’s existing modeling environment. The flow of information between the components of the model system will be described, along with the specification of each component. PECAS and UrbanSim will collectively replace most of the subregional allocation role that UDM occupied in the SANDAG modeling ecosystem. PECAS will drive the demand-side of the real estate market and UrbanSim will drive the supply-side.

Under this proposal, SANDAG would use the following models in its land use forecast: 1.) DEFM, 2.) PECAS AA, 3.) UrbanSim, 4.) PASEF, 5.) PopSyn, 6.) ABM. A primary motivation behind subregional allocation is the need to provide socioeconomic and land use inputs to the travel model (ABM), which will update accessibilities and inform the subsequent allocation. There will be substantial feedback of information between models as the simulation progresses in annual time steps through the forecast period. The interface between models will be defined so that integrated model runs are automated and convenient. A principle in the proposed model system is that, to the extent possible, data exchanges are automated but all pre-existing models are kept intact. Beyond Phase 1 there is potential for tighter integration and customization.

# High-level Model System Overview

This section gives a brief overview of the proposed model system. Figures 1 and 2 illustrate diagrammatically. In a nutshell: DEFM will handle regional predictions, PECAS AA will handle LUZ-level household and employment predictions, and UrbanSim will handle parcel-level predictions.

*Goal: Small-area real estate development predictions for the SANDAG region using the UrbanSim proforma-based developer model. The proforma model will be implemented for the SANDAG region, along with infrastructure needed to generate the parcel-level proforma predictions.*

Starting at the regional level, DEFM generates capacity-constrained regional forecasts of residential units by type (single-family, multi-family). DEFM also produces a regional population and employment forecast. These aggregate forecasts are provided for every simulation year.

In the SANDAG region, there are 229 LUZs, an intermediate modeling geography. On the real estate demand side, PECAS AA allocates population and employment from the regional level to the LUZ level. PECAS also predicts an LUZ-level price by development type. The real estate supply side will primarily be modeled at the parcel level, but a mechanism will be provided for asserting residential unit totals at the LUZ level as needed (see ‘DU Overrides’ in Figure 1 and ‘Manual Overrides’ in Figure 2).

Once LUZ-level values are in place (households, employment, price, and optionally residential units), UrbanSim makes parcel-level predictions for households and jobs, treating the LUZ values as sub-regional controls. For each LUZ, UrbanSim allocates the increment of change in households and jobs from the previous year to the building and parcel level using location choice models estimated on local data. UrbanSim also predicts building-level prices, in a way that accounts for the PECAS LUZ-level price prediction.

Once prices and demand-side agents are at the disaggregate building/parcel-level, the UrbanSim proforma-based real estate development model makes built-space predictions. First, the set of all profitable developments on all parcels in the region is calculated. This calculation accounts for model inputs such as zoning (capacities and allowable development types), proportion of parcel area that is undevelopable, construction costs, and fees. Next, potential developments are sampled for construction such that the DEFM-provided regional residential unit controls by type are matched. Here, type means single-family or multi-family. Sampling of projects is weighted by development profitability at the project-level (project revenue – project costs). Only profitable developments that fit within parcel-level zoning constraints are considered. If optional LUZ-specific residential unit totals are asserted via the override mechanism, UrbanSim will match this residential unit target subject to the availability of profitable development projects within the LUZ.

Updated built space quantities are then reported back to PECAS AA for the next simulation year. During travel model years, households are aggregated to the MGRA level and passed to PASEF, which will in turn pass demographic predictions to PopSyn, which will synthesize a population for ABM. As needed, employment and built-space predictions are aggregated to the MGRA level and passed to ABM during travel model years. Results from ABM (logsums, skims etc.) will in turn update the accessibility metrics used by both PECAS AA and UrbanSim allocation in the following simulation year.

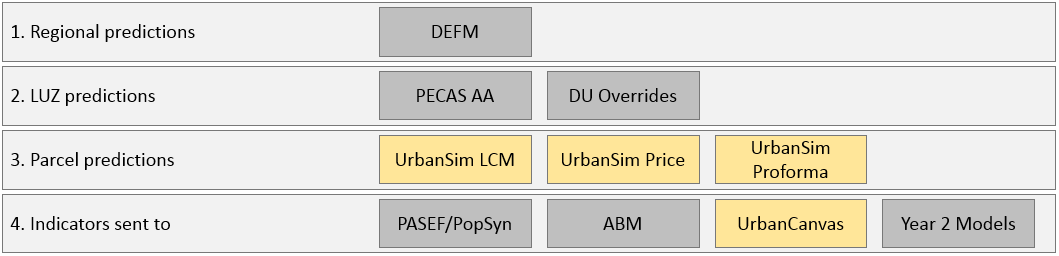


Figure 1: Model system overview by geographic level of operation

In terms of modeled inputs to the UrbanSim components of the model system: PECAS AA will feed three tables to UrbanSim: 1.) LUZ controls for number of households by type, 2.) LUZ controls for employment by sector, and 3.) LUZ price by space type. For the base-year (2012), UrbanSim requires a synthetic population as a starting point for the simulation.  The base-year synthetic population will be generated by PopSyn (recommended attributes of the household table are listed in the Data Memorandum).

The mechanics of model integration are described in the Model Integration section, further down in this memorandum. Also, specific tasks related to integration of model system components are listed in the “Model Integration” section of the task roadmap.

This section was just a brief overview of the proposed model system. Each point is elaborated on later in the document. Some points to keep in mind as a recap: Number of households by type and employment by sector for LUZs is predicted by PECAS AA. Prices by space type are predicted by PECAS AA (as noted later, these PECAS prices are better interpreted as a measure of relative attractiveness). The proforma model will handle allocation of residential units to the parcel level, skipping the LUZ intermediate geography entirely unless the LUZ is included as part of an override table where a specific residential unit total is asserted. For LUZs represented by this override table, specified quantities of residential units will be allocated down to the parcels within the LUZ.

Figure 2 shows the model system and inter-model interactions. See the caption for some relevant notes.

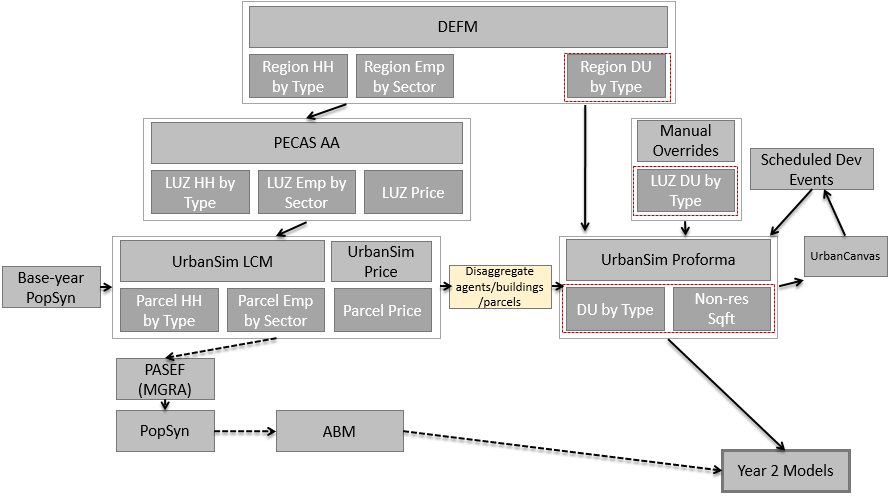


Figure 2: Model system overview- component outputs and model interactions. Red dotted lines indicate a zoning-capacity-constrained outcome. Black dotted lines indicate model interactions that will only take place during travel model years (typically not every simulation year). The UrbanSim proforma model generates built space predictions at the end of each simulation year, and newly generated built space is not occupied by demand-side agents (i.e. households and jobs) until the following year.

# URBANSIM COMPONENTS

Each UrbanSim component model to be implemented will be described in turn. The models will run in the following order: demand-side allocation, price model, supply-side predictions. Referenced model overview tables can be found at the end of this document.

## Demand-side Allocation

The UrbanSim location choice models will be implemented for the purpose of allocating LUZ level demand-side agents (households, employment) to the building and parcel level. Each simulation year, PECAS will generate for each LUZ the number of households by type and the amount of employment by sector. UrbanSim will compare these totals with the number of households and employment already in each LUZ. If the totals indicate an increase in agents, new agents will be created and allocated to a building within the LUZ. If the totals indicate a decrease in agents, existing agent records will be sampled for deletion. The allocation of new agents down to buildings will be carried out using multinomial logit (MNL) location choice models, with random sampling of alternatives from the universe of vacant units within each LUZ.

In Figure 2, see the gray box titled “UrbanSim LCM”. In the model system, the location choice models will be run right after PECAS AA. The location choice models are divided into a household location choice model (HLCM) and an employment location choice model (ELCM). Each model is further segmented into submodels (e.g. households by income quartile, employment by sector). Each segment will be estimated and simulated separately, and model specifications will vary. The variables used in the model will be drawn from the literature in urban economics, urban geography, and urban sociology.

See Tables 1 and 2 for model details and an initial specification in terms of explanatory variables. The household models will be estimated using either observed recent-movers in a travel survey or synthetic recent-movers from a synthetic population. The employment models will be estimated off of base-year synthetic job records. See the Data Memorandum for additional information on estimation data.

After demand-side agents are allocated to buildings, the disaggregate records will be summarized for the purpose of passing information to other models. For example, MGRA-level number of households will be calculated and sent to PASEF in travel model years. Because the demand-side will be represented in terms of disaggregate agents in buildings, the data can be summarized at any geographic level.

Base-year data requirements for simulating the location choice models include a synthetic population from PopSyn, disaggregate job records (which can be generated from aggregate employment totals by sector), and a ‘building\_sqft\_per\_job’ table. The base-year synthetic population for each LUZ will be evolved as new LUZ totals are received from PECAS. For example, if high-income households are predicted to increase in the LUZ, existing high-income households will be sampled for replication, then located to a building. If households with the needed attributes are found within the LUZ, this will be the pool that will be sampled from for replication; otherwise households will be sampled from the entire pool of appropriate households in the region. The ‘building\_sqft\_per\_job’ table indicates the amount of square footage each job can occupy in a particular development type and geography. This is used to calculate the number of job spaces in each building: job\_spaces = non\_residential\_sqft/building\_sqft\_per\_job. These data inputs will be further elaborated on in the Data Development memo.

In terms of modeled inputs during simulation, PECAS will generate LUZ households by type/year and LUZ employment by sector/year, and these tables should be formatted as defined in Pivotal and the Model Integration section of this document. The LUZ household controls from PECAS should not include military households locating in military housing, as military and GQ will not be modeled by UrbanSim.

## Price Model

Prices are a key input to the UrbanSim proforma model. Building-level price per square footage will be simulated in UrbanSim using hedonic regression models. Specifications may be borrowed from the existing hedonic regressions that SANDAG estimated for PECAS as a started point in the estimation process, and then additional variables added. The real estate price model will be segmented by structure type- separate regression equations will be estimated and simulated for each structure type. PECAS will continue to make predictions of LUZ-level prices in the proposed model system. The PECAS-predicted base-year LUZ price will be included as an explanatory variable when estimating the building-level regressions, and a positive coefficient is expected. During simulation, PECAS-predicted LUZ prices will influence UrbanSim’s building-level price predictions via this explanatory variable. As PECAS-predicted prices increase, all else equal, building-level prices in UrbanSim will also increase.

In Figure 2, see the gray box titled “UrbanSim Price”. The price models will be run after the UrbanSim location choice models and prior to the UrbanSim proforma-based developer model. Predicted prices will be influenced by location choice model outcomes- for example, if employment density, walking-scale proximity to jobs, or vacancy rate within a quarter mile are included as explanatory variables. And predicted prices will in turn impact the profitability of projects as predicted by the proforma model.

Price will play a role in mediating between supply and demand in the simulation. High-demand areas will have their price bid upwards. These higher prices will ration units for demand-side agents and also make areas more attractive to real estate development. A disamenity- say nearby industrial uses- may depress prices, resulting in greater affordability for demand-side agents but less profit opportunity for real estate developers. Adjustments in price will alter location preferences within each LUZ. The price models are an avenue of feedback between the demand and supply sides of the model. For this reason, and because price is such an important input to the proforma model, which is a focus of this modeling effort, extra attention will be paid to the real estate price model specifications.

To summarize the link between PECAS AA LUZ prices and UrbanSim: base-year PECAS AA LUZ prices will be included as explanatory variables in the building level regressions, and this will be carried forward as a predictor during annual simulations (as the LUZ price values are updated by PECAS over time, UrbanSim will use these updated prices to inform its building-level price predictions).

See Tables 3 and 4 for an overview of the real estate price model structure and specification. Real estate prices are modeled using a regression of the log-transformed property value per square foot on explanatory variables. Prices are updated by UrbanSim annually.

## Supply-side Predictions

This section will discuss how the supply side of the real estate market will be represented in the proposed model system. This is the core focus of the present modeling effort, and the preceding steps (location choice models, price models) are run to get parcel inputs in place that are needed for the supply-side modeling. The supply side will be represented by the UrbanSim proforma-based real estate development model along with a ‘scheduled development events model’ (UrbanSim parlance for the SANDAG Site Spec). The proforma model will be used for predicting new market-driven built-space construction. The scheduled development events model will be used to represent known pipeline projects, built-space assertions, and other specific development projects with various degrees of potential for being constructed. See Tables 5 and 6 for a tabular overview of these supply-side models.

The proforma-based model simulates the location, type, and density of new real estate development, including redevelopment, at the level of specific parcels. The model predicts multiple parameters of development projects in order to maximize profitability of development outcomes, subject to local physical, regulatory, and market contexts. The model explicitly represents redevelopment and the impact of zoning (and other development constraints), in line with SANDAG’s supply-side modeling priorities.

A real estate proforma is an analytic method, usually implemented in a spreadsheet, which developers often use for evaluating the cash flow associated with specific development projects. The UrbanSim proforma model is similar, except implemented in Python code and run for every parcel in the entire region. The goal is to represent the behavior of real estate developers (see Figure 3)

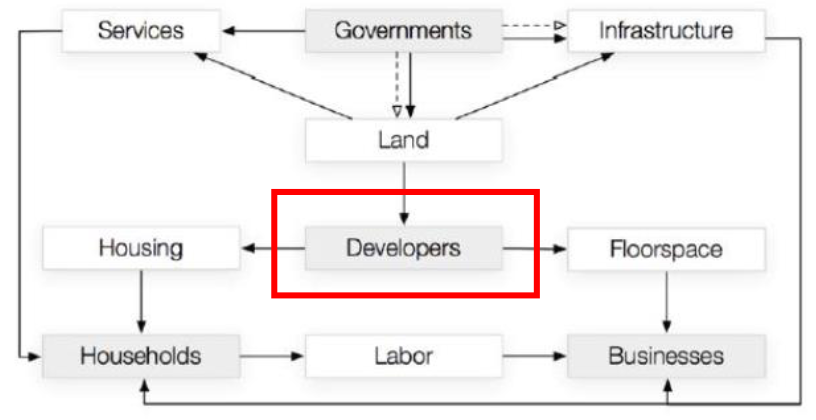


Figure 3: Developers in the real estate market

For each allowable building type on every parcel in the region, the developer model predicts the building form that results in the greatest profitability. Only buildings that fit within zoning constraints and developable-land constraints are considered. An optional redevelopment flag can be used to filter out redevelopment projects from consideration for parcels on which redevelopment should not occur. In addition to regulatory constraints and fees, key inputs to the proforma model include real estate prices and construction costs. Once the set of profitable projects is calculated, specific projects are sampled for actual construction in order to meet demand. Demand can be expressed as either a unit target to match or a structural vacancy rate, and the geographic specificity of this demand input is flexible (e.g. LUZ or regional). Sampling of projects is weighted by predicted profitability. If a sampled project involves redevelopment, the existing structure located on the parcel is demolished and agents associated with the original structure are unplaced and re-allocated to buildings within the LUZ.

The SANDAG proforma model will be configured to receive regional LUZ controls for dwelling units by type from DEFM. If dwelling units controls are provided at the regional level only, the proforma model allocates dwelling units to parcels directly, skipping the LUZ geography entirely. SANDAG will have the option of controlling residential unit results at the LUZ level using an override mechanism. For each LUZ for which SANDAG asserts a residential unit total, the proforma model will allocate this total to parcels within the specified LUZ. The LUZ residential unit target will be matched unless sufficient profitable projects do not exist within the LUZ or zoned capacity does not exist. If the override target is not met, the remaining units that belonged to the target LUZ are allocated to LUZs that are not subject to overrides. This carry-over from controlled LUZ to uncontrolled LUZ serves to ensure that DEFM regional residential unit targets are met.

See Figure 4 for an illustration of the override mechanism. In this mock region, there are 4 LUZs (A, B, C, D) and each LUZ contains two parcels (A1, A2, B1, B2, etc.). The model user specifies residential unit override targets for LUZs C and D, but leaves LUZs A and B uncontrolled. Thus there are two groups of residential units- those involved in an LUZ override and uncontrolled residential units that can be allocated to any parcel within any LUZ that is not subject to overrides. For each LUZ that is subject to an override, the UrbanSim proforma allocates the target number of residential units to parcels within the LUZ. For example, the target for LUZ C gets allocated to parcels C1 and C2. Residential units in the uncontrolled residential unit bucket get allocated to any parcel in an uncontrolled LUZ (left side of the figure, parcels A1, A2, B1, B2). If an LUZ for which an override is specified has its residential unit target undershot due to unprofitability or lack of zoned capacity, the remaining units that could not be built are moved to the uncontrolled residential unit bucket.

Please also see the Model Integration section of this document for further information on how DEFM controls should be formatted for UrbanSim and how the LUZ override table should be formatted.

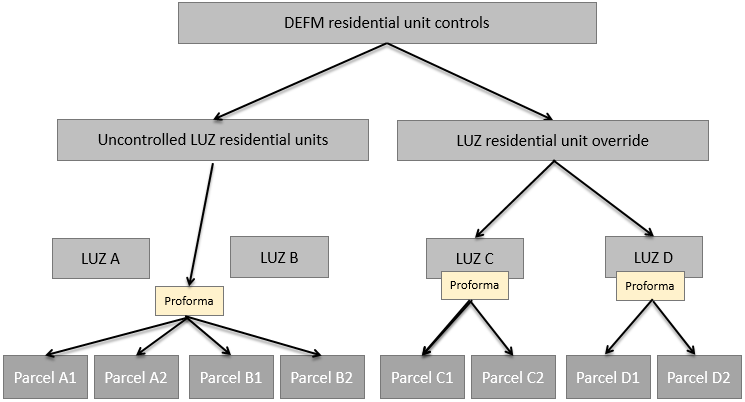


Figure 4- Illustrating the LUD residential unit override mechanism

The proforma model will run on the same disaggregate buildings, parcels, and agents tables used and updated by the UrbanSim location choice models and price models (see the light-orange rectangle near the center of Figure 2). The proforma model will run annually in the simulation, after the location choice models have allocated agents to the building level and after the price models have updated each building’s price per square foot prediction.

Note that if an insufficient number of profitable projects exist to match override targets in certain LUZs, UrbanSim will not be able to meet the residential unit target in those LUZs. This under-shooting of controls is logged. Future efforts can incorporate tools such as Vision Solver into the model system, which gives the option of forcing controls to be met regardless of profitability and then reporting on the development subsidy needed to achieve the development vision. Built space targets for specific areas (such as a priority-development TOD area) are provided to Vision Solver, which then calculates the development subsidy needed to meet each target in the case that development is not profitable enough to achieve the vision without help.

Built space capacity at the parcel level is a key input to the proforma model. For each parcel, the proforma requires information on maximum floor-area-ratio (FAR), maximum dwelling units per acre (DUA), proportion of parcel that is undevelopable, and allowable development types under zoning. Military land should be wholly undevelopable, as SANDAG will model military built-space development in a separate process.

Known pipeline projects, built-space assertions, and other specific development projects with various degrees of potential for being constructed will be represented in the simulation by the scheduled development events model. This model supplements the proforma predictions with information on known or scenario-based development projects. It is anticipated that this infrastructure will be able to represent the SANDAG business-rules for forecasting built space. Development projects can either be committed or uncommitted. Committed projects are inserted into the buildings table as scheduled, regardless of circumstance. Uncommitted projects trigger an upzoning to accommodate the project’s proposed built space (if upzoning is necessary), and then allows the proforma model to decide how much of the zoned capacity gets utilized. Development projects can be input to UrbanSim either via a csv file or via UrbanCanvas. UrbanCanvas also allows for the scheduling of rezoning/upzoning events directly in the simulation.

After the UrbanSim supply-side models run, predictions are aggregated and sent to other models. For example, built-space LUZ aggregations will be sent to PECAS AA for the following year’s simulation. This will inform the amount of households and employment PECAS can allocate to each LUZ in the subsequent year (e.g. PECAS will respect the number of residential units in each LUZ when allocating households). If the ABM requires MGRA-level values for number of residential units and amount of non-residential square-footage, these indicators will be generated for travel model years. Parcel-level built-space predictions are sent to UrbanCanvas after each simulation year for visualization purposes.

# Model Integration

Aspects of model integration have been mentioned throughout this document, but this section dives into more depth and describes the links between models in the proposed model system with a focus on specifying data exchange file formats. Each model will need to export data in the appropriate form for consumption by other models (tasks associated with generating output files are listed in the Roadmap section, and also in the data memo). Note that every example table shown below has an associated example file that is uploaded to Pivotal and attached to the relevant story.

*DEFM - UrbanSim*

The first integration link to be discussed is the one between DEFM and the UrbanSim proforma-based real estate development model. This linkage refers to the subset of Figure 2 shown below in Figure 5. DEFM is run on its own, in advance of the small-area simulation, and predicts the number of residential units in the region for every year in the forecast horizon. The residential unit totals are broken down by type (single-family vs multi-family).

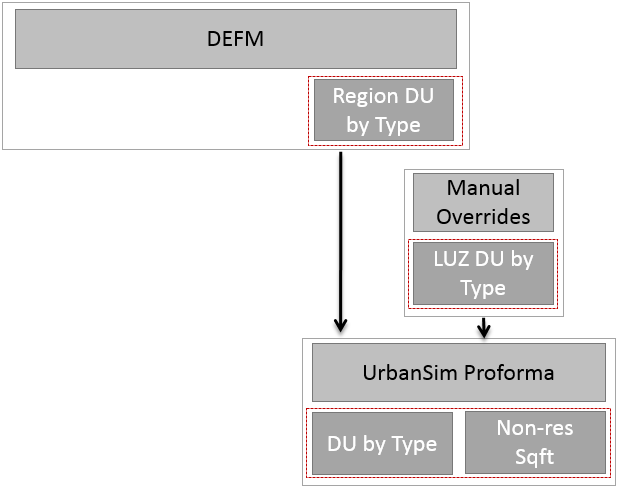


Figure 5- DEFM - UrbanSim linkage

For input to UrbanSim, the DEFM forecast should be formatted as shown in Table 7. For each year and unit type, the total number of residential units in the region should be given in this table. Each row corresponds to a unique combination of year and unit type, and the ‘total\_number\_of\_residential\_units’ column gives the number of units for that combination. The ‘unit\_type’ column should be populated with values of 1 (single-family) or 2 (multi-family). In the mock example shown in Table 1, the number of single-family residential units grows from 900,000 in year 2012 to 1,093,956 in year 2016. The number of multi-family residential units grows from 250,000 in year 2012 to 327,699 in year 2016.

This annual residential unit control total table is stored as a .csv file in the UrbanSim data input directory and named ‘annual\_residential\_unit\_control\_totals.csv’. This will be the default file name UrbanSim will look for, but the configuration file for the proforma model can be edited to point towards a csv with a different name. Multiple control total .csv files can be generated, reflecting different DEFM assumptions about regional growth, and swapping between them in UrbanSim is as straightforward as editing the configuration argument to point to the appropriate DEFM-generated file.

Table 7- Annual residential unit control totals

|  |  |  |
| --- | --- | --- |
| year | unit\_type | total\_number\_of\_residential\_units |
| 2012 | 1 | 900000 |
| 2012 | 2 | 250000 |
| 2013 | 1 | 945000 |
| 2013 | 2 | 267500 |
| 2014 | 1 | 992250 |
| 2014 | 2 | 286225 |
| 2015 | 1 | 1041863 |
| 2015 | 2 | 306261 |
| 2016 | 1 | 1093956 |
| 2016 | 2 | 327699 |

In each simulation year, UrbanSim will compare the existing number of residential units in the simulation (aggregating the parcel data up to the regional level) with the target value from the control totals. For example, if the year 2013 is being simulated, the following steps would be taken for single-family residential units (assuming the mock table above):

1. Calculate number of single-family residential units in the existing buildings/parcels table. 900,000 single-family residential units are found.
2. Look up the target number of single-family residential units for year 2013 in the annual\_residential\_unit\_control\_totals. This lookup finds a target of 945,000.
3. Subtract the existing value from the target to arrive at the number of new single-family residential units to introduce into the simulation. 945,000 – 900,000 = 45,000. The proforma model will then build 45,000 single-family residential units in year 2013. Note that since this is the last UrbanSim model to be run in year 2013, these new units will not be available for occupancy by households until the 2014 simulation year.

In addition to annual\_residential\_unit\_control\_totals, UrbanSim will accept an optional input table specifying LUZ level residential unit overrides. This table, stored as a .csv file, contains asserted values for number of residential units by LUZ, type, and year. It can be populated with as many or as few rows as desired. Each row represents a year/LUZ/unit-type combination and gives the number of residential units for that combination. Any year/LUZ/unit-type combination that is not represented in the table will be freely developed by the UrbanSim proforma model (subject to the usual constraints like zoning and profitability). Any combination that does exist means that the proforma model will respect this LUZ-level control and build enough to exactly match this target (assuming sufficient profitable projects exist in the LUZ).

The override table is represented by a .csv file named ‘luz\_residential\_unit\_overrides.csv’, to be stored in the UrbanSim data input directory. See Table 8 for an example of this table. Three LUZs are controlled (luz\_id 1, 2, and 3), and for each unit type (1: single-family, 2: multi-family) the number of residential units in the LUZ is given. In year 2014, LUZ 3 will contain 860 multi-family residential units. Just as with the DEFM-based annual\_residential\_unit\_control\_totals, residential unit numbers in this table should give the total number of residential units (rather than the increment of change). UrbanSim will then calculate the needed increment/decrement by comparing the target value for the LUZ with the existing number of residential units in the LUZ.

Table 8- LUZ residential unit overrides

|  |  |  |  |
| --- | --- | --- | --- |
| year | luz\_id | unit\_type | total\_number\_of\_residential\_units |
| 2012 | 1 | 1 | 400 |
| 2012 | 1 | 2 | 200 |
| 2012 | 2 | 1 | 350 |
| 2012 | 2 | 2 | 100 |
| 2012 | 3 | 1 | 30 |
| 2012 | 3 | 2 | 780 |
| 2013 | 1 | 1 | 420 |
| 2013 | 1 | 2 | 210 |
| 2013 | 2 | 1 | 368 |
| 2013 | 2 | 2 | 105 |
| 2013 | 3 | 1 | 32 |
| 2013 | 3 | 2 | 819 |
| 2014 | 1 | 1 | 441 |
| 2014 | 1 | 2 | 221 |
| 2014 | 2 | 1 | 386 |
| 2014 | 2 | 2 | 110 |
| 2014 | 3 | 1 | 34 |
| 2014 | 3 | 2 | 860 |

The proforma model will allocate the units in luz\_residential\_unit\_overrides first before proceeding to allocate the remaining units in the region. If DEFM control totals imply an increment of 45,000 new residential units in a given year, and 10,000 of that increment is represented in luz\_residential\_unit\_overrides, UrbanSim will first allocate the 10,000 units to the override LUZs before allocating the remaining 35,000 units to the rest of the LUZs in the region. Any undershooting of the LUZ override targets due to lack of profitable projects is logged, and the unbuilt units are added to the amount to be built in the rest of the region. If desired, the luz\_residential\_unit\_overrides can be used to completely specify LUZ residential unit growth in the region (every LUZ/type combination is represented in the table). In this case, the proforma model acts only as a sub-LUZ residential unit allocation mechanism.

*PECAS - UrbanSim*

The second model integration link to be discussed is that between PECAS and UrbanSim. This linkage refers to the subset of Figure 2 shown below in Figure 6. UrbanSim receives LUZ-level inputs from PECAS: households by type, employment by sector, and price by space type. These inputs should come in the form of three CSV files: households\_by\_luz.csv, employment\_by\_luz.csv, and price\_by\_luz.csv. These three CSVs should be exported to the UrbanSim ‘runs’ directory. When PECAS AA finishes running and exports these tables, UrbanSim will read them and proceed with LUZ-to-parcel allocation.

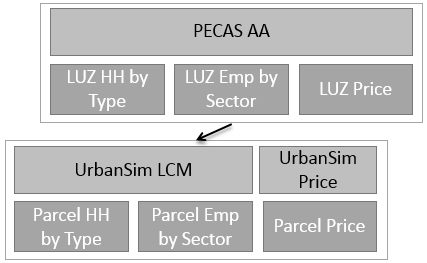


Figure 6- PECAS AA - UrbanSim linkage

Table 9 shows the columns in the LUZ control table for households by type. The first column contains the LUZ ID, the second column indicates household type, and the third column shows the number of households that belong to the corresponding type in the specified LUZ. Each record represents a combination of LUZ and household type, and every LUZ/type combination should be represented unless the combination does not exist.

Table 9- LUZ household controls from PECAS, ‘households\_by\_luz.csv’

|  |  |  |
| --- | --- | --- |
| luz\_id | hh\_type | total\_number\_of\_households |
| 1 | 1 | 600 |
| 1 | 2 | 650 |
| 1 | 3 | 700 |
| 1 | 4 | 750 |
| 1 | 5 | 800 |
| 2 | 1 | 400 |
| 2 | 2 | 500 |
| 2 | 3 | 600 |
| 2 | 4 | 700 |
| 2 | 5 | 800 |
| 3 | 1 | 100 |
| 3 | 2 | 200 |
| 3 | 3 | 300 |
| 3 | 4 | 400 |
| 3 | 5 | 500 |

The household type definitions are defined in pecas\_household\_types.yaml, to be placed in the UrbanSim ‘configs’ directory. YAML is a simple and convenient markdown format that can succinctly express information in a form that is easy to edit by humans. The household type definitions are flexible, as long as the referenced attributes are contained in the synthetic households table used by UrbanSim. If PECAS household type definitions change in the future, just edit this YAML file. The file contains key-value pairs where the key is the household type ID, and the value is the definition for that type. The definition can be any expression that can be resolved by pandas.DataFrame.query (DataFrame query method of the Python pandas library). The IDs defined here correspond to the type ID values found in the hh\_type column of households\_by\_luz.csv.

1: income < 25000

2: (income >= 25000) \* (income < 75000)

3: (income >= 75000) \* (income < 100000)

4: (income >= 100000) \* (income < 150000)

5: income >= 150000

Similarly, see Table 10 for the columns in the LUZ control table for employment by type. The first column contains the LUZ ID, the second column indicates sector ID, and the third column shows the number of jobs that belong to the corresponding sector and LUZ. Each record represents a combination of LUZ and sector, and every LUZ/sector combination should be represented unless the combination does not exist.

The sector\_id should correspond to whatever industrial sector typology SANDAG uses in their PECAS model. The UrbanSim jobs table will use this same sectoral typology so that no translation is necessary. See the Data Memo for additional information on sectors.

Table 10- LUZ employment controls from PECAS, ‘employment\_by\_luz.csv’

|  |  |  |
| --- | --- | --- |
| luz\_id | sector\_id | total\_number\_of\_jobs |
| 1 | 1 | 1200 |
| 1 | 2 | 1300 |
| 1 | 3 | 1400 |
| 1 | 4 | 1500 |
| 1 | 5 | 1600 |
| 2 | 1 | 800 |
| 2 | 2 | 1000 |
| 2 | 3 | 1200 |
| 2 | 4 | 1400 |
| 2 | 5 | 1600 |
| 3 | 1 | 200 |
| 3 | 2 | 400 |
| 3 | 3 | 600 |
| 3 | 4 | 800 |
| 3 | 5 | 1000 |

Table 11 shows the table PECAS should generate annually for price per square-foot by space type and LUZ. This example assumes that there are 3 LUZs and 5 space types. This table connects the PECAS-predicted LUZ price with the UrbanSim-predicted prices at the parcel level. As the PECAS predictions vary, the corresponding UrbanSim-predicted prices will vary in the same direction, all else equal.

Table 11- LUZ price by space type from PECAS, ‘price\_by\_luz.csv’

|  |  |  |
| --- | --- | --- |
| luz\_id | space \_type\_id | price\_per\_sqft |
| 1 | 1 | 30 |
| 1 | 2 | 32.5 |
| 1 | 3 | 35 |
| 1 | 4 | 37.5 |
| 1 | 5 | 40 |
| 2 | 1 | 20 |
| 2 | 2 | 25 |
| 2 | 3 | 30 |
| 2 | 4 | 35 |
| 2 | 5 | 40 |
| 3 | 1 | 5 |
| 3 | 2 | 10 |
| 3 | 3 | 15 |
| 3 | 4 | 20 |
| 3 | 5 | 25 |

PECAS space types are very similar to UrbanSim development types because UrbanSim development types were derived from the space types with only minor modification. A YAML file containing the space type to development type cross-reference will be placed in the UrbanSim ‘configs’ directory.

*UrbanSim - PASEF*

The third model integration link to be described is that between UrbanSim and PASEF. Every travel model year, parcel-level household results from UrbanSim will be summarized at the MGRA level and passed to PASEF. The purpose of this data transfer is so that PASEF can make detailed demographic estimates, which are then used as control variables for PopSyn. PopSyn then generates the year-specific synthetic population needed for the travel model, ABM.

The default option will be for UrbanSim to generate household totals by MGRA. A table will be generated in the form shown in Table 12 and exported to csv: ‘households\_by\_mgraYEAR.csv’, where YEAR is replaced with the simulation year for which the summary is generated.

Table 12- Households by MGRA for PASEF

|  |  |
| --- | --- |
| mgra\_id | households |
| 1 | 90 |
| 2 | 15 |
| 3 | 200 |
| 4 | 170 |
| 5 | 75 |

Alternatively, a simple text file called pasef.yaml can be placed in the UrbanSim ‘configs’ directory. This text file uses YAML to denote PASEF household type, if categories beyond simply the total number of households is desired. In the following example, three household types are defined according to income.

1: income < 25000

2: (income >= 25000) \* (income < 75000)

3: income >= 75000

If pasef.yaml contained the three lines above, UrbanSim would generate a .csv file for PASEF that, instead of just listing total number of households by MGRA, lists number of households by type. See Table 13.

Table 13- Households by type and MGRA for PASEF

|  |  |  |  |
| --- | --- | --- | --- |
| mgra\_id | households\_type1 | households\_type2 | households\_type3 |
| 1 | 40 | 30 | 20 |
| 2 | 4 | 1 | 10 |
| 3 | 120 | 40 | 40 |
| 4 | 65 | 60 | 45 |
| 5 | 35 | 0 | 40 |

If PASEF requires a unique input file format, SANDAG can specify this and provide an example for UrbanSim to match.

*UrbanSim - PECAS*

Next, the integration link between the UrbanSim Proforma in year t and PECAS AA in year t + 1 is described. The UrbanSim Proforma model constructs new built space in the region, and it is the last UrbanSim model to run in each simulation year. At the end of the simulation year, total built space quantities are summarized at the LUZ level and passed to PECAS AA for use in the next simulation year. PECAS AA’s region-to-LUZ demand-side allocation is informed by these built space quantities. For example, PECAS AA cannot allocate more households to an LUZ than there are residential units. LUZ household capacity is determined by the number of residential units, and PECAS respects this. If UrbanSim predicts a large quantity of residential unit construction in a particular LUZ in the previous simulation year, PECAS is able to allocate a correspondingly large number of households to the LUZ. PECAS will view the number of residential units as a binding constraint on the number of households that can be allocated to each LUZ. Similarly, if UrbanSim predicts large increases in non-residential square-footage in a particular LUZ in the previous simulation year, this would inform the subsequent PECAS region-to-LUZ employment allocation.

Built space quantities are sent from UrbanSim to PECAS AA via the FloorspaceI.csv file (a standard PECAS input). UrbanSim will generate an output file that matches the FloorspaceI.csv format and the metrics within it.

*ABM - UrbanSim*

Finally, the link between ABM and UrbanSim is discussed. After each travel model run, updated skims and accessibility metrics are used by the land use models.

UrbanSim will utilize whatever accessibility metrics are conveniently output from the SANDAG ABM: skims, composite measures, and other standard output. These can be read from the SQL Server database.

In the reverse direction, the ABM requires MGRA-level residential unit density measures, and possibly MGRA-level employment information as well, from UrbanSim. In travel model years, UrbanSim will export this information to the SQL Server database used to store travel model input data.

Note that network-based accessibility variables similar to “retail\_employment\_within\_half\_mile” will be calculated internally to UrbanSim using its network aggregation library. This library is described in the next section.

# Network-based Accessibility Variables

One of the core building blocks of UrbanSim is Synthicity's open source accessibility engine, Pandana (“pandana” = pandas + network analysis). While not a model, this accessibility calculator is internal to UrbanSim and worth mentioning here. It will be applied as part of the present project because UrbanSim explanatory variables often utilize it. Pandana performs extremely fast network aggregations. While integral to UrbanSim, Pandana can also be used as a stand-alone tool on any spatial dataset that SANDAG has.

Pandana’s value is its ability to calculate network-based metrics of accessibility extremely quickly and for large datasets. Pandana performs buffer queries along networks, so it generates fine-grain variables that correspond to human perception of accessibility. This is especially valuable for representing pedestrian accessibility and "walkability". An example of a Pandana query might be “number of restaurant buildings within 800 meters along the network”.

Pandana also allows modelers to describe and visualize spatial patterns without relying on zonal boundaries. Pandana queries are not subject to the "modifiable areal unit problem". We can create explanatory variables like “average household income within 400 meters along the network”. Network aggregations avoid arbitrary boundary effects, are easily adjusted, and can better measure spatial variation.

Pandana’s speed (operates on a multi-threaded C++ library and state-of-the-art routing algorithm ‘contraction hierarchies’), pythonic API, ease of installation, integration with pandas, and performance on large datasets (queries for all U.S. local streets in seconds) make for a powerful analytic tool.

Please refer to the Pandana documentation for further information: <http://synthicity.github.io/pandana/>

The Data Memo describes data inputs to Pandana.

# Model Estimation Approach

While specification information is found throughout this document and in the model overview tables at the end of this document, this section discusses the overall approach to be taken with respect to model estimation. The following UrbanSim models in the proposed model system require statistical estimation of parameters: the household location choice model (MNL), the employment location choice model (MNL), and the hedonic model of real estate prices (OLS regression). All estimated coefficients will be generated within UrbanSim. Coefficients are estimated on local SANDAG data and not borrowed. A tabular overview of each model’s specification, including an initial list of independent variables, is found at the end of this document.

Specification of the location choice models in UrbanSim involves deciding which alternative characteristics to be considered in the model (i.e. explanatory variables). It also involves determining whether to stratify the estimation by some characteristic of the agents making location choices (i.e. segmentation). Stratification reflect the hypothesis that different groups of agents have different locational preferences. Both adding/dropping explanatory variables and changing the model stratification are easy to do in the UrbanSim framework. New variables are defined using simple pandas expressions (syntax of the Python pandas library). Each model can be iteratively re-specified and re-estimated in seconds during the process of developing a desired model specification. In UrbanSim, the model estimation process is tied closely to simulation. Estimation and simulation both take place within the same code-base and framework. In a properly configured model, simulation can occur right after estimation.

We have variable categories in mind when starting the specification/estimation process (based on hypotheses in the literature), but the specific variables to use depend on local data, review of estimation results (examining coefficient sign, significances, and measures-of-fit), and an iterative process of trying different specifications. Variable categories we seek to include in location choice model specifications include real estate characteristics, regional accessibility variables, local accessibility variables, and price. For example, a regional accessibility variable we might try is: employment within 30 minutes auto travel time in the A.M. peak period. This variable would be calculated based on skims from the travel model (stored in the UrbanSim travel\_data table, see the Data Memo). A local accessibility variable we might try is whether there is a school within one mile along the local street network, or retail square footage within a half mile. These kinds of variables would be calculated using Pandana. In the location choice models, price, potentially interacted with chooser characteristics such as income, is a key variable. It is hypothesized that ceteris paribus, households/employment will prefer lower prices (i.e. negative coefficient), though it is not uncommon in discrete choice models of housing location to find insignificant or even counter-intuitive signs on price variables due to omitted variables that are correlated with price. We also typically include clustering variables. For example, household income interacted with mean income within 400 meters may be tried as an explanatory variable to identify tendencies for income clustering. Similarly, in the employment location choice model, we may try a variable for the number of jobs of the same sector within one-mile to capture agglomeration economies.

See the tables at the end of this memo for more information on variables that will be explored in the estimation process. See in particular the ‘Independent Variables’ section of each table. These tables also show the initial segmentation to be tried for each model.

# Phase 1 Calibration Strategy

The proforma developer model will be subject to an initial round of calibration in the Phase 1 model.

Prior to calibration, results are evaluated. Uncalibrated built-space predictions are summarized at the LUZ and MSA levels and compared with observed growth in a representative time period. Next, parcel-level results are visually examined. For a selection of parcels that were predicted to develop, inputs to and outputs from the proforma are evaluated for reasonableness. Next, a set of sensitivity tests are run to ensure that the proforma model is sensitive to price, zoning, undevelopable constraints, and fees.

After assessment of uncalibrated results, the proforma model will be calibrated at the MSA level. A script will be prepared to adjust MSA-level prices shifters (constants that scale the price of real estate as perceived by developers) until proforma predictions for the calibration period are within a small range of calibration target values. The targets should be expressed as the share of residential unit growth captured by each MSA in some representative time period (e.g. when the economy was healthy).

After the MSA-level calibration, results will be summarized and evaluated again, and a decision about whether to drop to a smaller geography for further calibration will be made. Any additional calibration will be beyond the Phase 1 scope. In general, the recommended approach is to calibrate as little as possible and only as much as necessary (and using more aggregate geographies first). Extensive calibration can excessively constrain the model and make it less policy-sensitive (and it is not clear whether or how calibrated constants should change in the future).

# Roadmap and Responsibilities

This section lists tasks and deliverables (with estimated effort measured in terms of \*’s). Each task is assigned an owner.

|  |  |  |
| --- | --- | --- |
| TASKS | TIME ESTIMATE | OWNER |
| Initial |  |  |
| Set up GitHub commit hooks to Pivotal | \* | Synthicity |
| Define model linkages | \*\*\* | SANDAG/Synthicity |
| Write Memo 2- Model Development Plan | \*\*\* | Synthicity |
|  |  |  |
| Data and Setup |  |  |
| Determine development types | \* | SANDAG |
| Prepare base-year parcel table | \*\*\* | SANDAG |
| Prepare base-year building table | \*\*\* | SANDAG |
| Assess SANDAG data sets | \*\* | Synthicity |
| Identify other proforma-specific data requirements | \*\* | Synthicity |
| Write Memo 1- Assessment of SANDAG Data Sets | \*\* | Synthicity |
| Prepare base-year synthetic population from PopSyn | \*\* | SANDAG |
| Assign every base-year synthetic household record a building\_id | \*\* | Synthicity |
| Prepare base-year employment by parcel (or MGRA) and sector table | \*\* | SANDAG |
| Synthesize UrbanSim jobs table | \* | Synthicity |
| Assign every base-year synthetic job record a building\_id | \*\* | Synthicity |
| Attach all relevant location\_id’s to parcel table | \*\* | SANDAG |
| Format zoning table | \*\* | SANDAG |
| Format parcel-level fees table | \* | SANDAG |
| Populate proportion\_undevelopable field on parcel table | \* | SANDAG |
| Attach local distance fields to parcel table | \* | SANDAG |
| Prepare building-sqft per job table | \*\*\* | SANDAG |
| Check that base-year vacancy rates implied by data are reasonable | \* | Synthicity |
| Prepare local street network table (initially OSM) | \*\* | Synthicity |
| Implement Pandana so network queries can be used as explanatory variables | \*\* | Synthicity |
| Prepare base-year zone-to-zone skims (AM peak period auto travel times and costs) | \*\* | SANDAG |
| Prepare zonal/MGRA table with travel model-generated logsums | \*\* | SANDAG |
| Prepare other proforma-specific data | \*\* | SANDAG |
| Prepare households-for-estimation table, if recent-movers observed in travel survey. | \*\* | SANDAG |
| Format regional DEFM residential unit targets by year (MF/SF) | \* | SANDAG |
|  |  |  |
| UrbanCanvas |  |  |
| Load building/parcel/zoning tables into UrbanCanvas database | \*\* | Synthicity |
| Prepare pipeline projects table for scheduled development event and import into UrbanCanvas | \*\* | SANDAG/Synthicity |
| Set up UrbanSim-UrbanCanvas integration features | \*\*\* | Synthicity |
| UrbanCanvas feature walk-through/demo | \* | Synthicity |
|  |  |  |
| Model Integration |  |  |
| Define data file format for inputs to PASEF from UrbanSim | \* | SANDAG |
| Define data file format for any ABM inputs from UrbanSim (MGRA residential units, employment, or non-residential-sqft?) | \* | SANDAG |
| Define data file format for LUZ-level residential unit controls with an “other LUZ” category. | \* | Synthicity |
| Define data file format for proforma-generated inputs to PECAS AA in year 2: LUZ space quantities, remaining zone capacities | \* | SANDAG |
| Define data file format for accessibility variables generated from ABM so UrbanSim can use updated accessibilities after each travel model year | \* | Synthicity |
| Ensure that every model system component can be called from the command line and/or python | \* | SANDAG/Synthicity |
| Code python script to orchestrate model components and data transfers. The UrbanSim simulation framework can be used for this orchestration. | \*\*\* | Synthicity |
| Prepare LUZ employment by sector/year control table. Automate its generation from PECAS AA. | \*\* | SANDAG |
| Prepare LUZ households by type/year control table. Automate its generation from PECAS AA. | \*\* | SANDAG |
| Prepare LUZ price by year table. Automate its generation from PECAS AA. | \*\* | SANDAG |
|  |  |  |
| Modeling |  |  |
| Implement existing hedonic specifications in UrbanSim | \*\*\* | Synthicity |
| Flesh out price model specifications | \*\* | Synthicity |
| Implement skeletal location choice models for parcel allocation | \*\*\* | Synthicity |
| Flesh out location choice model specifications | \*\* | Synthicity |
| Implement skeletal proforma with placeholder inputs | \*\*\* | Synthicity |
| Implement proforma with additional SANDAG-specific data inputs | \*\* | Synthicity |
| Implement checks to ensure consistency between PECAS AA’s LUZ household predictions, UDM’s LUZ residential unit predictions, and each LUZ’s vacancy/capacity situation | \*\* | SANDAG/Synthicity |
| Simulation tests and review of uncalibrated results | \*\*\* | SANDAG/Synthicity |
| Prepare calibration targets (observed growth in households, employment, residential\_units, non-residential sqft over some representative time period ) for sub-regional geography. Targets at the Major Statistical Area geography or similar | \*\*\* | SANDAG |
| Initial calibration | \*\*\* | Synthicity |
| Initial validation and basic sensitivity test | \*\*\* | SANDAG |
| Write model estimation and validation report | \*\*\* | Synthicity |
| Write real estate model development report | \*\*\* | Synthicity |
|  |  |  |
| Potential future tasks beyond Phase 1 |  |  |
| Refinement module |  |  |
| Additional calibration/validation/sensitivity-testing |  |  |
| Building type choice model |  |  |
| Demographic microsimulation |  |  |
| Inventory policies that may be included in operational scenarios to ensure policy sensitivities are in place |  |  |
| Mobile home simple allocation model |  |  |
| Vision solver |  |  |
| Price equilibration |  |  |

# MODEL OVERVIEW TABLES

**Table 1: Household Location Choice Model**

|  |  |
| --- | --- |
| ***HLCM*** |  |
| **Agent** | Household |
| **Estimation Dataset** | Recent-mover households, either observed or synthesized |
| **Simulation Dataset** | Synthetic households with building\_id value of -1 (new households) |
| **Location set** | buildings within the appropriate LUZ |
| **Dependent Variable** | building\_id |
| **Model Type** | MNL |
| **Submodels** | By income quartile. Segmentation may be subject to change during the model estimation process and based on discussion with SANDAG. |
| **Independent Variables** | * price interacted with income * income interacted with average income within specified distance * residential building characteristics (e.g. year\_built) * regional accessibility (based on skims and logsums, or any other accessibility metrics that are standard output from the SANDAG ABM. Custom accessibility metrics do not need to be generated by the travel model, though UrbanSim will use travel model output to create configurable explanatory variables like ‘jobs\_within\_30\_min’). * local accessibility (e.g. local street-network based variable, ‘retail\_sqft\_within\_400\_meters’) * neighborhood composition and density (Generally “neighborhood” refers to street network buffers so that specific boundaries do not need to be defined, unless SANDAG has a set of neighborhood boundaries they prefer to use and tag parcels with. Neighborhood buffer examples include “income within 800 meters along the network” or “job-housing mix within 800 meters along the network”. These along-the-network aggregations avoid having to use arbitrary boundaries, are easy to adjust, and are very fast to calculate.) |
| **Sampling alts in estimation** | Yes, unweighted sample of 100 |
| **Sampling alts in simulation** | According to choice\_set/agent ratio. |
| **Simulation method** | Unit-level sampling of vacant residential units without replacement from the universe of vacant units in the LUZ being simulated.  Building\_id of selected residential unit is the new household’s location. |
| **Filter on alternatives** | Building is of type residential and building must be within LUZ |
| **Unplaced during simulation** | Denoted with building\_id of -1 (not NaN) |
| **Segment-specific alt filters** | None |
| **Interaction filters** | None |
| **Agent/observation filter** | None |
| **Max agents in estimation** | 15000 |
| **Underlying module** | urbansim.models.MNLLocationChoiceModel/SegmentedMNLLocationChoiceModel |
| **Config file** | hlcm.yaml |

**Table 2: Employment Location Choice Model**

|  |  |
| --- | --- |
| ***ELCM*** |  |
| **Agent** | Job |
| **Estimation Dataset** | Synthetic job records |
| **Simulation Dataset** | Synthetic jobs with building\_id value of -1 (new jobs) |
| **Location set** | buildings within the appropriate LUZ |
| **Dependent Variable** | building\_id |
| **Model Type** | MNL |
| **Submodels** | By industry sector, and, if available, home-based status. Segmenting the ELCM by sector is the standard in UrbanSim. The industrial sector typology in the UrbanSim jobs table will match that used by PECAS. During estimation of the ELCM, it is possible that sectors will be aggregated into higher-level sectors and fewer segments, but this is to be determined based on sample size and initial results. |
| **Independent Variables** | * price * building characteristics (e.g. development type, year\_built) * agglomeration/clustering (e.g. number of jobs within same sector within one mile) * density (e.g. employment density, population density) * regional accessibility (skim-based or logsums. e.g. population\_within\_20\_minutes) * local accessibility (e.g. local street-network based variable) * composition of households and employment in neighborhood. Neighborhood boundaries need not be defined in advance, we typically use flexible network-based buffers. * If retail-sector, population-seeking variables |
| **Sampling alts in estimation** | Yes, unweighted sample of 100 |
| **Sampling alts in simulation** | According to choice\_set/agent ratio |
| **Simulation method** | Unit-level sampling of vacant job spaces without replacement from the universe of vacant job spaces in the LUZ being simulated.  Building\_id of selected job space is the new household’s location. |
| **Filter on alternatives** | Non-residential-sqft > 0, and building must be within LUZ |
| **Unplaced during simulation** | Denoted with building\_id of -1 (not NaN) |
| **Segment-specific alt filters** | None |
| **Interaction filters** | None |
| **Agent/observation filter** | None |
| **Max agents in estimation** | 15000 |
| **Notes** | Job spaces calculated using a building\_sqft\_per\_job table-  ideally by zone and building space  type |
| **Underlying module** | urbansim.models.MNLLocationChoiceModel/SegmentedMNLLocationChoiceModel |
| **Config file** | elcm.yaml |

**Table 3: Real Estate Price Model- Residential Submodels**

|  |  |
| --- | --- |
| ***Residential REPM*** |  |
| **Estimation Dataset** | Buildings |
| **Simulation Dataset** | Buildings |
| **Dependent Variable** | Log of price per residential sqft |
| **Model Type** | OLS Regression |
| **Submodels** | By residential development type. Separate regression equation estimated for reach. |
| **Independent Variables** | * Constant * LUZ price from PECAS (PECAS-generated measure of relative attractiveness of corresponding space type in the LUZ) * distance to local amenities/disamenities (e.g. the beach) * building characteristics (e.g. year\_built) * regional accessibility (skim-based or logsums. e.g. employment\_within\_30\_minutes) * neighborhood characteristics-   density, local accessibility, composition. Neighborhood boundaries need not be defined in advance, we typically use flexible network-based buffers. * Possible: vacancy rate within 200m/400m * Possible (only as needed): geographic dummies for local fixed effects |
| **Simulation method** | * When updating prices in simulation, the predicted values are exponentiated and stored directly as price instead of log(price). |
| **Observation filter** | Filter out invalid records and outliers |
| **Underlying module** | urbansim.models.RegressionModel/SegmentedRegressionModel |
| **Config file** | rsh.yaml |

**Table 4: Real Estate Price Model- Non-residential Submodels**

|  |  |
| --- | --- |
| ***Nonresidential REPM*** |  |
| **Estimation Dataset** | Buildings |
| **Simulation Dataset** | Buildings |
| **Dependent Variable** | Log of price per non-residential sqft |
| **Model Type** | OLS Regression |
| **Submodels** | By non-residential development type. Separate regression equation estimated for reach. |
| **Independent Variables** | * Constant * LUZ price from PECAS (PECAS-generated measure of relative attractiveness of corresponding space type in the LUZ) * distance to local amenities/disamenities * building characteristics (e.g. year\_built) * regional accessibility (skim-based or logsums. e.g. population\_within\_20\_minutes) * neighborhood characteristics-   density, local accessibility, composition. Neighborhood boundaries need not be defined in advance, we typically use flexible network-based buffers. * Possible: vacancy rate within 200m/400m * Possible (only as needed): geographic dummies for local fixed effects |
| **Simulation method** | * When updating prices in simulation, the predicted values are exponentiated and stored directly as price instead of log(price). |
| **Observation filter** | Filter out invalid records and outliers |
| **Underlying module** | urbansim.models.RegressionModel/SegmentedRegressionModel |
| **Config file** | nrh.yaml |

**Table 5: Developer Model**

|  |  |
| --- | --- |
| ***Proforma Real Estate Supply Model*** |  |
| **Model Type** | Rule-based profitability calculator |
| **Description** | Computes development feasibility (profitability subject to zoning), and selects which feasible buildings to construct to match target built space totals at specified geography (region or LUZ) |
| **Inputs** | Zoning, costs, prices, building\_sqft\_per\_job, parcel characteristics |
| **3 Steps:** | 1.)  Feasibility:  computes development feasibility (the set of profitable projects accounting for development constraints) |
|  | 2.)  Residential Developer:  builds actual residential buildings to meet residential demand |
|  | 3.)  Nonresidential developer:  builds actual nonresidential buildings to meet nonresidential demand |
| **Override mechanism** | Optional overrides in certain LUZs where SANDAG will assert residential unit totals by type and year. In the case that an override is specified for an LUZ, the UrbanSim proforma model will respect this target and allocate this quantity of residential units to parcels within the LUZ, subject to the availability of profitable projects. |
| **Underlying modules** | urbansim.developer.sqftproforma AND urbansim.developer.developer |

**Table 6: Scheduled Development Events**

|  |  |
| --- | --- |
| ***Scheduled development events*** |  |
| **Model Type** | Rule-based model |
| **Input table** | From UrbanCanvas development projects |
| **Notes** | * This model serves the same function as the SANDAG Site Spec * Inserts new buildings (known pipeline projects) in specified simulation years. * Different treatment of committed and uncommitted projects. * Can also be used as a supply-side refinement tool * Specific upzoning events can also be inputted via UrbanCanvas (note that upzoning events are determined by the analyst and are not modeled by UrbanSim) |