
How To Run the Eugene-Springfield Model

Center for Urban Simulation and Policy Analysis
University of Washington

July 3, 2007

Daniel J. Evans School of Public Affairs
University of Washington
Seattle, Washington 98195
USA
Email: info@urbansim.org
Web Site: www.urbansim.org

Introduction

This tutorial covers the essentials of running an UrbanSim simulation. We do not cover some complications that are not essential to running UrbanSim, like setting up a database server, creating a base year database, and estimating the model parameters. These omitted items are part of the larger process of preparing to use UrbanSim in a new area, and will be covered elsewhere.

For the simulation, we use the 1980 base year data for the Eugene-Springfield, Oregon area. This is provided in a non-database format that we refer to as a “cache”, since it is a copy of the required data from the associated MySQL database. The base year data includes all the model parameters. The entire process of downloading and installing the system, and getting a simulation running on a local computer should require no more than 20 minutes. Since the model runs quickly on this data, one should be able to compute and visualize indicators from a simulation over 2 years within another 20 minutes.

We have tested this tutorial on Windows with the Enthought Python Edition installation, as well as on Linux and Mac with the appropriate Enthought packages.

This tutorial is the first of what we expect to become a fairly extensive set of tutorials to assist new and more experienced users in taking advantage of the functionality in the UrbanSim model system, using the new Open Platform for Urban Simulation (OPUS).

Install Software and Download Sample Data

If you have not already done so, install Opus and UrbanSim, along with any needed supporting software. Directions for the current version of the code are at <http://www.urbansim.org/docs/installation/>. If you are using a (perhaps older) stable release rather than the current version fresh out of the subversion repository, the table of releases at <http://www.urbansim.org/download/> includes a link to the installation instructions corresponding to that particular version.

Next, download the Eugene 1980 baseyear cache zip file by following the appropriate link on the download page <http://www.urbansim.org/download/>. If you are using the source code for a stable release, there will be a link to the Eugene 1980 baseyear cache file that works with this version in the table of stable release versions. Alternatively, if

you are getting the latest version of the code from the subversion repository, there is a link in the “Current Source Code” section. (We frequently update the Opus and UrbanSim source code, but the sample database doesn’t change very often — so the stable release version and latest version will likely be the same.)

Unzip the file into a cache directory of your choice, such as `c:\urbansim_cache` on Windows, or `/Users/yourname/urbansim_cache` on the Mac. This example Eugene-Springfield database contains all of the data that UrbanSim needs to run a simulation of Eugene starting for simulated year 1980.

Running a Simulation

In this part of the tutorial, you will execute a two-year simulation run of the Eugene-Springfield metro area.

1. First, open a command window (yes, that archaic interface to the computer where you actually have to *type commands*). On Windows, go to the ‘Start’ menu, choosing ‘Run...’, typing “cmd”, and clicking ‘OK’. On the Mac, open a terminal window by picking “terminal” from Applications/Utilities in the Finder. On Linux, open a shell.
2. Next, change directory to the workspace subdirectory containing the Eugene tools, e.g. ‘C:\workspace\eugene\tools’ (modify this appropriately if your workspace has a different name). For example, on Windows type:

```
cd C:\workspace\eugene\tools
```

On the Mac, type:

```
cd /Users/yourname/workspace/eugene/tools
```

replacing `yourname` with your login name on your machine.

3. Now enter the following command at the command line:

```
python run_simulation_on_baseyear_cache.py
```

This will launch a very simple graphical user interface that will allow you to run a test simulation. *Note that the first time you launch one of these graphical user interfaces, it may take up to a minute to load. Subsequent launches are much faster.*

If you don’t see the graphical user interface, check your task bar, as the application may be hidden behind another window.

You may see a warning message like this:

```
DeprecationWarning: ScipyTest is now called NumpyTest; please update your code
test = ScipyTest().test
```

Just ignore this warning — it is a problem in the `scipy` package, which hopefully will be fixed in a later release of `scipy`.

4. In the “Get baseyear data from this cache directory” field , enter the path to the location where you unzipped the baseyear cache, above, e.g. ‘C:\urbansim_cache\eugene_1980_baseyear_cache\’ on Windows, or ‘/Users/yourname/urbansim_cache/eugene_1980_baseyear_cache/’ on the Mac.

The “Use this configuration” entry is pre-set to use the default baseline configuration in the eugene Opus package (`eugene.configs.baseline`), so you don’t have to modify that now. If you had multiple scenarios configured, this is where you could set which one to run. Note that the name is in parts, separated by periods. These parts correspond with the system path to the actual module that has the configuration. Don’t believe me? Look in the ‘eugene\configs’ directory.

5. In the “Create the output cache in this directory” field, set the output directory to a directory into which you want the simulation results to be written, e.g. ‘C:\urbansim_cache’. UrbanSim will create a new subdirectory at this location, named with an embedded date/time, such as ‘run_2006_11_15_15_12’, so that you can know when it was started and to reduce the chances of accidentally over-writing simulation results from earlier runs.
6. Leave the “Number of years to run” at 2.
7. To start the simulation, press the “OK” button. The tool will close, and UrbanSim will start running. While the simulation runs, a series of messages are sent to the command window to provide feedback on what is happening, but this will generally only need to be consulted if you are diagnosing a problem. The simulation takes about 3 1/2 minutes per year on a computer using a 3.2 GHz Intel®Pentium®4 processor, and about the same on a 2Gh Pentium®M. On a 3Ghz Xeon®64-bit processor running Linux (Fedora Core 6) it runs in 35 seconds per year.

If you want to stop the simulation while it is running, open the command prompt window and type Ctrl-C several times. When the simulation is finished, it will say something like:

```
Done running simulation for years 1981 thru 1982
```

That’s it! You have now installed Opus and UrbanSim, a full application for Eugene-Springfield, Oregon, and run a simulation using a default scenario.

The results are all stored on your computer in a subdirectory of the output cache, with a name that indicates when the output was created. If you run this multiple times, it will create a new directory for each run. The contents of these directories are the complete set of “primary” variables and values predicted for all objects in the model, from which UrbanSim can recompute any “computed” variables defined by an Opus . However, these results are in binary files (arrays) that are not easy to read directly. Now we move on to how to examine the results by creating indicators from them.

Computing an Indicator

We have now run the simulation, so let’s explore the results. UrbanSim supports the concept of “indicators” that are useful representations of dataset attributes. Conceptually, an indicator is a dataset attribute that is presented in a useful manner, such as a map, comma-separated-value file, or a tab-delimited file for use by another program.

First, let’s produce a simple map (don’t get too excited, it is only an image map – we’re working on fancier maps with ArcGIS and other systems too, but this is a quick visualizer that doesn’t have all the overhead of a GIS).

1. In your command window, make sure you are in the ‘eugene\tools’ directory. Then type the following command to open a tool that will allow us to specify and then view an indicator:

```
python create_indicator.py
```

2. In the “Cache directory” field, enter the location of the cache directory created by the simulation, e.g. ‘C:\urbansim_cache\run_2006_11_15_15_12’. This will be a directory in the output directory you specified when starting the simulation.

3. Leave the “Compare to another cache directory” unchecked. This option lets you compare results from two different simulations.
4. Select “Matplotlib map” from Type drop down menu. You can see there are currently 4 other indicator types available: Comma-separated table, Tab-separated table, Chart, and Dbf export. We’ll try tab-delimited table in a moment.
5. In the “Attribute” field, enter:

```
urbansim.gridcell.population
```

This entry contains the “Opus path” for a variable that shows the population within each gridcell. More specifically, this Opus path specifies the location of a Python file defining this gridcell : it is in the *urbansim* Opus package, is defined for the `gridcell` dataset, and is located in the Python file named ‘`population.py`’. You can find the directory that holds the code for the *urbansim* package in your workspace directory.

6. Leave the optional “Name” entry empty. If entered, it is used as the name for the created indicator (and also determines the filename of the resulting indicator). We’ll let the program use the default name (“population” in this case).
7. Type in “gridcell” in the “Dataset” entry. This is consistent with the attribute we’re going to create indicator for.
8. Type in 1982 in the “Year(s)” entry. Leave the optional Scale entry as it is.
9. Press “Run Request” to generate this indicator.
10. The tool will then compute the indicators. When the “View results” button becomes active, it means the computation has finished. Press this button. A web browser will be launched and page loaded with a link to the requested indicator.
11. Click on the “1982” link to see the map. This simple map was produced by matplotlib, a Python graphing and mapping package. More sophisticated maps can be generated by importing the data into other tools, such as ArcMap.
12. Now let’s produce a tab-delimited file for a different attribute. Switch back to the indicators interface, change the “Type” to “Tab-delimited table”, and change the “Attribute” field to be:

```
urbansim.zone.residential_units
```

(Note that `zone` refers to a traffic analysis zone.)

13. Type in “zone” for the “Dataset” field and a single year for which you want to generate the data (e.g. 1981 or 1982) in the “Year(s)” entry, and press “Run request” to generate the data.
14. When the “View results” button is active again, press it to take you to the resulting web page which will now also contain a link to the tab file. In Windows, the default viewer for a tab file is often Excel, which will open your file automatically when you click on the link in this web page. Alternatively, you can of course load it into any software package that can read an ASCII, tab-delimited file — even a text editor.
15. Just for fun, change the selection in the tool from “Tab-delimited table (*.tab)” to “Matplotlib map (*.png)”, and run the indicator again. Note that the map is a gridcell map but shows the data by traffic analysis zone. The map shows “jaggies” because of the resolution of the gridcells (150 x 150 meters) at this scale.

See <http://www.urbansim.org/wiki/external/index.php/Indicators> for a partial list of indicators that are known to work on the Eugene-Springfield data. Also, note that maps only work for indicators associated with a geography. In the case of Eugene, this includes indicators for `gridcell` and `zone`. You cannot, for instance, create a map for `urbansim.household.is_minority`, since `household` is not a geography. You can, however, define and use a variable that links this information with a geography, such as done by `urbansim.gridcell.number_of_minority_households`.

16. Press “Close” to exit this tool.

Note that you can examine any of the data in the `urbansim_cache` by using these indicator tools. You can also export the data for any year to a MySQL database, or to tab-delimited or comma separated ASCII files to be able to explore them in other software systems. If you want to export the cache data, either from the base year database or from the simulation output, change directory to the `'eugene\tools'` directory and run one of the three tools we have provided for this purpose to export the output in the format you prefer:

```
python export_cache_to_mysql.py
```

```
python export_cache_to_tab.py
```

```
python export_cache_to_csv.py
```

These tools open a simple tool, which at this point you should be able to figure out from the preceding parts of the tutorial. If you are uncertain about the function of an edit window, click on its label for a bit of help. Enjoy!

Closing Comments

At this point, you have run a simulation and generated and visualized a couple of indicators from the simulated results. You may have also exported the UrbanSim database for a year and examined it in another system. This tutorial has just scratched the surface of what you can do with UrbanSim, of course. For more information, see the accompanying Reference Manual and User Guide. In addition, we plan to expand the content in this tutorial and add additional tutorials as we get time and as the functionality develops. We would welcome user contributed tutorials also!

If you have any particular requests for documentation, let us know by emailing the users@urbansim.org email distribution list.

Finally, note that while the above tools are currently located in the eugene Opus package, they will work just as well on any other UrbanSim data set, as long as you have updated that data and code as noted in the `'docs\release_notes.html'` file.

Acknowledgements

This tutorial was put together by the UrbanSim project team in response to various requests for a sample application that would allow quickly installing UrbanSim, running an application, and examining the data, and has been tested by several users.

A special thanks is in order to Bud Reiff, Bob Denouden and others at the Lane Council of Governments for their generosity in sharing the data we use in this tutorial. This research has been funded in part by grants from the National Science Foundation (EIA-0121326 and IIS-0534094), and in part by a grant from the Environmental Protection Agency (R831837). We'd like to thank several users as well for their financial support of the development of UrbanSim and OPUS.