SoundCasting at PSRC:
Activity-Based Model Development with Emme

Brice Nichols
Suzanne Childress
Stefan Coe

Puget Sound Regional Council

INRO Conference, Seattle
October 2014
Overview

• **SoundCast** is:
  – PSRC’s nearly-complete activity-based model
  – A model “package” including DaySim activity generation models as well as assignment and skimming with Emme

• In this presentation:
  – SoundCast model structure
  – Managing model development and code
  – Interfacing DaySim models with Emme
  – Lessons learned in Emme and Python
Development and Philosophy

• SoundCast is **open source**, to build:
  – Transparency
    • To public, partner agencies
  – Accessibility
    • for model users
  – Developer community
    • Of travel modelers sharing tips, techniques, results, code

• We encourage other Emme-based modelers to follow this approach!
Development and Philosophy (2)

• Version control and open-source hosting with GitHub
  – Helps manage internal code development
  – Controls model releases
    • Easier to share, run, and manage model versions
    • Helps keep track of versions used for analyses
Model Structure

Inputs and Configuration

Daysim Demand

Trips

EMME Assignment + Skimming

Skims

Converged?

Summarize
Python Controller Scripts

• Python scripts control the “arrows” in the model structure
  – Set run iterations, convergence, process flow
  – Initializes directories, projects, banks
  – Controls demand models
  – Transfers data from demand model to Emme for assignment and skimming
  – Tests for convergence
  – Summarizes model results
• Runs are managed by an “input configuration” file

• Switch on/off model processes like:
  – Basic directory setups
  – Use seed trips, run assignment only
  – Run specific sub-models like truck and external trips

• Holds variables and assumptions
  – Values that might change across analyses and important to quickly validate later
HDF5 Data handoffs

Daysim Demand → HDF5 Trips → EMME Assignment + Skimming

HDF5 Skims → Nope.

Converged? (Nope. → Yep.)

Summarize
HDF5 is a more general form of Open Matrix (OMX) File Format

https://sites.google.com/site/openmodeldata/
Trips to EMME

Daysim Demand → HDF5 Trips → EMME Assignment + Skimming

HDF5 Skims

Converged?

Nope.
Yep.

Summarize
### Populating Trip Tables

#### By Mode, Time

<table>
<thead>
<tr>
<th>ID</th>
<th>Time</th>
<th>Mode</th>
<th>Origin Zone</th>
<th>Destination Zone</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>567</td>
<td>SOV</td>
<td>123</td>
<td>456</td>
</tr>
<tr>
<td>2</td>
<td>890</td>
<td>Walk</td>
<td>789</td>
<td>1011</td>
</tr>
</tbody>
</table>

The table above shows the ID, Time, Mode, Origin Zone, and Destination Zone for each trip. The diagram illustrates how the data can be organized by mode and time, with a highlighted cell indicating a specific trip pattern.
if vot < 15:
    vot_category = 1
elif vot < 25:
    vot_category = 2
else:
    vot_category = 3
Map Minutes to 12 Periods

Daysim

Assignment
Feedback convergence

Daysim Demand → Trips → EMME Assignment + Skimming

Skims → Trips

Converged? 🚫

Summarize
Convergence Test

Relative Trip-Weighted Average
Absolute Skim Change
By John Gibb, Of DKS:

\[
\frac{\sum_{ij} q|t_a - t_i|}{\sum_{ij} qt_a}
\]

\(q\) = demand (trip table)
\(t_a\) = cumulative average skim travel time
\(t_i\) = this iteration travel time
When do we stop?

Configurable parameters:

A list of travel time skims sent in for comparison

STOP_THRESHOLD = 0.025
Overview

Assignment/Skimming Process

- 12 Time Periods
- 21 User Classes
- Run Auto Assignment for each Time Period
- Transit for 5 Time Periods
- Skim for Time and Cost for all Time Periods
- Skim for Distance for Two Time Periods
- Run almost everything in Parallel
Populating Demand Matrices Cont.

• Get the Origin and Destination TAZ's for the trip
• Add the trip to the in-memory numpy 'svnt1' trip table
• Once through all trips, write out numpy matrices to emme.

Emme Prep

Create 12 TOD Banks
Create 12 TOD Projects
Import Networks

Emme Assignments & Skimming

- Define Matrices
- Import Trips
- Run Highway Assignments
- Skim for Time/Distance/Cost
- Transit Assignment
- Transit Skims
- Check Skim Convergence
- Export Skims to HDF5

Seed Trips

Demand Models

Daysim
Trucks
External, GQs, SGs
Use Flexible/Extensible Code Design

class EmmeProject:
    def __init__(self, filepath):
        self.desktop = app.start_dedicated(True, "cth", filepath)
        self.m = _m.Modeller(self.desktop)
        #delete locki:
        self.m.emmebank.dispose()
        pathlist = filepath.split("/")
        self.fullpath = filepath
        self.filename = pathlist.pop()
        self.dir = "/".join(pathlist) + "/
        self.bank = self.m.emmebank
        self.tod = self.bank.title
        self.current_scenario = list(self.bank.scenarios())[0]
        self.data_explorer = self.desktop.data_explorer()

    def create_extra_attribute(self, type, name, description, overwrite):
        NAMESPACE="inro.eme.data.extra_attribute.create_extra_attribute"
        process = self.m.tool(NAMESPACE)
        process(extra_attribute_type=type,
          extra_attribute_name= name,
          extra_attribute_description= description,
          overwrite=overwrite)
```
from EmmeProject import *

# EmmeProject is a custom class
# Provides access to Modeller Object
# And methods and properties used by Soundcast

# Open Project
my_project = EmmeProject('c:/soundcast/projects/7to8.emp')

# Create link extra attribute to store the auto equivalent of bus vehicles
my_project.create_extra_attribute('LINK', '@trnv3',
    'Transit Vehicles', True)
```
Store inputs in Config file or JSON files

```python
import json

def json_to_dictionary(dict_file):
    my_dictionary = json.load(open(dict_file))
    return(my_dictionary)

file_path = 'z:/soundcat2/inputs/skim_params/user_classes.json'
my_user_classes = json_to_dictionary(file_path)
print(my_user_classes['Highway'][0]['Name'])
print(my_user_classes['Highway'][0]['Value of Time'])

>>> svt11
1200
>>> |
```
Parting Thoughts

• Use a Software Development Paradigm
• Emme Python API’s are intuitive and easy to program against.
• Python is (relatively) easy to learn and powerful
• Check out our code on Github: 
  https://github.com/psrc/soundcast