SANDAG: ActivitySim Visualizer Assessment

Overview

This document describes SANDAG's assessment of the ActivitySim Visualizer and the process undertaken to test its extensibility to SANDAG implementations of ActivitySim. The assessed visualizer can be found here. This assessment will be driven by the following key items:

- ***Organization***
- ***Look and Feel***
- ***Extensibility***

Assessment

Organization

This item assesses the organization and clarity of the workflow needed to set up and run the visualizer for an ActivitySim scenario. This assessment is made from the perspective of a user who is already familiar with ActivitySim.

In summary, to set up and run the visualizer, users need to:

1. Add the *summarize* step to the model run list
2. Add the *summarize.yaml*, *summarize.csv*, and *summarize_preprocessor.csv* files to a *configs* directory
3. Create a *summarize* subdirectory within the *output* directory and add any necessary dashboard configuration files
4. Install and run SimWrapper on a completed ActivitySim scenario

Steps 1 & 2

These are standard procedures when adding a new step to ActivitySim. Therefore, it is clear how the *summarize* component of the visualizer is enabled for an ActivitySim scenario.

Step 3

In ActivitySim, subdirectories within the *output* directory (log, trace, cache) organize model outputs. Likewise, the created *summarize* subdirectory organizes outputs from the *summarize* step. Therefore, this addition is in line with the current ActivitySim structure.

However, the dashboard configuration files included in the *summarize* subdirectory are not read during an ActivitySim model run. All other files in a subdirectory, such as the caching file *chunk_cache.csv*, are used during a model run. Despite this difference, to simplify the connection of ActivitySim and SimWrapper, it is a reasonable setup.

Step 4

This step adds a layer of user involvement to the ActivitySim workflow. Ideally, the visualization/dashboard would be built during a model run and output as a separate file (e.g.,

HTML); this would help streamline an end-to-end execution of ActivitySim. However, user involvement is minimal as the SimWrapper install happens only once and a single command needs to be executed to run SimWrapper. Also, the aim of the visualizer may have been to strictly wrap around ActivitySim, since SimWrapper is a standalone package. This setup offers flexibility to the user to carry out tasks such as hosting dashboards on a server.

Look and Feel

This item assesses the design and user experience of the visualization software used by the visualizer, SimWrapper. A SANDAG version of the example MTC SimWrapper dashboard was used for the assessment. This version uses the same dashboard configuration files as in the MTC example but with SANDAG 1-Zone example results.

Design

SimWrapper dashboards are simple in design yet very configurable. The row and tab-based structure of the dashboard allows users to neatly organize different reports, charts, and objects. The ability to toggle light/dark mode, split panels, and save charts are nice features.

For modeling purposes, comparing different scenarios within the same dashboard tab would be needed. The current implementation only supports scenario comparison across panels which is not ideal and ultimately does not scale well when comparing more than two scenarios. Such capability could also enable the comparison of scenario results against observed data for validation purposes.

User Experience

SimWrapper dashboards are easy to generate from a completed ActivitySim scenario and intuitive to use. Minimal documentation is needed to bring users up to speed on navigating the dashboards.

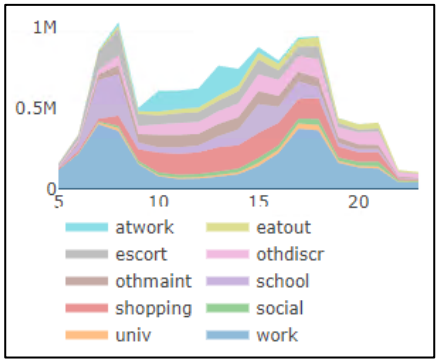
There are, however, some observations that were made while testing SimWrapper:

- *Browser Support* - Dashboards generate when using Chrome but not Internet Explorer and possibly other browsers.
- ***Performance*** (key observation)- Dashboard performance varies heavily across different machines. Dashboards are unusable on machines that SANDAG uses to carry out model runs but are usable on lighter machines with GPUs. Additionally, dashboards may become slow, unresponsive, or even timeout if navigating maps, working with different panels, or toggling between tabs. If navigating between tabs, the user must wait for the charts/maps to be generated every time even if a tab was previously loaded. This may raise issues when developing more complex dashboards and ultimately presenting them to stakeholders.
- *Error Logging* - If there is an underlying issue with a chart (e.g., wrong table name), it may endlessly load without throwing any error.

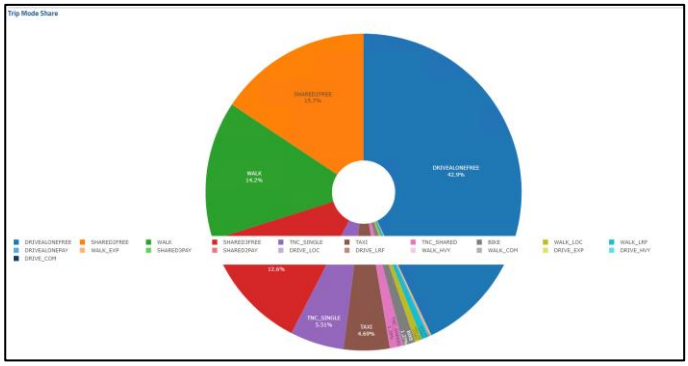
Additionally, the following bugs were encountered while navigating through the dashboard:

- Maps are centered on San Francisco by default despite the underlying GeoJSON being for San Diego -- manual dragging was required to view San Diego maps.
- Chart legend can overlap with charts in default or expanded view:

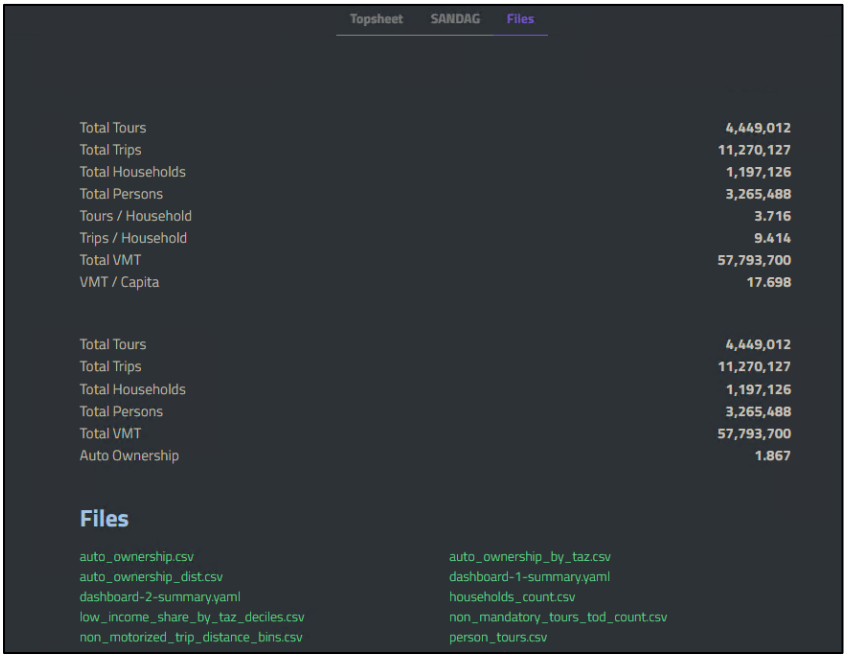
Default view



Expanded view



- Statistics tables feed into the *Files* tab:



Extensibility

This item assesses the extensibility of the visualizer to different ActivitySim examples, and the work/understanding required to develop region-specific output summaries and dashboards. For this assessment, a simple dashboard was created for the SANDAG 1 and 3-Zone examples. These custom dashboards are based on the MTC dashboard.

SANDAG 1-Zone Dashboard

To create a custom dashboard for the SANDAG 1-Zone example:

- Generate a TAZ-based GeoJSON file using GeoPandas

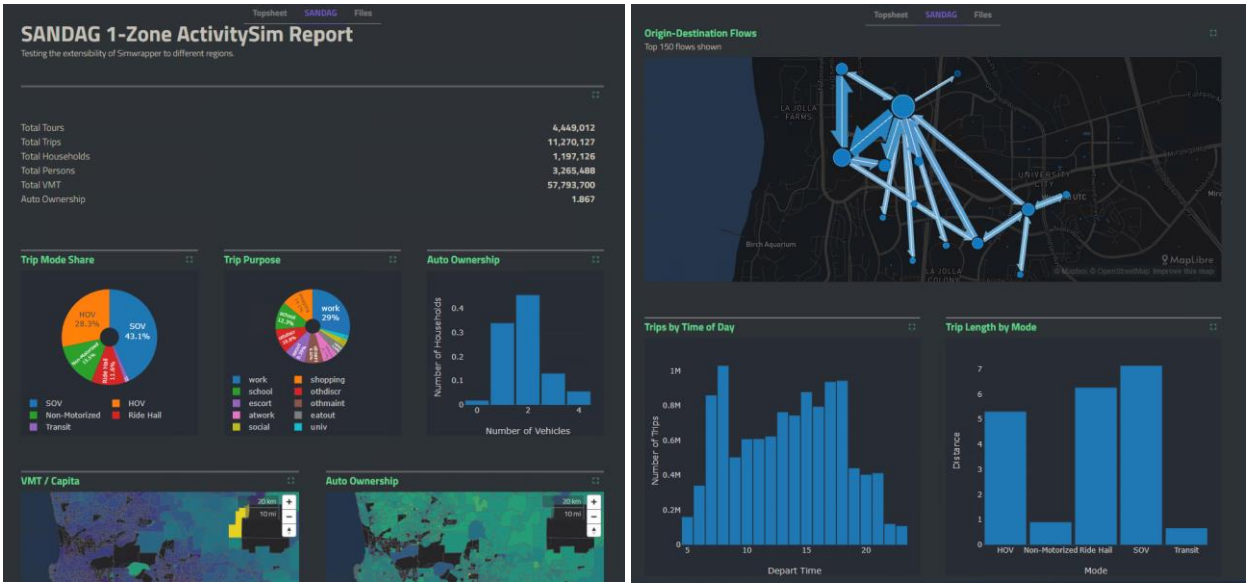
- Edit the *summarize* step files in the *config* directory:

File	Update(s)	Notes
summarize_preprocessor.csv	Single expression added to compute total trip distances.	Requires understanding of the underlying data in the pipeline and how to write simple Python/Pandas expressions.
summarize.csv	Various expressions added to compute auto ownership, trip length, and trip O-D summaries.	Requires deeper understanding of Python/Pandas.
summarize.yaml	Left unchanged.	Requires understanding of the underlying visualizer functions.

- Edit the dashboard files in the *summarize* directory:

File	Update(s)	Notes
topsheet.yaml	Regional auto ownership added to statistics table.	Requires understanding of the structure of the file and the SimWrapper API
dashboard-*.yaml	Title change, AO distribution chart, AO by TAZ map, trip length by mode chart, trips by time of day chart, and trip length by mode chart added.	Requires understanding of the structure of the file and the SimWrapper API

After making the noted updates, the following dashboard was generated:



Since the SANDAG 1-Zone example is based on the 1-Zone MTC example (the same variables and skims), creating dashboard charts and maps was simple; building a dashboard from scratch would be more involved. Ultimately, this test shows that custom dashboards can be created for different ActivitySim examples and that the pipeline/procedures that have been developed work.

Additionally, the following improvements would be useful for the dashboard creation process:

- Ability to create temporary variables in *summarize.csv* to prevent long one-liners
- Ability for the user to specify an axis ordering (as in the Trip Length by Mode chart)
- Legends/Colormaps for maps

SANDAG 3-Zone Dashboard

The aim of building a dashboard for the SANDAG 3-Zone example was to assess if the SimWrapper works for different zone systems. This is necessary as SimWrapper has only been tested on a 1-Zone system.

To create a custom dashboard for the SANDAG 3-Zone example:

- Generate a MAZ-based GeoJSON file using GeoPandas
- Edit the *summarize* step files in the *config* directory:
    - Files from the 1-Zone example were reused with minimal changes
- Edit the dashboard files in the *summarize* directory:
    - Files from the 1-Zone example were reused with minimal changes

The resulting dashboard is similar to the 1-Zone dashboard shown in the previous section. However, transit-based summaries were unable to be produced due to the *summarize* preprocessor (summarize_preprocessor.csv) not having access to the underlying TAP skims used to model transit travel. The summarizer only supports TAZ skims as in the 1 or 2-Zone examples. Further developments will be required for the visualizer to be fully compatible with different zone systems.

Additionally, the preprocessor assumes that the user will be making modifications to the table *trips_merged* only and not *persons_merged*. If any modification to *persons_merged* is needed, the user will be unable to do so through the preprocessor.

Key Takeaways

The organization, look and feel, and extensibility of the visualizer have been assessed in detail.

The key takeaways are the following:

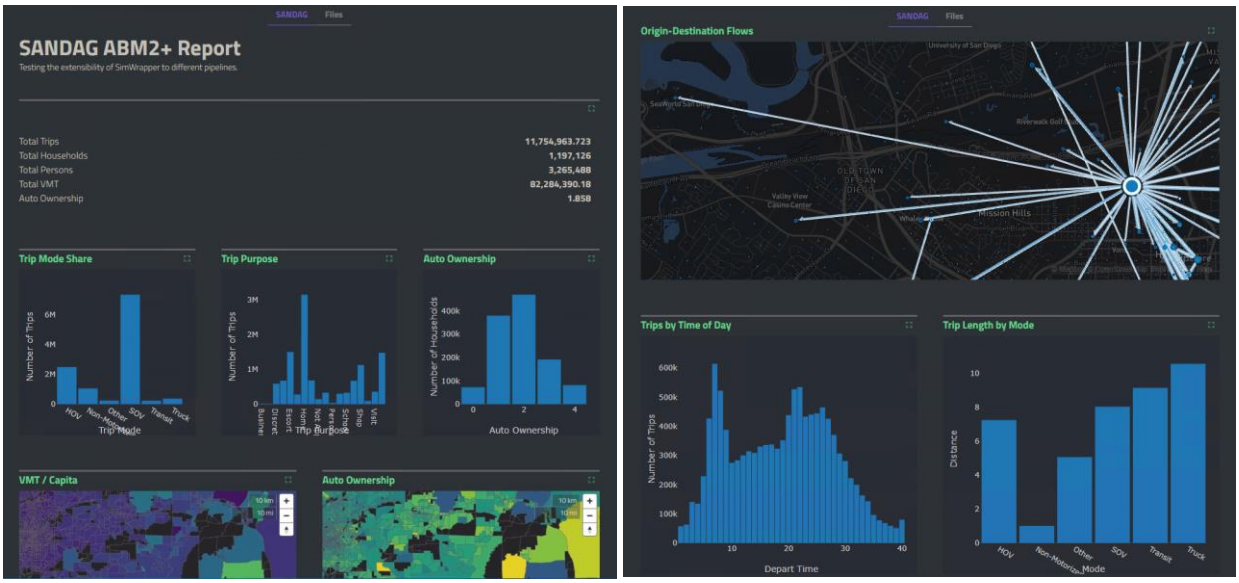
- Visualizer components are well organized
- A simple and working procedure exists to build dashboards from ActivitySim scenarios
- Visualizer dashboards are well designed but do contain bugs
- **Performance-related issues**
- Visualizer is extensible to different ActivitySim scenarios
- Data availability issues for the *summarize* preprocessor

Additional Assessments

Applications beyond ActivitySim

To assess SimWrapper's applications beyond ActivitySim, SimWrapper was used to visualize SANDAG's ABM2+ outputs. For simplicity, the same dashboard as in the *Extensibility* section, with slight modifications, was produced using ABM2+ results.

Like ActivitySim, ABM2+ outputs are large and require an added data pipeline to summarize those outputs, as was done through the *summarize* step in ActivitySim. Therefore, a data pipeline was created to transform ABM2+ outputs into summary tables. These summary tables were used to create the resulting dashboard:



This test proves that SimWrapper can be used for applications beyond ActivitySim. SimWrapper only requires summary tables and dashboard configuration files to create a dashboard; there are no dependencies on prior processes, such as ActivitySim or ABM2+. This offers flexibility and, with possible performance improvements, support for many applications.