

VISUALIZATION FOR TRAVEL DEMAND MODEL OUTPUT

Authors:

Alex Bettinardi, Oregon Department of Transportation
Rolf Moeckel, Parsons Brinckerhoff
Erin Wardell, Parsons Brinckerhoff

ABSTRACT

As public agencies strive to create more accurate travel demand and land-use models, they must also find ways to visualize the model output. The visualization of results is essential in model development to assess how the model is performing. Once a model is out of the development process and into applications, visualizations serve a different purpose. They must communicate the story that the model is telling, deftly showing how the model answers complex policy questions. The range of visualizations required to show these answers includes plots, tables, and maps. This paper will outline a visualization plan developed for the Oregon Department of Transportation Statewide Integrated Model (SWIM).

INTRODUCTION

As public agencies strive to create more accurate travel demand and land-use models, they must also find ways to visualize the model output. The visualization of results is essential in model development to assess how the model is performing. Once a model is out of the development process and into applications, visualizations serve a different purpose. They must communicate the story that the model is telling, quickly illustrating the comparison of complex policy questions.

Models are not the only public agency project that can benefit from a coordinated approach to data processing for visualization. Agencies have an enormous need to quickly process and visualize data to support plan analysis, inform public meetings, and produce reports. Budget restrictions severely limit their ability to purchase commercial software or attend necessary training. Furthermore, commercially available products often do not serve ideally the purpose of model visualization.

This paper describes the visualization process developed by the consultant team of Parsons Brinckerhoff (PB) along with the Oregon Department of Transportation (ODOT), and is specific to the ODOT Statewide Integrated Model (SWIM). This process can be adapted to any project that has large amount of data to analyze and visualize. This paper illuminates the process and solution selected by ODOT, and will hopefully lead to further discussion and discovery.

BACKGROUND: EVOLUTION OF VISUALIZATION

Visualization for SWIM has evolved over the past ten years as the model has gone through updates and enhancements. An initial tool to display the results of model calibration, and then application results, was developed using SAS statistical analysis software. This tool consisted of a set of SAS scripts that created tabular and map data. The SAS scripts then created HTML files and the data was displayed on an internal website. SAS was selected because it can process very large data sets, however, only the consultant team had the knowledge to use the SAS scripts and the licenses for the software. The consulting team therefore retained all responsibility over developing and executing the scripts, while all members of the team could see the final results on the website.

For calibration of the second generation model, the client preferred using open source software and wanted to take advantage of the visualization potential of R. ODOT and PB staff created a list of calibration metrics for each model module. These metrics were key performance indicators for the model developers. The team held periodic review meetings during the development of the process to ensure that the scripts were acceptable for all members' purposes. The process was written to be flexible so that additional metrics could be added if desired. R creates high quality plots and graphs, and has sophisticated mapping capabilities. In terms of the performance and the results, R was a very good tool to use for calibration metrics, but it has some data size limitations. To

compensate, sqlite3 was utilized to do the data processing ‘heavy lifting.’ Sqlite3 works very well in coordination with R, and runs quickly compared to other options. It is also open source.

The team decided on a structure for the visualization scripts whereby each module’s scripts could be run separately or run all together. Each module had a script for running the single year diagnostics, and a script for running the diagnostics over a period of time. The sqlite portion of the process read the model output files into a database file and output .csv format files that were summaries of the large data sets. These outputs went into the same folder as the visuals. The visuals for each module were created as .pdf files. This format was chosen for its size and ease of use for multiple users. The end user could see the .pdf and .csv outputs, and if desired they could access the database file for further data analysis.

As a part of the model development, a Graphical User Interface (GUI) was developed for the team members to create and run model scenarios. The GUI was written using custom software by the PB team. The GUI contained a drop down menu for running the metrics as well, making it convenient for any member of the team to create them. The GUI also allowed end users at different locations to download the model output and any analysis files to their local computers. However, creating and moving multiple files around to different users’ computers has some drawbacks. The .pdf files varied dramatically in size and could not be manipulated easily by the end user.

When the second generation model moved into the application phase, the team decided to start at the beginning and create a visualization plan for the future of the project. Both previous approaches had run into growing cumbersomeness as the team asked for more, or for different, visuals. Additionally, since the tools were developed for calibration purposes, they were not always ideal for showing the results of model application. The results of model application are shared with a much broader audience than calibration results, including policy makers who need to see high quality visuals that communicate a great deal of information.

DEVELOPMENT STRATEGY

Partially based on the calibration visualization experiences, the modeling team realized the need for a more deliberate approach for the model application results. The team evaluated the pros and cons of several visualization approaches and attempted to answer the following questions:

- What do we need to visualize?
- Who are the end users of the tool?
- Who is the audience for the visualizations?
- What is the time frame/budget for the development of tools?
- Which tools provide the best visualizations for our purposes?
- Is licensing an issue for the preferred software?
- Should the tool be available on-line?

- What training is available?

Answering these questions helps to identify priorities and then select which software packages will best meet goals and purposes.

Primarily, visualizations help to understand model results. The large amounts of data generated in every model run are difficult to comprehend without producing graphs and maps. Visualizations help to understand model output and communicate results to others. They may provide evidence that the model is working properly. For example, presenting a graph showing that population is changing over time as one would expect helps to convince skeptical users that the model output is plausible.

It was determined that prospective users of visualization will be modeling staff at ODOT and other end-users of the SWIM framework. Though these users are very experienced with computer applications, it was desirable to create a solution that is intuitive to use and does not require extensive software knowledge. The audience for the generated visualizations is broad, ranging from planning staff and legislators to fellow modelers or technical non-modeling agency staff. Ideally, the level of detail or information density of the output should be flexible enough to address all these audiences.

The proposed visualization approach shall cover model results in economics, land use, transportation assignment and the environment. The tool needed to be flexible enough to visualize non-spatial data, zonal data, raster cell data, link attributes and directional flows, both across time and across scenarios.

- Non-spatial data are model outputs that are aggregated for the entire study area. Common examples are Gross Domestic Product (GDP), population and employment growth or Greenhouse Gas (GHG) emissions. Commonly, non-spatial data are visualized in line graphs, box plots, or scatter plots.
- Zonal data are spatially explicit by showing data for every zone in a map.
- Similarly, raster presentations show spatial data in equally sized raster cells.
- Link attributes are visualized to show results of the transportation model. Common aspects are link volumes and travel times, location of bottlenecks or deficiencies, or the volume-to-capacity (V/C) ratio.
- To analyze trip distributions, the tool further should be able to show directional link flows. This more schematic view of flows is desirable for a statewide view, over a more literal view of link flows, to display data such as origin-destination commodity flows.

The visualizations also must show data at one point in time (such as a base year or the year 2030) as well as results over time. It is also necessary to compare two or more scenarios, such as a base scenario versus a scenario with very high fuel costs.

Finally, in addition to model analysis by the model developers, the end product had to be visualizations of high enough quality to be presented to policy makers and the general public. In order to communicate a large amount of information to people who are not familiar with the technical aspects of the project, the visuals must be clean, efficient, and

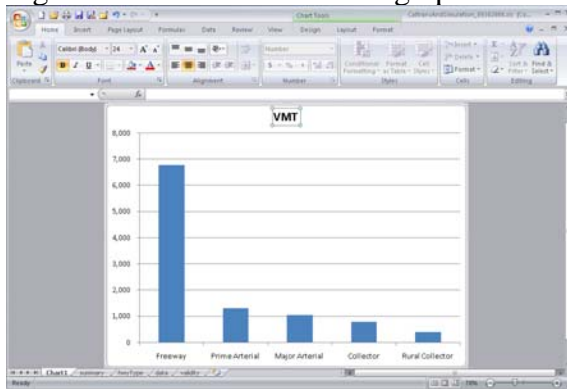
easy to interpret. The demand for visualization tools that could serve the purposes of both the analysts and the public lead to the necessity of having a master plan in place for the best way to develop these tools and utilize them.

OVERVIEW OF SOFTWARE

Solutions for data processing and visualization are almost unlimited and approaches vary as widely as simulation model designs. This section is a discussion of various software options evaluated by the team.

Microsoft Excel and *Access* have a great advantage because they are comparatively easy to use for a large variety of visualizations. Once a plot has been made many aspects of a graph may be changed with a simple mouse-click. Another advantage is its wide availability on almost every PC, reducing costs for additional software purchases. *Access* furthermore allows working with a database structure. However, *Excel* and *Access* are limited to the use of graphic types that are provided, such as box plots, line diagrams, or scatter plots. Scatter plots are limited to 32,000 observations. No graphics outside these templates can be generated. If preprocessing calculations are required, calculations may be cumbersome in *Excel*. *Access* is limited to work with databases of a maximum of 2 gigabytes. Updating graphics with new model results is fairly cumbersome unless macros are written.

Figure 1: Microsoft Excel bar graph



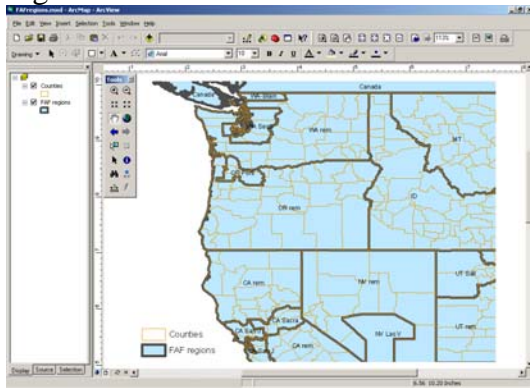
SAS is ideal for quickly processing large software sets. It is a script-based language which means that it is extremely versatile. It can easily do high level calculations and produce nicely laid out tables, plots, graphs, and maps. There is a steep learning curve for this software and it is proprietary so it is necessary to purchase a license.

Sqlite3 is an open source version of *MySQL*. The 'lite' in the name refers to reduced functionality of this software, which consists of a small (approximately 73 kb) executable file. It is an open source software. For most data processing, the reduced functionality is not noticeable. There are some types of joins and some mathematical operations that are not possible. However, this is a program that can very quickly summarize huge data files that most software cannot deal with. This software does not have any visualization

component on its own, but can work in tandem with another software program (i.e. Excel or R) that can do the visualization but not handle large data sets.

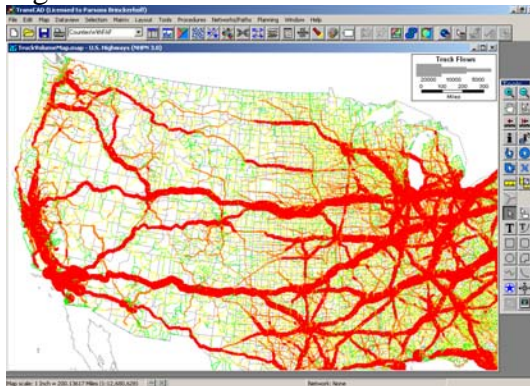
Geographic Information System (GIS) software can be used to create very rich maps. ESRI ArcGIS is one of the leading GIS packages, and is already used by ODOT. Professional appearance of both two- and three-dimensional maps can be achieved even for a new user. Depending on features included the software is currently priced between \$1,500 and \$2,500. Maptitude, by the Caliper Corporation, is another GIS package with comparable capabilities. The open source Quantum package is another possibility, although its features are far less developed than the commercial products. This software is limited for visualizations of data that are not in map format. Another drawback is the restricted capability to draw maps of traffic flows.

Figure 2: ESRI ArcView screenshot



Commercial Transportation Software (i.e. CUBE, EMME3, TransCAD, VISUM) are very powerful tools to visualize transportation flows. However zonal mapping and plots other than maps are fairly limited. For both GIS and commercial transportation software it tends to be cumbersome to update a plot with updated model results. Either a specific script needs to be written or updated data has to be linked to the plot manually.

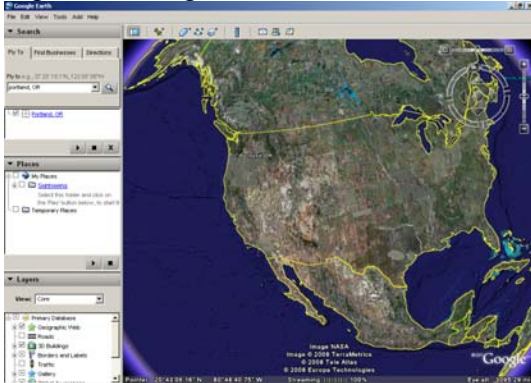
Figure 3: VISUM screenshot



Google Maps and *Google Earth* offer road maps and satellite images for most parts of the world. Google Earth does require a software license for professional users. Google Earth has a rich API that allows many traditional GIS functions to be applied from within the

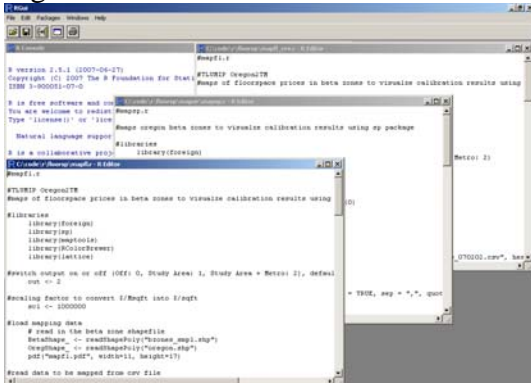
program, as well as to use GIS data. It also shows satellite images as a background, which may be desirable for some maps. The map generation is very slow when many shapes are visualized therefore for most modeling purposes it is not adequate.

Figure 4: Google Earth screenshot



R scripts offer a highly flexible way to produce quality maps, plots, and graphs. R is a script language created for statistical analysis. Several Graphical User Interface (GUI) packages allow the development of a user-friendly tool for executing R scripts. Though R is open source, this functionality comes with the cost of a steep learning curve. R has some memory limitations therefore is not ideal for large data sets. R is not multi-threaded, so cannot take advantage of the multiple computer system that runs the model.

Figure 5: R screenshot



Programming Language Software written in a Higher Programming Language allows unlimited visualization options, only constrained by the knowledge of the programmer. Common examples include Java, C++, Python or FORTRAN. A GUI is available for most programming languages. Some languages require the purchase of a compiler, while free compilers are available for other languages. Training needs for developing and modifying code are at least as high as they are for R scripts.

Some models, such as PECAS or Urbansim, are moving to database storage of inputs and outputs attempting to facilitate data flows and visualizations. However, databases themselves do not provide visualization, and thus must be combined with graphical software packages listed here. For example, in the PECAS model for the state of

California, SQL-server databases are combined with ArcGIS and Excel/Access to visualize results.

CURRENT APPROACH

Currently, the statistical language R is used by ODOT and PB to develop visualizations. Challenges have arisen, and the idealized process laid out in the development strategy has not been fully adhered to. During this construction phase of the tool, graphics and data storage has been kept fairly loose to allow for multiple software programs to quickly access the data. This is so that team members can test different approaches, identify and solve problems, and analyze results.

In the near future, the first GUI will be utilized. Shortly after that, additional visualization tools will be built in and plot and data output options will be added. Examples of the visualizations currently produced are shown in Figures 6-9. As development continues, the tool is anticipated to continue to approach and conform to the structure/vision outlined in the following sections.

Figure 6

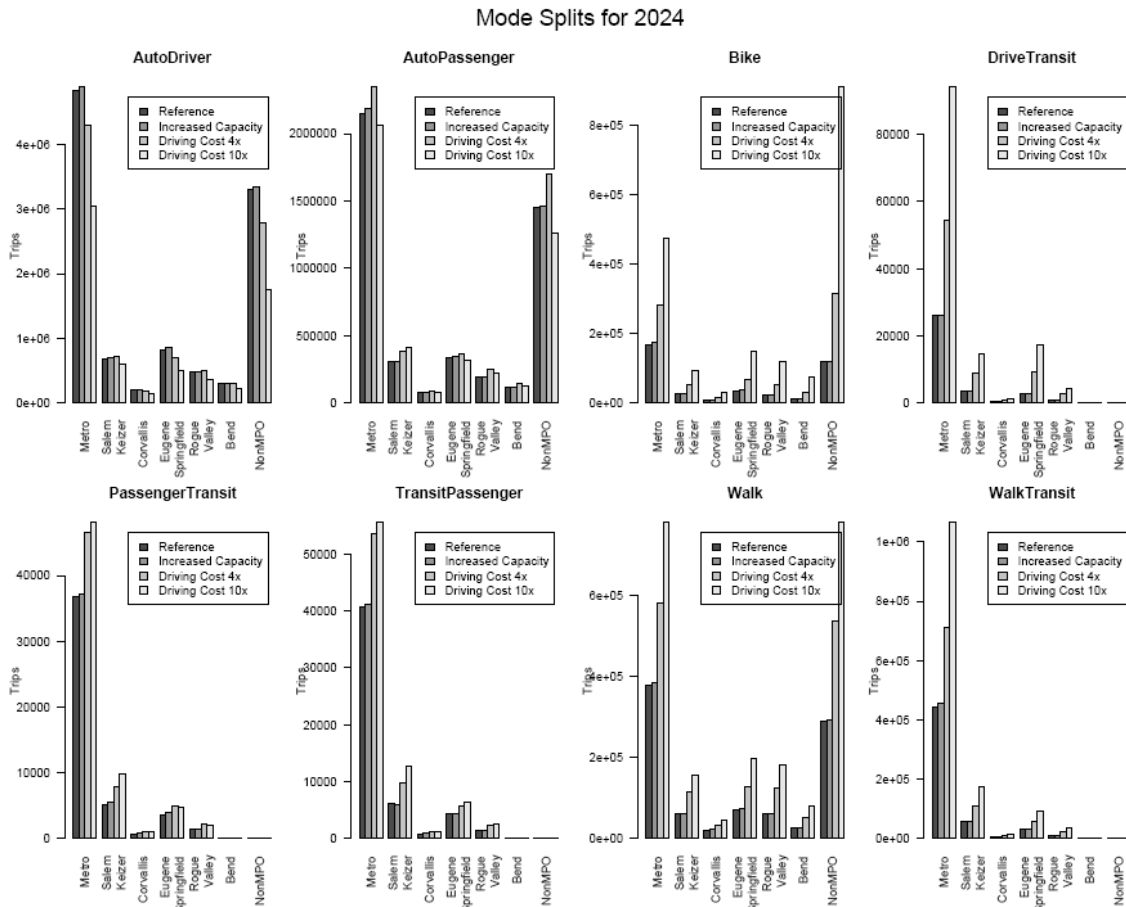


Figure 7

Difference Between High Cost (4x) Scenario and Reference:
Employment - Total 2024 Comparison

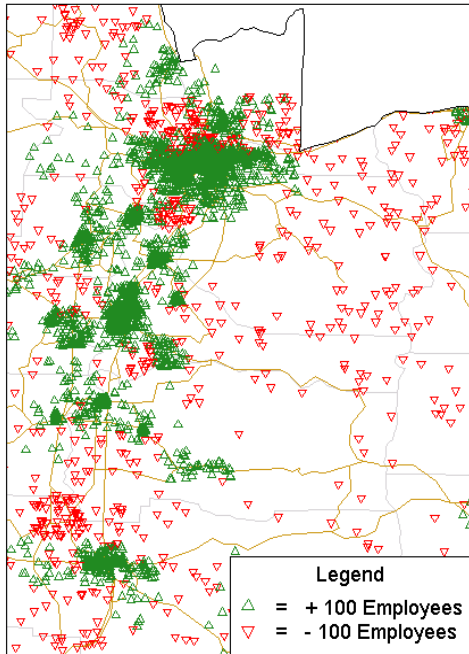


Figure 8

Difference Between High Cost (10x) Scenario and Reference:
Synthetic Population - TotalHHs Summary

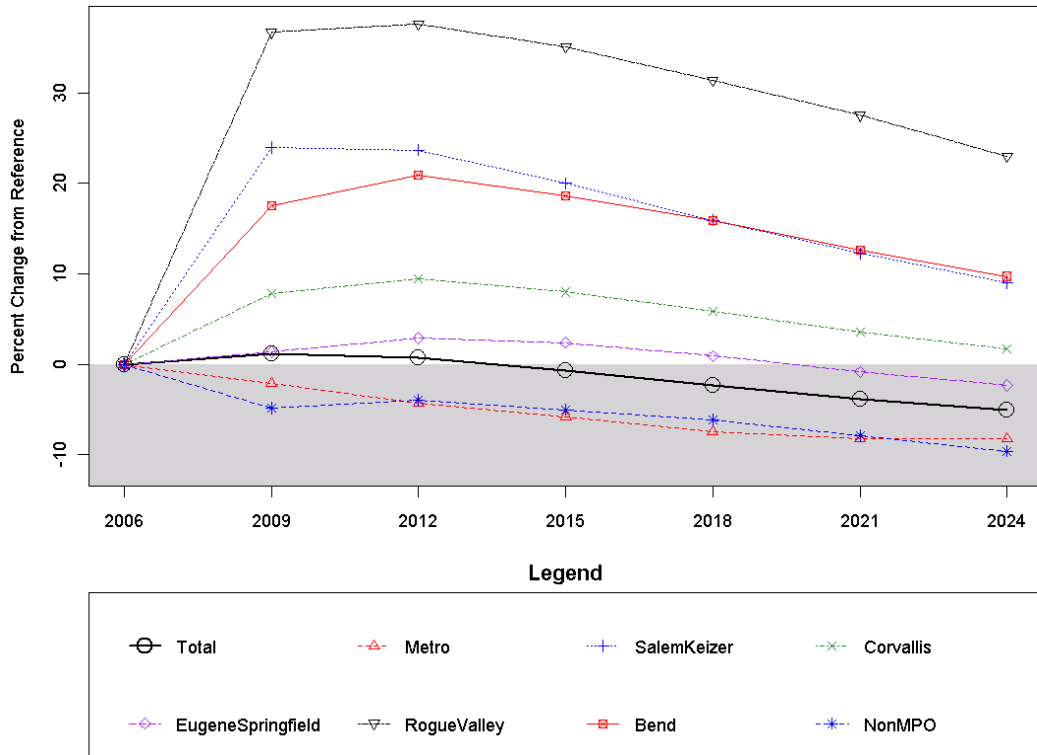
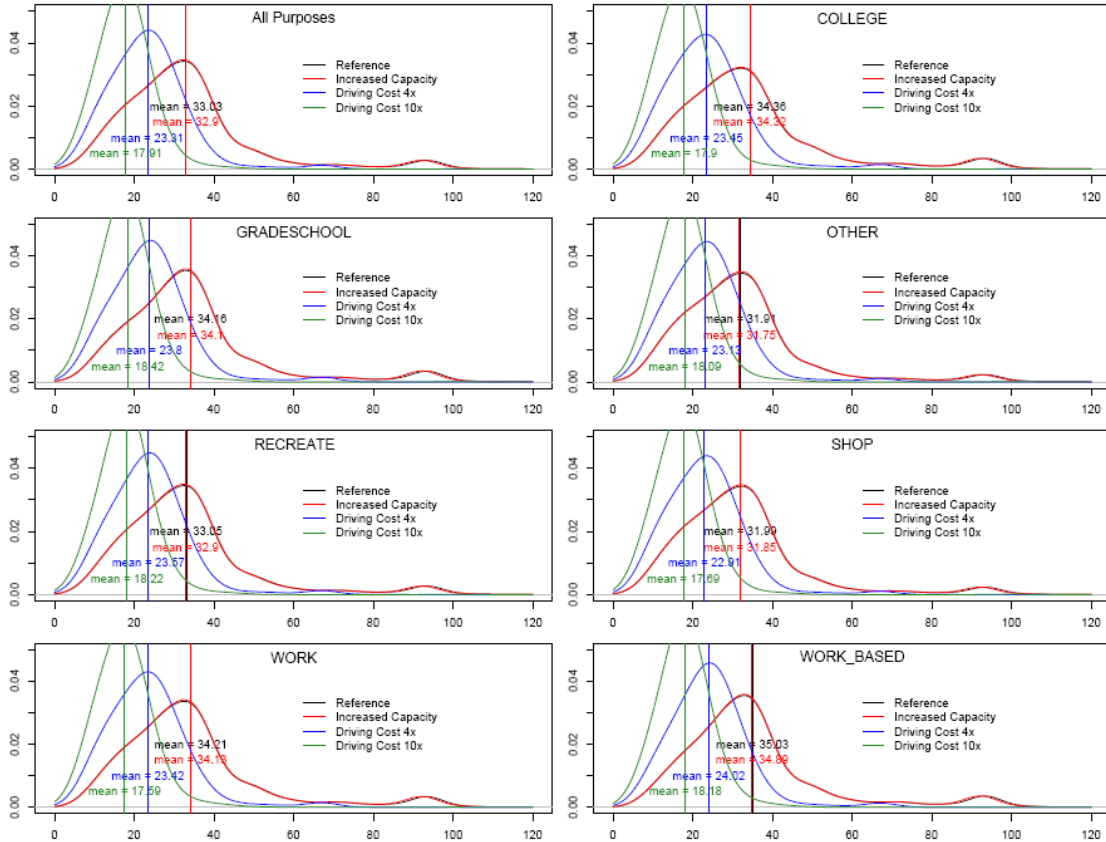


Figure 9

2024 – Tour Travel Time Distribution – For the State of Oregon



SWIM RECOMMENDATION FOR FUTURE

The examples given in section 0 only give a glimpse of possible tools to visualize model output. Given the advanced status of SWIM, which shall serve statewide and regional planning in the long run, it was recommended that either R or a higher programming language are the best options for SWIM visualizations. This allows ODOT to visualize an enormous variety of model output that is requested today or may be demanded in the future. Policy makers are likely to ask complex questions, and the model is designed to deal with these questions. Consequently, the visualization tool should be able to explore the model output to the largest and most flexible extent possible.

The programming language used for generating such a tool is a minor issue, as long as the language is capable of generating meaningful, easily comprehensible and professionally looking graphics. One major concern in choosing a language should be how widely distributed it is. If there is only one staff member or consultant who is trained in a certain language, additional scrutiny should be given to weigh how dependent ODOT becomes on this one person.

Some staff members at ODOT are proficient in using R, which might be an argument to build such a tool in this language. R has the additional advantage that it is open source

and has contributed tools. Since R was used for the previous visualization, some scripts could be reused or modified. Potentially higher run times of R scripts could be reduced by pre-processing output data as far as possible. Sqlite was used for pre-processing in the earlier phase, but other software including Python or Awk could be used for that purpose as well. Preprocessing would be required only one time after every model run. This would keep the visualization tool efficient and the user could produce a larger number of plots in a relatively short time.

It is also recommended that a GUI is the ideal way to execute the visualizations; particularly since the team members will not all be at the same competence level for any given software. A GUI can be shared and used across staff with different levels of involvement or software knowledge. Since R was developed for statistical analyses, its GUI capabilities might be too limited for this purpose. As in the earlier phase, the R scripts could be linked to a GUI written in a higher programming language, such as Java, FORTRAN or Python. These three higher programming languages would also be capable of supporting the entire visualization tool.

Regardless of which language is chosen, such a tool does not preclude results from being exported from the visualization tool to other graphic packages to produce more sophisticated or more polished graphics for specific audiences.

DESIGN PROPOSAL

This section develops a proposal of the design for the visualization tool. It is assumed that either R scripts or a higher programming language are chosen to build this tool. Such a tool offering complex visualizations has to grow step by step as it is developed. This design proposal is meant to be an initial approach which shall be refined as the tool develops. By building up the capabilities of this tool over time in accordance with visualization needs, ODOT gets some useable functionality quickly, remains flexible when conditions change and avoids a big outlay of time/cost up-front.

Summary File

SWIM produces a very large amount of output data, totaling 65 gigabytes of files for a 20 year model run. Reading these datasets every time the visualization tool is started would be very cumbersome and time consuming. Instead, a pre-processor should read all data and produce a summary file that contains the relevant data. This summary file could be copied from the computer cluster where the model runs to a local desktop machine, which would also eliminate user frustration due to possibly unstable network connections. If desired, permission to copy data could be limited to a selected set of authorized users.

This summary file should be kept as sparse as possible, including only data the visualization tool is prepared to visualize. At the beginning, this file might be very small and grow as additional features are added to the visualization tool. At some point, it might be advantageous to automate the preprocessing step to run at the completion of a SWIM model run. This would run the visualization tool immediately after a model run. At this point, however, it is recommended to keep the preprocessing tool separate to be

able to reproduce the summary file every time the visualization tool is enhanced and requires more information in the summary file.

Visualization output

There are many different ways to visualize data. The following are the most likely visualizations that a model user would want to create. All of these examples can be created by R scripts or by a higher level programming language.

Box plot. A common way to visualize non-spatial data is box plots, using either vertical or horizontal boxes.

Line diagram. Line diagrams are used to visualize development over time, where the x-axis represents time.

Range plot. These plots are used to analyze the range of values, showing for instance the range of simulated land prices. It shall visualize the minimum and the maximum (with whiskers), the mean and the median (with bold lines), and the 25 and 75 percent quartiles (with box plots).

2D Choropleth Maps, 2D Isoline Maps and 3D Choropleth Maps. These maps show zonal data in a 2D or 3D map. Choropleth maps show densities, as absolute values cannot be analyzed reasonable due uneven zone sizes. Alternatively, density dot plots could be used. Density dot plots, however, hide information for small zones, as one dot in a small zone can stand for high and low density. Density dot points may be helpful if values do not change abruptly from one neighboring zone to another, or if larger zones (such as counties) are mapped.

Network maps serve to visualize, for instance, simulated volumes, volume-per-capacity (V/C) or congestion on the network.

Arrow Map. Arrow or spider maps are special map types used to visualize origin-destination flows. In SWIM these flows include person trips, truck trips and commodity flows.

Scatter plots. Scatter plots compare two variables. One of them could be model result and the other one could be target data to validate model output. A third variable could be visualized by color.

File output

Using a GUI gives the option to display Visualizations on the screen, allowing the user to analyze model output immediately without opening another application to view the generated visualization. However, there are up to three additional different files that are desirable as output. First, the tool should provide the option to write a simple ASCII file of the values visualized on the screen. While the screen may show a nice box plot diagram, it would be helpful in some cases to know the exact value of every box. Such values could be found in this ASCII output file. The ASCII file could be a comma-

delineated file (*.csv), which can be opened in MS Excel and other software. The R format (*.RData) could be helpful for further adjustment of plots using R. This output is also helpful for producing specialized graphics that are not included in the standard output of the visualization tool.

Second, the tool should offer to plot the graph shown on the screen into a graphical output file (such as *.wpg or *.png). Possibly, an additional pdf file of the graphic shown on the screen could be helpful as well, though for including plots into a word document, true graphic files are preferred. The GUI should offer to switch off the writing of these files, as some file types may slow down the tool.

Future extensions

Most likely, there is a standard series of plots that ODOT wants to produce after every model run. This could include a map with change of population, a range plot of floorspace prices and a map with traffic volumes, and mainly serves to check that the model finished successfully and produced reasonable results. Once this tool is developed it would be comparatively easy to define such a series of around 15 visualizations, and create an ASCII file where these 15 plots are defined. The GUI could have an option for the user to create the standard set of maps with one simple selection. Automatically, a pdf file with these 15 standard plots would be created. This could provide a quick overview that all simulation modules have worked as expected, and would also allow a quick comparison of several scenarios.

In the long run, it may be desirable to extend this tool with online capabilities, allowing users at different locations to produce plots of model results. However, if the tool is built using an output summary file (as proposed at the beginning of this section), this file could be easily shared among users and an online capability appears to be of minor significance. It is less complex and error-prone to simply send the output summary file to all users that may want to create visualizations with this tool.

There are uncountable further software tools that provide unique pre-programmed visualizations that are relatively easily adapted for this tool. This might include the following formats:

- P-plots, which could plot place attributes or flows if one end was fixed, over time
- Lattice graphics of small multiple plots/maps across potentially 2 dimensions (left-to-right or top-to-bottom).
- Iso- or R-Creeping of small multiple maps
- Circle plots on maps, rather than same size dots
- Animation to spin a 3D scatter plot.
- Map with blow-ups of MPO areas
- Visualizations written in other languages for other models (e.g. Fortran tools)

As the need for such additional plot options arises, the GUI could be extending accordingly and the tool would call such procedures with specialized visualization.

CONCLUSIONS

This paper is meant to be a tool for introspection for the modeling team working on the SWIM model as well as for other teams dealing with data intensive projects. It is recommended that agile software development concepts be followed in the development of a visualization strategy. This concept is borrowed from computer science. It starts with the simplest implementation possible and extends the visualization tool step by step. At any given point of time, the visualization tool is fully operational. It is recommended to start with the simplest visualization first, such as box plots. Step by step, the tool can be extended, guided by actual visualization needs.

The visualization program developed is not an end state, but a process with a growing number of output results, conforming to a small set of modifiable templates for visualizing key types of data. With each new model application, there will be an interest in new visualizations.

No one solution fits all problems. With limited resources to develop the visualization tool, only a finite number of options to visualize results will be available. There always will be special cases where a very particular visual is needed. For example, a certain set of maps may be necessary for a conference session. Visuals like those do not need to be a part of the visualization strategy, but can be generated manually for the purpose at hand. The purpose of the strategy and tool laid out in this paper shall serve to quickly analyze the most important model output data with a substantial number of present visualizations.