VisWaterNet: Visualization of Water Distribution Networks

Meghna Thomas, S.M.ASCE[1]; Tyler Trimble[2]; and Lina Sela, Ph.D., A.M.ASCE[3]

[1]Ph.D. Student, Dept. of Civil, Architectural, and Environmental Engineering, Univ. of Texas at Austin, Austin, TX. Email: meghnathomas@utexas.edu
[2]Undergraduate Researcher, Dept. of Civil, Architectural, and Environmental Engineering, Univ. of Texas at Austin, Austin, TX. Email: tylerl.trimble@utexas.edu
[3]Associate Professor, Dept. of Civil, Architectural, and Environmental Engineering, Univ. of Texas at Austin, Austin, TX. Email: linasela@utexas.edu

ABSTRACT

Many researchers and engineers working with water distribution systems have the need to model, simulate, analyze, and visualize these systems. Several open-source tools, such as EPANET and the Python package WNTR, allow users to perform extended period simulations and analysis of water distribution system models under a range of different conditions. However, while these tools provide basic visualization functionality, they are limited in their flexibility and capabilities. Here, we present VisWaterNet, a Python package that enables clear and customizable visualization options specifically catered toward water distribution network models. VisWaterNet provides users with straightforward functions to plot simulation results and intrinsic characteristics of system nodes (e.g., pressure and demand) and links (e.g., flow rate and pipe diameter) by customizing the size and color of network elements. The package allows for the visualization of single time step simulation data as well as data summarized over extended periods. Users may specify if data should be represented in continuous fashion with a color bar or as discrete values with an accompanying legend, and can also plot categorical data that may be generated or imported independently. Furthermore, this package includes a function to produce animations that display changes of network properties, such as nodal demand or link head loss, over time. VisWaterNet includes scripts that highlight its functionalities, as well as applications that demonstrate how functions can be applied in conjunction to create illuminating plots. For example, the package can be used to display possible sensor placement locations, highlight the concentration of a chemical species throughout the network, or present statistical summaries of model simulation outputs. The result of our work is a streamlined, user-friendly package that facilitates the creation of neat, comprehensible plots to aid researchers and practitioners in communicating results of applications of design and analysis.

INTRODUCTION

Water distribution systems (WDSs) can be very large and may contain thousands of elements. These elements have static characteristics (e.g., pipe diameter, nodal elevation) as well as time-varying states (e.g., pipe flow rate, pump status, nodal pressure). In current practice, software such as EPANET, WaterGEMS, and InfoWater Pro are used to run extended period hydraulic simulations to ascertain the state of a WDS (Rossman et al. 2020; Bentley Systems 2022; Innovyze 2022). Visualizing the characteristics of WDS elements and overlaying simulation results over the network topology can allow designers and practitioners to better understand and communicate their findings.

Open-source visualization tools, such as Gephi, Cytoscape, Graphviz, and Pajek, can generate sophisticated visual representations of networks from large datasets (Bastian et al. 2009; Shannon et al. 2003; Ellson et al. 2004; de Nooy et al. 2018). These tools allow the easy customization of network plots and automation of updates and changes; however, they typically cater to general network structures and it is difficult to apply them to the context of WDS network visualization. Some software that are primarily tailored towards hydraulic analysis can provide basic visualization functionalities. A widely-used open-source hydraulic simulation software, EPANET, can be used to build and run hydraulics models of WDS networks through a Graphical User Interface (GUI). EPANET users can visualize networks to highlight system parameters and simulation results through the Map Browser functionality, and are able to query attributes and generate animations to observe the change of system states over time. However, the visualization options and the range of attributes that can be visualized on EPANET is limited, and the user must rely on the GUI to generate visualizations.

In order to increase the flexibility in modeling and simulating WDS networks, EPANET wrappers have been developed for a number of programming languages to facilitate code development and collaboration. Arandia and Eck (2018) developed an EPANET wrapper for R that enables hydraulic simulations but lacks an automated option to visualize network properties and simulation results in conjunction with the network topology. The EPANET wrapper developed for MATLAB by Eliades et al. (2016) offers some elementary network visualization functions, but their scope is mainly limited to drawing the WDS layout. WNTR, a free open-source Python package for WDS modeling, offers users the options to run hydraulic simulations either with an EPANET engine or its own hydraulic solver, as well as conduct resiliency analyses (Klise et al. 2017). WNTR contains a host of visualization options, such as the ability to generate basic network plots, interactive plots, map overlay plots, and network animations. However, there are limitations to using even WNTR's basic plotting functions; WNTR relies on the Python packages NetworkX and Matplotlib to construct its static plots, but the full range of plotting functionality offered by these packages is not available to the user (Hagberg et al. 2008; Hunter 2007). One wishing to generate a network visualization through WNTR must rely on its rigid functionality and will find it challenging to create plots of categorical or grouped data or to customize element parameters (such as node size, edge widths, or assigning labels).

EPANET and its programming language wrappers provide a good starting point for WDS network visualization but are limited in their capabilities. How might one generate a plot to display categorical data like nodal demand patterns? Emphasize the diameter of each pipe in a network with both color and link width? Represent the average and variation in nodal pressures over the duration of a simulation? These tasks will be taxing with existing visualization and simulation software. In this paper, we introduce VisWaterNet: a Python package that enables the visualization of WDS networks. VisWaterNet builds upon the Python packages WNTR, NetworkX, Matplotlib, Numpy, and Imageio to generate customizable and adaptable plots (Harris et al. 2020; Klein et al. 2022). Functions offered by VisWaterNet can replicate current plotting standards and have additional features that include plotting categorical and discretized data, displaying multiple dimensions of data in a single plot, and visualizing custom user-imported data (not extracted from simulation results). This paper provides a brief overview of the functionality offered by VisWaterNet, followed by a series of example applications. Additional examples are available in the GitHub repository: https://github.com/tylertrimble/viswaternet.

DESIGN

Package Framework

VisWaterNet is a collection of Python scripts that enables to easily generate and customize water network plots. Figure 1 describes a high-level overview of the process of generating a plot using VisWaterNet.

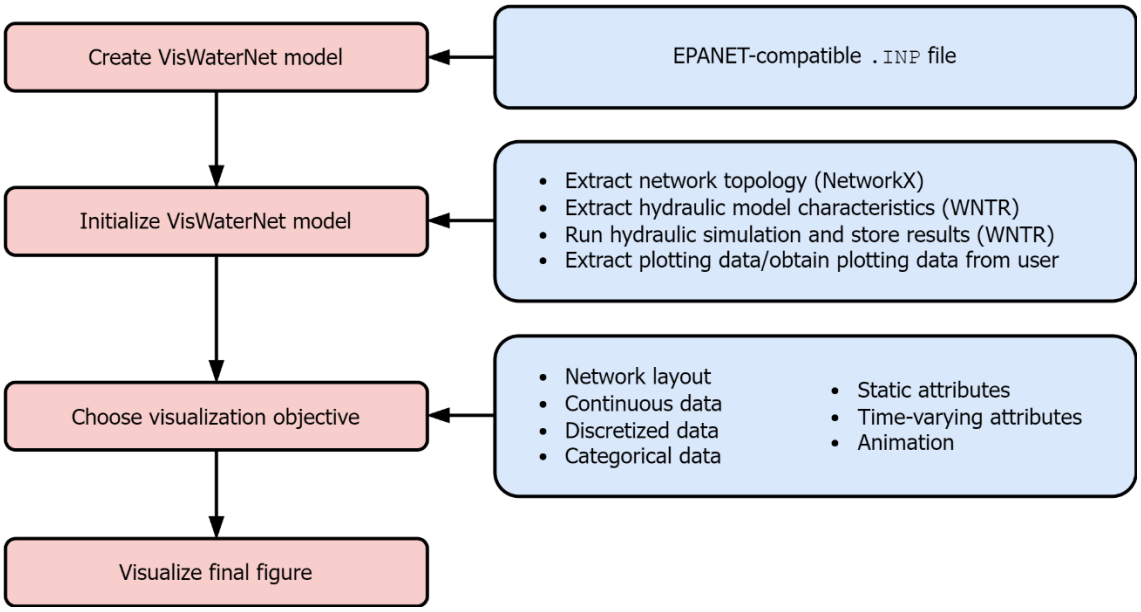

Figure 1: VisWaterNet framework.

First, the package constructs and initializes a VisWaterNet model from a .INP file provided by the user. This VisWaterNet model utilizes WNTR to extract information about the inherent network characteristics and hydraulic simulation results, and uses NetworkX to extract information about the network topology. Next, the user may specify the type of plot to generate. Most time series data or network characteristic data are represented on a plot by color-coding each data point using a corresponding color. VisWaterNet provides the option to visualize data in either a continuous manner, which means the range of colors representing the data points are selected along a color bar (a gradient color scale); or in a discrete manner, which means the data has been grouped into intervals and each interval is represented by a single color. Users may also choose to visualize data by varying node sizes and link widths. Some network characteristics are "categorical"; i.e., a specify property of each node or link will belong to one of a fixed set of categories (e.g., nodal demand pattern). VisWaterNet allows for the plotting of continuous, discrete, and categorical data, as well as custom data provided by the user (either through a list imported into the script or an Excel file). Table 1 contains a summary of the different visualization functions offered by VisWaterNet.

Each plotting function has a range of optional inputs that can allow the user to further customize the plots. For example, the user may provide input parameters to (1) change the units of the data from the WNTR default SI units to other units; (2) choose from Matplotlib options to customize plot color scheme, node markers and sizes, and link line styles and widths; (3) specify

the number of significant digits in legends and color bars; (4) customize the number of intervals and width of each interval for discrete data plots; (5) hide intervals with no data points in discrete data plots; (6) customize the minimum and maximum extent of color bars in continuous data plots; and (7) load and plot node/link data from Python lists or Excel files.

Table 1: Description of VisWaterNet functions.

Category	Function	Description
Initializing the VisWaterNet model	VisWNModel	Creates VisWaterNet model instance from .INP file and obtains information about the WDS such as network topology, inherent characteristics of network elements, and hydraulic simulation results. Hydraulic simulation of the network is performed using EPANET 2.2.0 solver.
Comprehensive plotting functions	plot_basic_elements	Plots the basic layout of the network.
	plot_continuous_nodes	Plot continuous node data, e.g., mean nodal pressure, with a color bar.
	plot_continuous_links	Plots continuous link data, e.g., flow rate at a particular time step, with a color bar.
	plot_discrete_nodes	Plots discretized node data, e.g. maximum water age, with a legend.
	plot_discrete_links	Plots discretized link data, e.g., maximum pipe velocity, with a legend.
	plot_unique_data	Plots categorical or user-generated data, e.g., pipe diameters, node demand patterns, data imported from a list or an Excel file, as either a continuous or discrete plot.
	animate_plot	Generates GIF of time-dependent parameters across the entire simulation duration or for a select range of time steps.
Additional plotting functions	draw_labels	Adds labels to specific elements to improve plot readability.
	draw_nodes	Plots specific nodes on the plot.
	draw_links	Plots specific links on the plot.
Data processing functions	get_parameter	Extracts static network characteristics or time-varying simulation results.
	bin_parameter	Fits continuous data into user-defined intervals.
	normalize_parameter	Normalizes data.

EXAMPLE APPLICATIONS

This section contains a selection of example applications to demonstrate the capabilities of VisWaterNet. A .INP file of the benchmark WDS network CTown (Ostfeld et al. 2012) was provided as the input network for each example. Figure 2 shows a code snippet that is included at the beginning of each example script that: (1) imports the VisWaterNet and Matplotlib libraries into the script; (2) receives the name of the .INP network file; (3) initializes the VisWaterNet model; and (4) creates an empty Matplotlib figure to populate with subsequent VisWaterNet functions.

```
#Import packages
import viswaternet as vis
import matplotlib.pyplot as plt
#Initialize model
model = vis.VisWNModel('CTown.inp')
#Create matplotlib figure and axis
fig, ax = plt.subplots(figsize=(12,12))
#Disables figure frame
ax.set_frame_on(False)
```

Figure 2. Setup script for example applications.

Example 1: Visualizing Pipe Diameters

Plotting the pipe diameters present in a network can deliver a clear visual representation of the different sizes of pipes in the WDS, as well as highlight major transmission lines. Since pipe diameters are typically chosen from a fixed set of options rather than a continuous range of values, VisWaterNet treats diameters as a categorical attribute. Figure 3 shows the code snippet used to generate the plot. Line 3 specifies the parameter to plot (i.e., diameter) and its units (inch); lines 4-5 control the width of each pipe according to its diameter and the colormap; lines 6-7 specify the location of the legend and title; and lines 9-10 define the appearance of the pumps. Figure 4 depicts the resulting visualization.

```
1  #Plot diameter of each pipe
2  model.plot_unique_data(
3        ax, parameter="diameter", unit="in",
4        interval_link_width_list=np.linspace(1,7,10), #Controls link widths
5        cmap="Blues",
6        legend_loc_2="lower left", #Location of legend with bins
7        legend_title="Pipe Diameter (in)",
8        legend_sig_figs=0, #Whole numbers only
9        pump_color="mediumseagreen",
10       pump_width=5) #Specify link width in pixels
```

Figure 3. Example script for generating a plot of the pipe diameters of a network.

Example 2: Visualizing Nodal Demand Patterns

Some WDSs have different demand patterns assigned to different junctions in the system. Being able to view the demand pattern of each junction on the network layout may lend insights into the location, boundaries, and characteristics of distinct zones containing junctions with the

same demand pattern. In this example, we demonstrate VisWaterNet ability to easily assign colors to junctions on the basis of their demand pattern, which is a categorical attribute. Figure 5 shows a code snippet used to generate the plot. Line 3 specifies the parameter to plot (i.e. nodal demand pattern); line 4-6 determines the node size and legend label for each demand pattern; line 7 assigns the plot color scheme; line 8 controls the appearance of the legend; and line 9 hides pumps and valves on the plot. The resulting plot is shown in Figure 6.

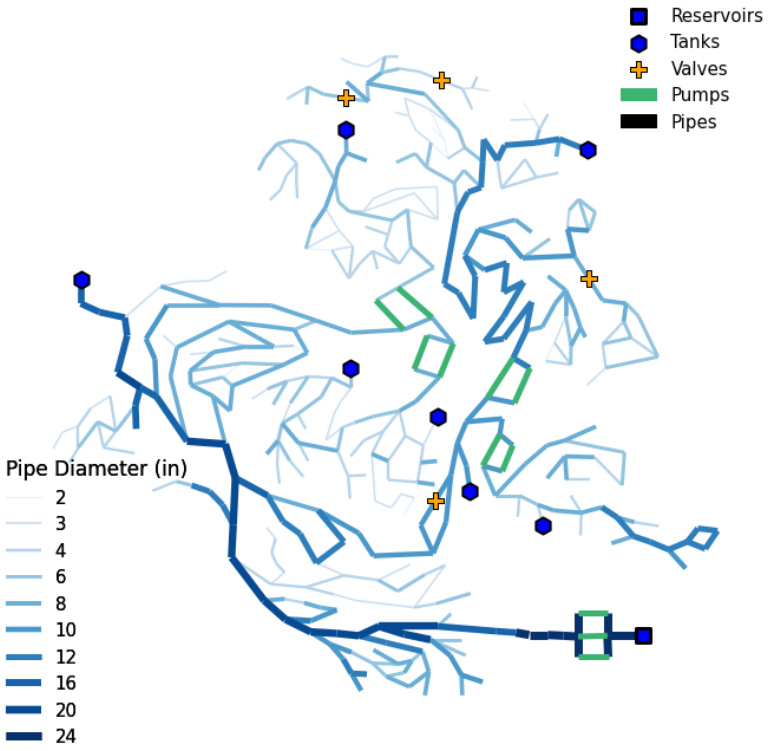

Figure 4. Categorical plot of pipe diameters.

```
1 #Plot demand pattern for each junction
2 model.plot_unique_data(
3       ax, parameter="demand_patterns",
4       interval_node_size_list=[300,300,300,300,300,300],
5       interval_label_list=['Pattern 1','Pattern 2','Pattern 3','Pattern 4',
6       'Pattern 5', 'No Pattern'],
7       cmap="Dark2", #Choose color scheme
8       legend_loc_2="lower left", legend_title="Demand Patterns",
9       pumps=False, valves=False) #Disable plotting pumps and valves
```

Figure 5: Example script for generating a plot showing junction demand patterns.

Example 3: Visualizing Nodal Water Quality

The network structure of WDSs facilitates not only the transport and supply of water across large areas, but also the risk of the spread of contaminants. There is keen interest in estimating and managing the quality of water throughout a WDS. Increased water age, which is the amount

of time since water entered the network, can lead to a higher likelihood of water quality problems, and, therefore, age is a useful measure to gauge the quality of water in a system (Kourbasis et al. 2020). In this example, we utilize VisWaterNet's ability to discretize and plot simulation data to visualize the maximum water age of each junction. Here, the parameter "quality" is set to "Water Age" in the input .INP file. Figure 7 shows the code snippet used to generate the plot, in which line 3 specifies the plotting parameter (i.e. maximum value reported for each node in the quality simulation results) and units (hour); lines 4-6 control the boundary values, node size, and node border width of each interval; and line 9 controls the number of significant digits in the legend labels. The resultant water age plot is displayed in Figure 8.

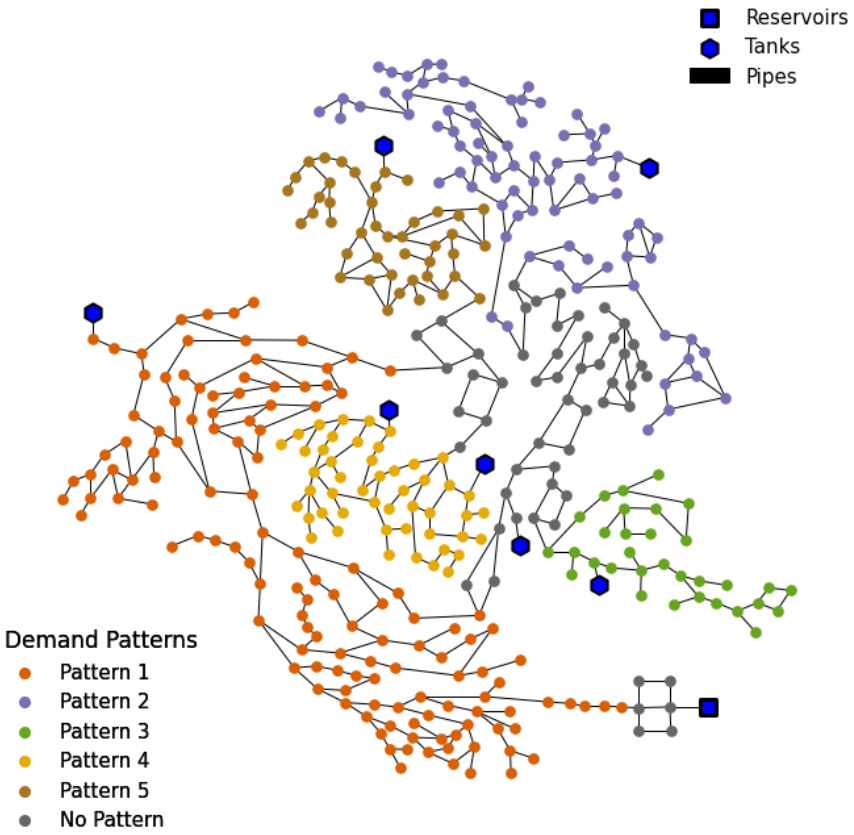

Figure 6: Categorical plot of junction demand patterns.

```
1 #Plot maximum water age in hours
2 model.plot_discrete_nodes(
3       ax, parameter="quality",value='max',unit="hr",
4       intervals=[0,6,12,24,48],
5       interval_node_size_list=[100,150,250,350,450], #Node sizes
6       interval_node_border_width_list=[1,1,1,1,1],
7       cmap='Greens',legend_loc_2="lower left",
8       legend_title="Max Water Age (hr)", legend_title_font_size="16",
9       legend_sig_figs=0, #No decimal places
10      pumps=False,valves=False)
```

Figure 7: Example script for generating maximum water age plot.

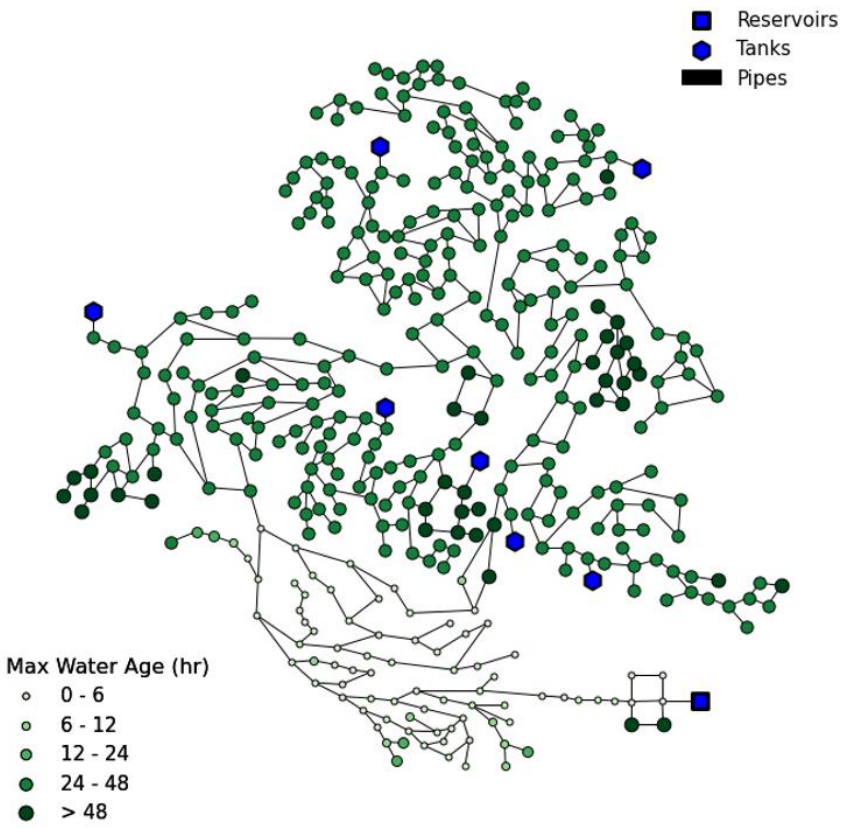

Figure 8: Discrete plot of maximum water age for each junction.

Another useful water quality parameter, trace, describes the percentage of water at a node/link that originates from a source node. Source tracing can aid in determining the fate and transport of a contamination event. Here, we utilize VisWaterNet discrete plotting functionality to visualize the maximum trace value from a specific tank at each junction. The "quality" parameter in the input .INP file is changed to the "Trace" option, and tank T1 is selected as the source node. Figure 9 shows the code snippet used to generate this plot. Here, line 3 selects the parameter (i.e. maximum value in the quality simulation results); lines 4-5 customize the legend and hide pumps and valves; and in lines 8-11, we demonstrate how to add a custom label to the plot. Figure 10 depicts the final visualization.

```
1 #Plot trace quality parameter
2 model.plot_discrete_nodes(
3       ax, parameter="quality", value="max",
4       legend_loc_2="lower left", legend_title="Max Trace from T1 (%)",
5       legend_sig_figs=0, pumps=False, valves=False)
6
7 #Label Tank T1
8 model.draw_label(
9       ax, labels=["T1"],nodes=["T1"],
10      x_coords=[0],y_coords=[150],
11      draw_arrow=False,label_font_size=9)
```

Figure 9: Example script for generating max trace plot from tank T1.

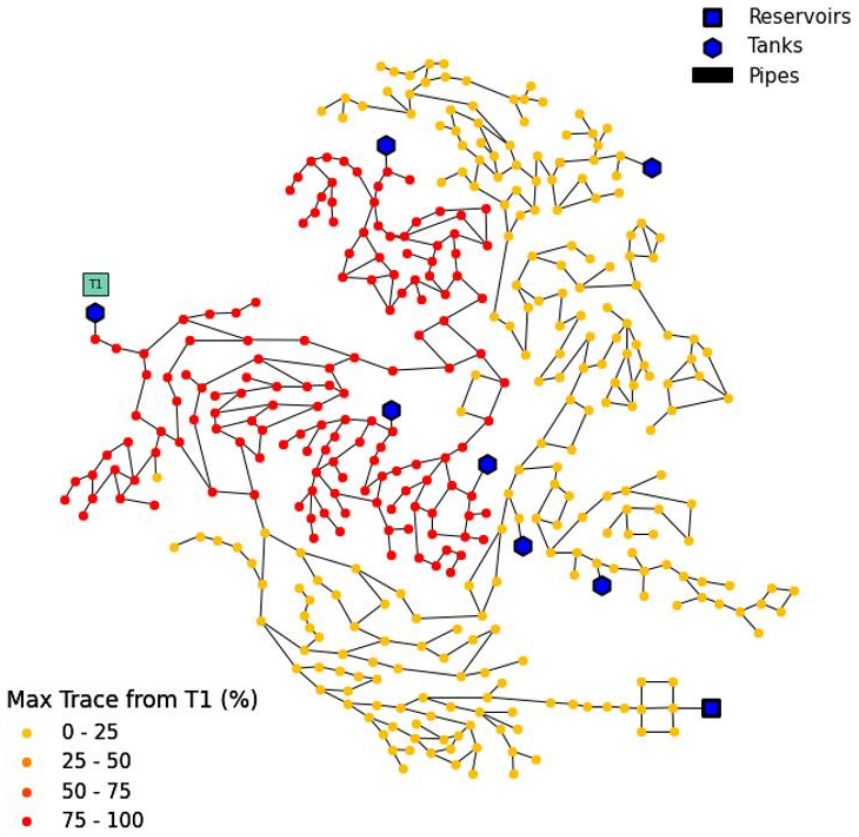

Figure 10: Discrete plot of maximum trace from tank T1.

Example 4: Visualizing User-Generated Data

This example demonstrates a method to visualize user-imported data. State estimation and optimization of network design and operation are widely explored topics in the field of WDS research. Visualizing the results of studies in these domains over network maps can prove challenging, as existing visualization tools provide little support for overlaying and customizing external data over WDS network layouts. As an example, we demonstrate a method to visualize the maximum nodal head deviation between hydraulic simulation data and estimated values from an external solver. The external data is imported in from another Python script in the form of "*element_list*", which includes a list of node names, and "*error_list*", which includes a list of error values corresponding to the nodes. The plot is generated using the code snippet shown in Figure 11. Line 5 specifies that custom data will be used; line 6 specifies that continuous node data will be plotted; and line 7 specifies the node list and values to be plotted. The resulting plot is shown in Figure 12. Note that VisWaterNet can scale the color map between values of equal magnitude when the data consists of both negative and positive values. This scaling, combined with choice of color scheme, highlights the location and magnitude where simulation results over- and underestimate the head at the nodes in the network. Similarly, external data can also be imported from other source files (e.g., CSV and Excel files) and loaded into Python. The functionality of plotting external user-imported data is useful to visualize results in different case-specific applications that do not directly rely on WNTR simulations.

```
#Import generated nodal error data from another script
from nodalerror_generation import element_list, error_list
#Plot custom data generated in nodalerror_generation.py
model.plot_unique_data(
      ax, parameter="custom_data",node_size=200,line_widths=1,
      edge_colors="k",parameter_type="node",data_type="continuous",
      custom_data_values=[element_list, error_list],
      color_bar_title="Error (%)",cmap="bwr",draw_color_bar=True)
```

Figure 11: Example script for generating plot showing junction head deviation.

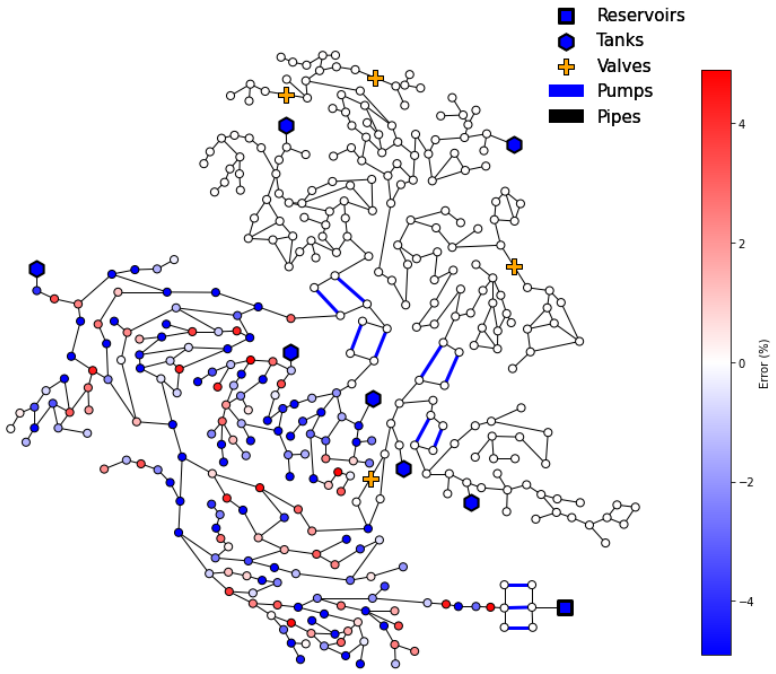

Figure 12: Continuous plot showing deviation in junction heads between simulation and estimation results.

Example 5: Visualizing Summary Measures

Visualizing the summary measures of WDS simulation results is useful in characterizing overall trends of the network over time. Here, VisWaterNet is utilized to plot the mean and standard deviation of pressure at each junction in the WDS. The mean pressure value is represented by the color of each node, while the standard deviation is depicted using node size. The code snippet used to generate this plot is shown in Figure 13. Lines 2 and 4 illustrate how the *get_parameter* function retrieves the mean and standard deviation of pressure for each node from the simulation results; line 6 shows the application of the *bin_parameter* function to create interval boundary values for the standard deviation data; lines 8-13 and 20 assign node sizes based on pressure standard deviation; and lines 17-19 assign node colors based on mean pressure. Figure 14 shows the final visualization.

```python
#Get mean pressure at each node
mean, elmnt_list = model.get_parameter("node",parameter="pressure",value="mean")
#Get standard deviation at  each node
stddev, elmnt_list = model.get_parameter("node",parameter="pressure",value="stddev")
#Bin standard deviation values
intervals, interval_names = model.bin_parameter(stddev, elmnt_list, 5)
#Set bin_sizes and create node_sizes array
interval_sizes = [100, 200, 300, 400]
node_sizes = [None]*len(elmnt_list)
#Set node_sizes according to bin_sizes
for interval_name, size in zip(interval_names, interval_sizes):
    for element in intervals[interval_name]:
        node_sizes[elmnt_list.index(element)]=size
# Plot continuous mean data and pass custom node_sizes
model.plot_unique_data(
      ax,parameter="custom_data",
      parameter_type="node",data_type="continuous",
      custom_data_values=[elmnt_list,mean],
      color_bar_title="Mean Pressure (m)",cmap="gist_heat_r",
      node_size=node_sizes,
      element_size_intervals=4,
      element_size_legend_title="Standard Deviation (m)",
      element_size_legend_loc="lower left",
      element_size_legend_labels=interval_names)
```

Figure 13: Example script for generating plot of mean and standard deviation of pressure at junctions.

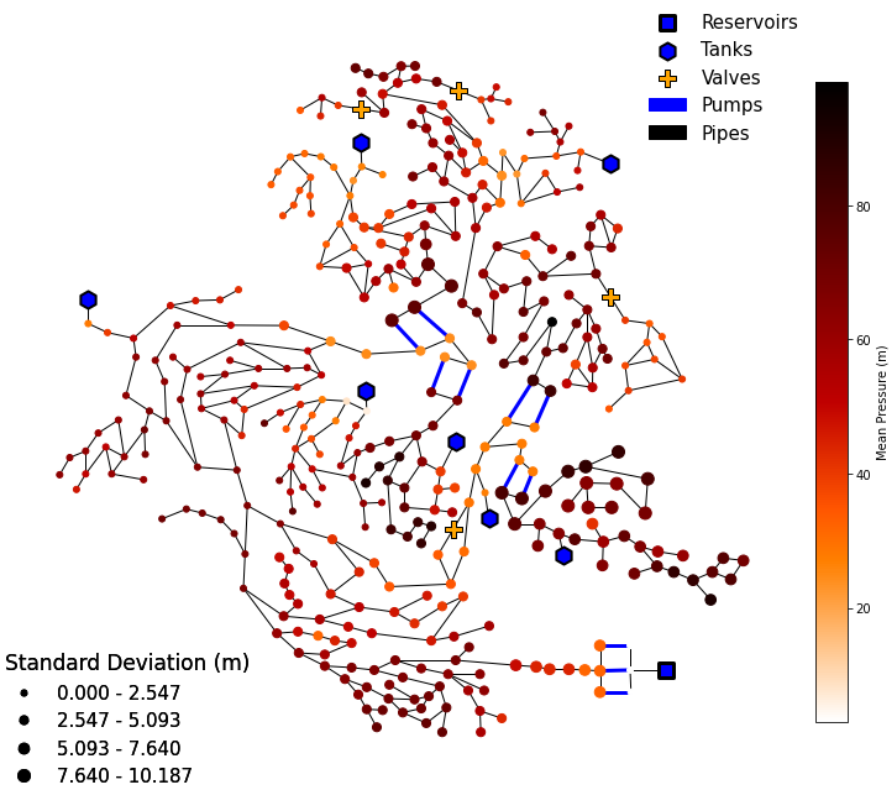

Figure 14. Continuous plot depicting mean and standard deviation of pressure at junctions.

CONCLUSIONS

VisWaterNet can prove to be a useful aide for researchers and practitioners in the field of WDSs. Its range of functions allow visualizing inherent WDS network characteristics as well as simulation results and custom data provided by the user. Enabling the visualization of network properties offers an easy way to contextualize system attributes, which could aid in decision-making processes. Since VisWaterNet is an open-source software, collaboration and contributions from potential users is encouraged and welcome. In the future, it may be possible to extend the functionality of VisWaterNet by adding more input options to each function, as well as features such as creating interactive plots or adding insets to provide more granular information on certain sections of the network. The VisWaterNet codes are under constant development, and the most up-to-date version of all codes and detailed example applications are available in the Github repository: https://github.com/tylertrimble/viswaternet.

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation under grants 1943428 and 1953206.

REFERENCES

Arandia, E., and Eck, B. J. (2018). "An R package for EPANET simulations." *Environmental Modelling Software*. 107, 59-63.

Bastian, M., Heymann, S., and Jacomy, M. (2009). "Gephi: an open source software for exploring and manipulating networks". In: *International AAAI Conference on Weblogs and Social Media*. Association for the Advancement of Artificial Intelligence.

Bentley Systems. (2022). *WaterGEMS V8i User's Guide*. Bentley Systems, Inc., Exton, PA.

De Nooy, W., Mrvar, A., and Batagelj, V. (2018). *Exploratory social network analysis with Pajek: Revised and expanded edition for updated software* (Vol. 46). Cambridge university press.

Eliades, D. G., Kyriakou, M., Vrachimis, S. G., and Polycarpou, M. M. (2016). "EPANET-MATLAB toolkit: An open-source software for interfacing EPANET with MATLAB." In *Proc., Computing and Control for the Water Industry (CCWI)2016*, 1–8. Amsterdam, Netherlands: Elsevier.

Ellson, J., Gansner, E. R., Koutsofios, E., North, S. C., and Woodhull, G. (2004). "Graphviz and Dynagraph—static and dynamic graph drawing tools." In *Graph drawing software* (pp. 127-148). Springer, Berlin, Heidelberg.

Hagberg, A., Schult, D., and Swart, P. (2008). "Exploring Network Structure, Dynamics, and Function using NetworkX." In *Proc., 7th Python in Science Conference (SciPy 2008)*, G Varoquaux, T Vaught, J Millman (Eds.), pp. 11-15.

Harris, C. R., Millman, K. J., and Van Der Walt, S. J. (2020). "Array programming with NumPy." *Nature*, 585(7825), 357-362.

Hunter, J. D. (2007). "Matplotlib: A 2D graphics environment." *Computing in science & engineering*, 9(03), 90-95.

Innovyze. (2022). "InfoWater Pro Help Documentation." Accessed December 16, 2022. https://help.innovyze.com/display/infowaterpro.

Klein, A., Silvester, S., Tanbakuchi, A., Müller, P., Nunez-Iglesias, J., Harfouche, M., and McCormick, M. (2022). imageio/imageio: v2.22.2 (v2.22.2). *Zenodo*.

Klise, K. A., Bynum, M., Moriarty, D., and Murray, R. (2017). "A software framework for assessing the resilience of drinking water systems to disasters with an example earthquake case study." *Environmental Modelling and Software*, 95, 420-431.

Kourbasis, N., Patelis, M., Tsitsifli, S., and Kanakoudis, V. (2020). "Optimizing water age and pressure in drinking water distribution networks." *Environmental Sciences Proceedings*, 2(1), 51.

Ostfeld, A., Salomons, E., Ormsbee, L., and Uber, J. (2012). "Battle of the water calibration networks." *Journal of Water Resources Planning and Management*, 138(5), 523-532.

Rossman, L., Woo, H., Tryby, M., Shang, F., Janke, R., and Haxton, T. (2020). "EPANET 2.2 User Manual." U.S. Environmental Protection Agency, Washington, DC, EPA/600/R-20/133.

Shannon, P., Markiel, A., Ozier, O., Baliga, N. S., Wang, J. T., Ramage, D., Amin, N., Schwikowski, B., and Ideker, T. (2003). "Cytoscape: a software environment for integrated models of biomolecular interaction networks." *Genome research*, 13(11), 2498-2504.