

---

# Predictive Evacuation Mortality Rate Model: Supervised Machine Learning Approach

---

**Alireza Mostafizi**

Department of Civil Engineering  
Oregon State University  
Corvallis, OR 97331  
[mostafia@oregonstate.edu](mailto:mostafia@oregonstate.edu)

## Abstract

Tsunamis have posed a great risk to the coastal communities, especially in Pacific Northwest region. Designing effective evacuation strategies necessitates a perfect understanding over the parameters - walking speed, driving speed, evacuation preparation time, etc. - affecting the mortality rate of any evacuation scenario. Due to inefficiency of statistical models explaining the evacuation process, fitting a predictive model on the evacuation mortality rate data, using machine learning approaches is strongly suggested. The purpose of this work is to apply three different machine learning algorithms - Regularized Linear Regression, Regression Tree, and Neural Network - to fit a predictive model to the mortality rate data generated from the agent-based simulator. The results has shown that Neural Networks work perfectly explaining the parameter affecting the mortality rate, followed by Regression Tree approach, while Linear Regression approach does not work very well in terms of prediction.

## 1 Introduction

Effectiveness of any tsunami evacuation scenario depends on many observed and unobserved parameters, such as average walking speed of the evacuees and its variation, maximum driving speed of the vehicles on the network, the split of the agents driving and walking, the evacuation preparation time and its variation, the shape of the transportation network, location of the shelters and their distance to the hazard zone, population distribution at the time of event, etc. Having perfect understanding on these parameters and the way they affect the mortality rate caused by the hazard is of great importance for designing a scenario that minimizes the number of casualties of the event.

Statistical models fail to explain the relationship of independent and dependent variables, mostly due to complexity of the problem. Therefore, the use of machine learning algorithms seems appropriate for this problem.

For this study, we simulated a near-filed tsunami hitting the city of Seaside, OR. Near-filed tsunamis are expected to come on-shore in less than 30 minutes after the initiation of earthquake. The setting of simulator is described in the next section.

### 1.1 Evacuation Simulator

The agent-based simulator is coded in Netlogo, which is widely used for agent-based modeling and simulation [1]. Using the simulator, one can assess the effect of the most important parameters affecting the mortality rate of a specific scenario. The evacuation speed, the preparedness of the evacuees, and the evacuation mode choice are the factors that have been focused in this study.

Figure 1 shows the screenshot of the platform used to generate the data. The shelters are shown in yellow circles, and the agents find the quickest path using A\* algorithm to their closest shelter. The walking speed is randomly drawn from a normal distribution which its mean and standard deviation will be assessed for this study. Vehicles' movement are calculated based on the car-following model specified by the acceleration and deceleration of the vehicles which are hard-coded in the platform. The speed of the vehicles is limited to the maximum speed. The preparation time is following a Rayleigh distribution which its  $\tau$  and  $\sigma$  are the inputs of the simulator [2]. The mortality rate is measured by the number of agents touched the Tsunami inundation which is hard-coded into the platform, and is based on the study done by Park et al [3].

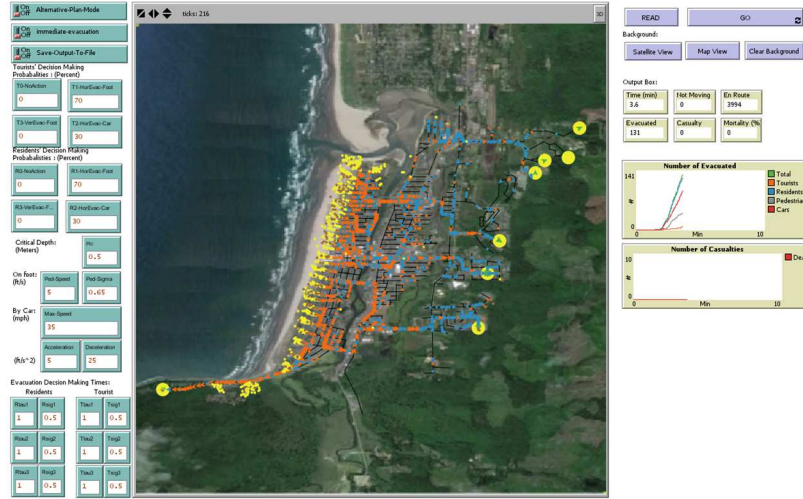


Figure 1: Screenshot of the simulator coded in Netlogo

## 1.2 Data

For the purpose of predictive modeling, the mortality rate data generated using the simulator, varying the following factors:

- The Evacuation Mode Choice Split (Percentage of agents evacuating on foot)
- The Mean Walking Speed
- The standard deviation of Walking Speed
- Maximum Driving Speed
- Minimum Preparation Time ( $\tau$ )
- Variation of Preparation Time ( $\sigma$ )

The mortality rate of 5 repetition for each set of inputs are averaged for each instance. The resulting data includes 540 instances, having 6 independent features as mentioned and 1 dependent variable which is the associated mortality rate.

## 2 Methodology

Based on the nature of the problem, three approaches are appropriate to investigate the prediction problem. The first approach is Regularized Linear Regression which is investigated in two part. First, with only linear features, and second, with added non-linear features. The second approach is the Binary Regression Tree, and the last one is Neural Network. The prediction accuracy of the models are measured by Sum of Squared Errors on both training and testing datasets. For comparison purposes, the learner is trained by 90% (486 instances) of the data, and the rest 10% (54 instances) is used for testing.

## 2.1 Regularized Linear Regression

The linear regression approach was applied to the case in hand, considering different regularizers to avoid overfitting. The formulation of the batch gradient decent, using sum of standard error as an objective function, is as following:

$$E(W) = \frac{1}{2} \left( \sum_{i=1}^N (W^T X_i - y_i)^2 + \lambda \|W\|^q \right)$$

$$\nabla E(W) = \left[ \frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_d} \right]^T$$

$$\frac{\partial E}{\partial w_k} = \sum_{i=1}^N (W^T X_i - y_i) x_{ik} + \frac{q}{2} \lambda w_k^{q-1}$$

$$\nabla E(W) = \sum_{i=1}^N (W^T x_i - y_i) X_i + \frac{q}{2} \lambda W^{q-1}$$

With this in mind, the algorithm can be summarized as following:

$W = W^0$

Repeat {

$$\nabla E(W) = \sum_{i=1}^N (W^T x_i - y_i) X_i + \frac{q}{2} \lambda W$$

$$W \leftarrow W - \alpha \nabla E(W)$$

} Until  $|\nabla E(W)| < \epsilon$

Where  $\lambda$  is the regularization coefficient, and  $\alpha$  is the learning rate. Both hyper-parameters are tuned as following to achieve the fastest and the most accurate predictive model. The  $\epsilon$  is set to  $10^{-4}$ . In addition, Lasso, Quadratic, and L-4 regularizer has been tested, and the best regularizer was picked at the end. It also has to be mentioned that the data has been normalized to the range of 0 to 1 for faster and better convergence.

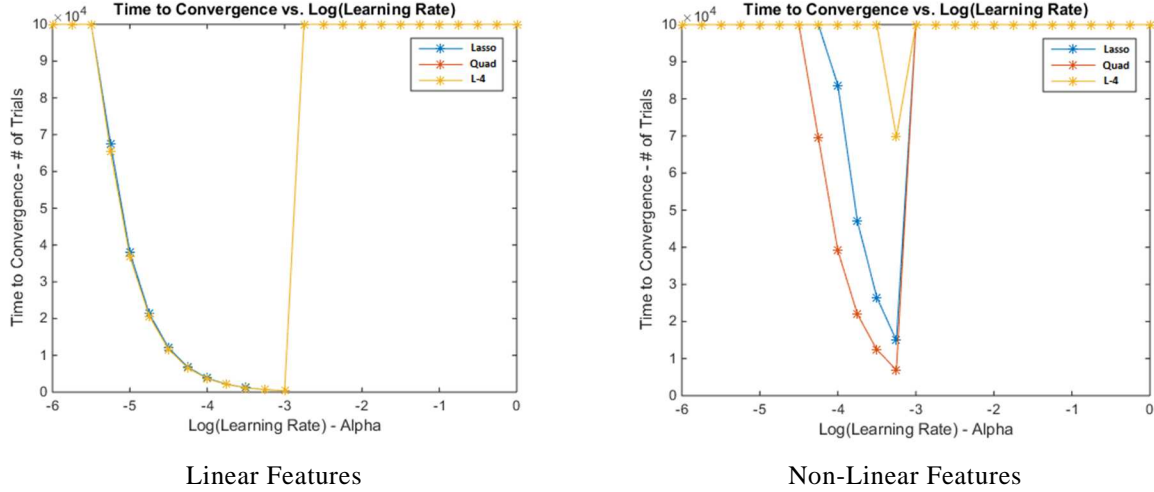


Figure 2: Time to convergence vs. Log Learning Rate

### 2.1.1 Linear Features

For the first trial, only linear features mentioned in the previous sections (6 features) were tested in the model. As shown in Figure 2, learning rate of 0.001 has the fastest convergence rate for all the regularizers. In addition, Figure 3, shows the variation of sum of standard errors for both training, and testing datasets. As explained by the figure, Lasso regularizer with  $\log(\lambda)$  of 0.5 leads to the lowest error on testing dataset, and therefore,  $\alpha = 0.001$  and  $\lambda = 3.162$  is the most appropriate parameters in this case.

The results of this model will be discussed in the discussion section.

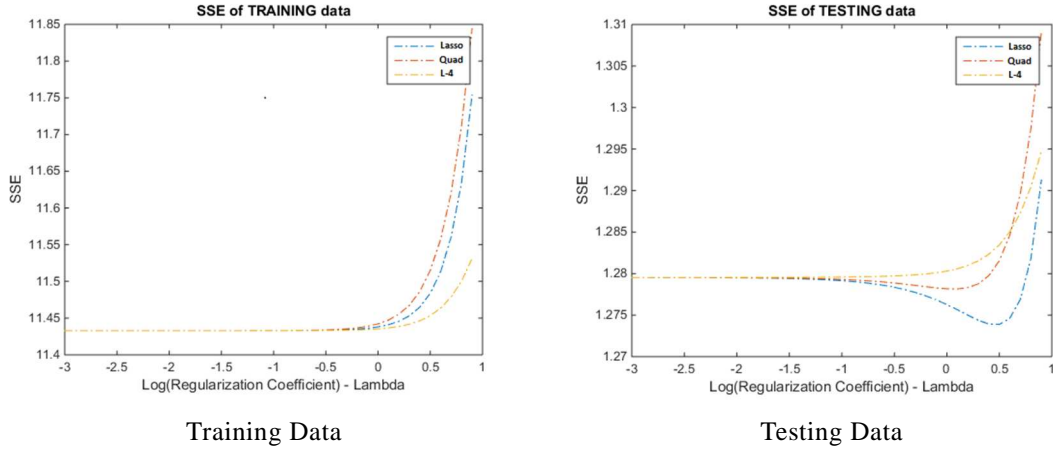


Figure 3: Sum Squared Error vs. Regularization Coefficient (Linear Features)

### 2.1.2 Non-Linear Features

Since using linear features were not adequately predictive, on top of the linear features, logarithm and squared of the features were added to the model, reaching to 17 features in total. Similarly to the previous part, the parameters have been tuned. Figure 2 represents the learning rate which leads to the fastest convergence. For this case,  $\log(\alpha) = -3.2$  provides the fastest convergence. Moreover, Figure 4 shows the process that leads to tuning of the regularization coefficient.

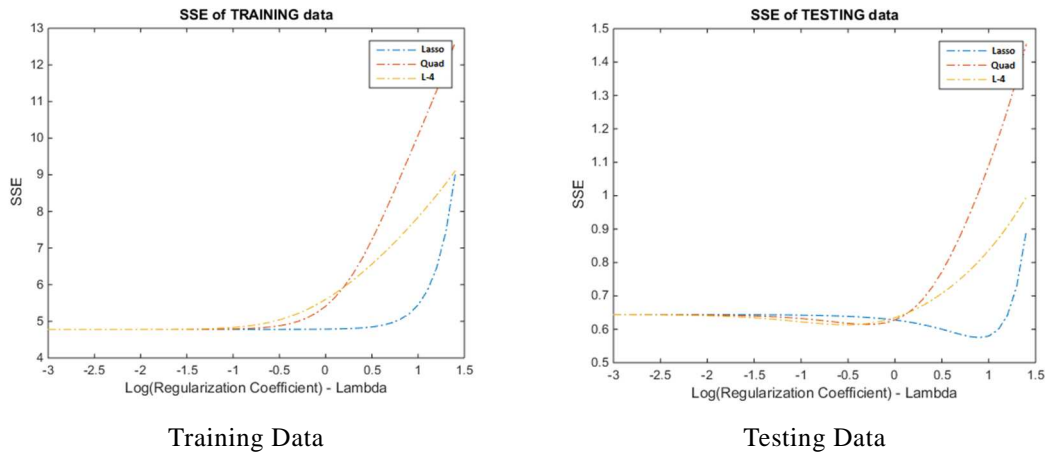


Figure 4: Sum Squared Error vs. Regularization Coefficient (Non-Linear Features)

As explained by the figure, Lasso regularizer with  $\log(\lambda) = 1$  has led to the lowest error, and thus,  $\alpha = 6.3 \times 10^{-4}$ , and  $\lambda = 10$  are the tuned parameters for Lasso Regularized Linear Regression model on the non-linear features.

## 2.2 Regression Tree

The second approach applied to this problem is Binary Regression Decision Tree. Regression Trees are expected to explain the complex data fairly accurately as the tree grows. Figure 5 shows the sum of squared errors of both training and testing datasets as the tree grows deeper. The size of the tree has been controlled with the number of splits of the tree. As you can see, the error decreases rapidly as tree grows larger, to the extent that a tree which has 50 splits, performs fairly well in terms of prediction.

The node splitting approach for the tree growth is as following:

- Compute the mean squared error for each node, using the following equation where  $T$  represents the node:

$$\epsilon_t = \sum_{j \in T} (y_j - \bar{y}_t)^2$$

- On all nodes, try all the predictors and all their cut points, and find the one that maximizes the following function:

$$\Delta I = n_t \epsilon_t - n_l \epsilon_l - n_r \epsilon_r$$

Where  $n$  = Number of Instances in each node

$\epsilon$  = MSE of the node

$l$  and  $r$  represent the left and the right child respectively, and  $t$  represents the parent node.

- Stop if reached to maximum number of splits allowed.

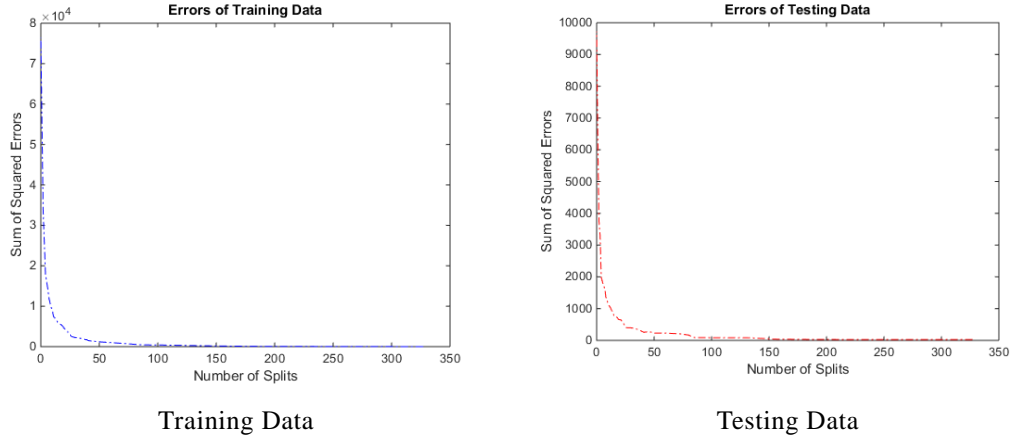


Figure 5: Sum Squared Error vs. Number of Splits (Tree Size)

Therefore, a Tree with the number of splits of roughly 50 has been used to compare the results of three models. Figure 6 shows the schematic representation of the tree applied to the data.

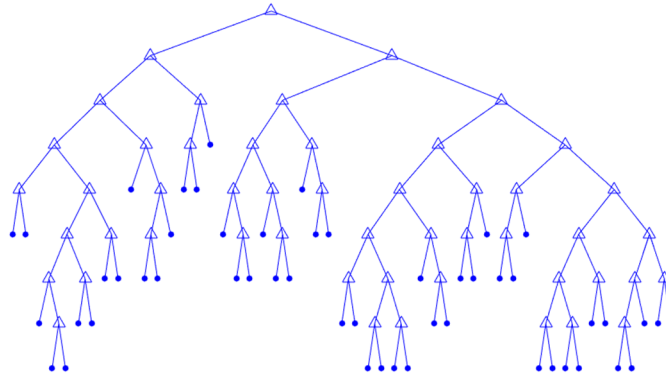


Figure 6: Schematic Representation of the Regression Tree applied to the data

The mentioned tree has 49 splits and 51 leaf nodes. The performance of this tree will be compared to the other approaches in the last section.

## 2.3 Neural Network

The third algorithm implemented on this problem is Neural Network. I implemented single layer neural networks with different number of neurons using MatLab Neural Network Toolbox. *Levenberg-Marquardt* backpropagation is used as network training function. Figure 7 shows the averaged performance functions (mean standard error of the datasets) over 5 repetitions as the number of neurons increase.

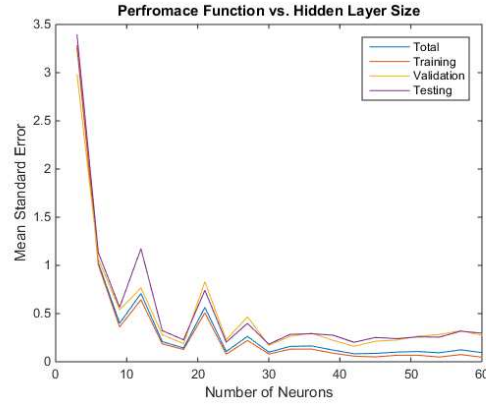


Figure 7: Performance Function vs. Number of Neurons

Based on the figure, it can be stated that with 50 neurons the network performs very well in terms of explaining the data. Therefore, for comparison purposes a 50-neuron with the schematic representation as shown in Figure 8 is applied to the data, and the prediction results are compared to that of other approaches in the results section.

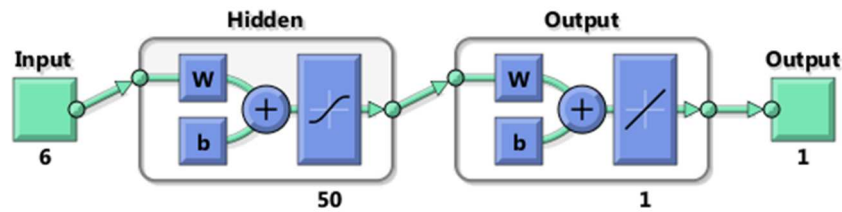


Figure 8: Schematic Representation of the Neural Network

Figure 9, in addition, shows the error on training, validation, and testing data set. The training has stopped at epoch 36 when the validation error stopped decreasing.

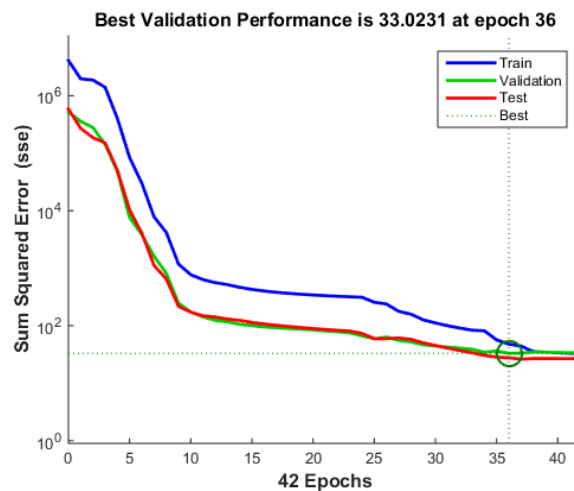
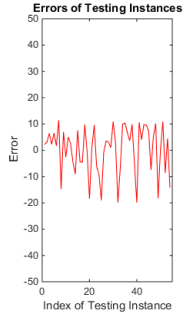
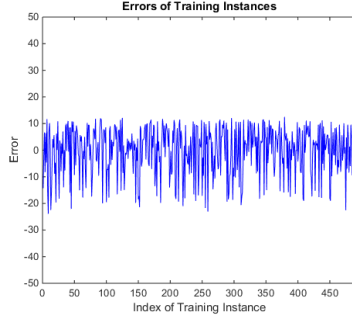
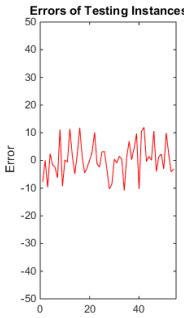
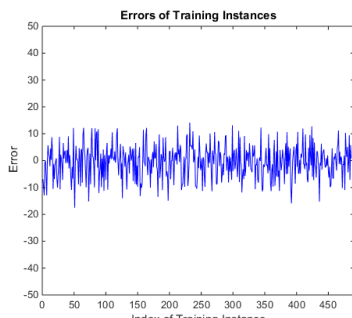
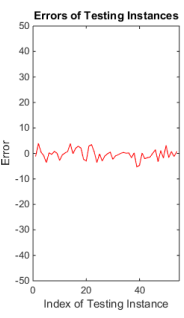
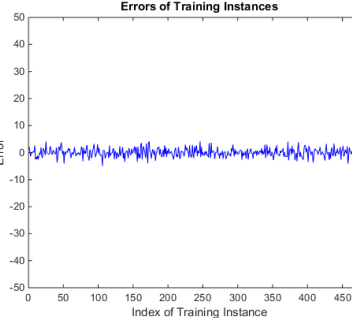
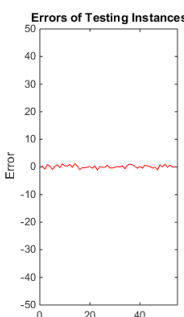
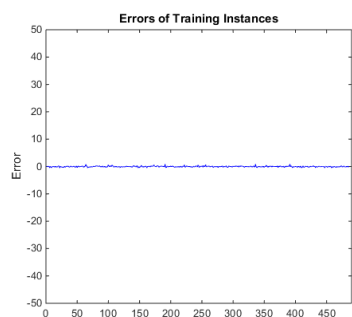


Figure 9: Sum squared errors vs. Number of epochs

### 3 Results and Discussion

This section shows the final comparison between three approaches took to fit a predictive model on the data. Table 1 shows the prediction errors of both training dataset (486 instances) and testing dataset (54 instances) for all three approaches.

Table 1. Prediction Error on datasets for each algorithm

Approach		Prediction Error	
		Testing Dataset	Training Dataset
Regularized Linear Regression	Linear Features		
	Non-Linear Features		
Binary Regression Tree			
Neural Network			

In addition, Table 2 shows the sum of squared errors over both training and testing dataset.

Table 2. Performance of approaches

Algorithms		Sum of Squared Errors	
		Testing Dataset	Training Dataset
Lasso Regularized Linear Regression	Linear Features	4.43e+03	3.99e+04
	Non-Linear Features	2.00e+03	1.81e+04
Binary Regression Tree (Max 50 Splits)		228.4289	1.20e+03
Neural Network (50 Neurons)		18.20	29.46

The above results show that neural network work perfectly in terms of prediction. On the other hand, linear regression, even with slight improvement with adding non-linear features, does not perform very well.

The prediction using linear regression with lasso regularizer, without non-linear features, may result in a prediction up to 20 percent off the true value. This range decreases to 10 percent error by adding non-linear features.

Regression Trees are expected to converge to zero training error as the tree grows deeper. Here we see that a tree with only 50 splits, improves the prediction results to 5 percent which, compared to the results of linear regression, improves the performance of predictive model significantly.

Finally, Neural Network works great in terms of prediction. Most of the prediction are off at most by 0.5 percent from the true value which is fairly close to the real data.

## 5 Conclusion

In this work, we applied three different supervised machine learning algorithm to the mortality rate data generated from the agent-based simulator, with the aim of fitting a predictive model, capturing the effect of evacuation mode choice, evacuation speed (e.g. walking and driving speed), and evacuation preparation time on the mortality rate of the evacuation scenario. The results have shown that although, linear regression models, even when incorporating non-linear features, fail to fairly predict the mortality rates, Neural Networks are highly predictive with less than 0.5% error. This reinforces the fact that Neural Networks work well in capturing complex relationship between variables. The performance of Binary Regression Tree is assessed to be somewhere in between with up to 5% prediction error.

## References

- [1] Wilensky, U. 1999. NetLogo. <http://ccl.northwestern.edu/netlogo/>. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.
- [2] Haizhong Wang, Alireza Mostafizi, Lori A. Cramer, Dan Cox, and Hyoungsu Park. An agent-based modeling of a multimodal near-field tsunami evacuation: Decision-making and life safety. Transportation Research Part C: Emerging Technologies, 2015.
- [3] Hyoungsu Park, Dan Cox, Patrick J. Lynett, Dane M. Wiebe, and Sungwon Shin. Tsunami inundation modeling in constructed environments: A physical and numerical comparison of free-surface elevation, velocity, and momentum flux. Coastal Engineering, 79:9–21, 2013.