Comparative Analysis of Two Forest-based Regression Algorithms

MAJ John R. Brence, Ph.D.
Adjunct Assistant Professor
Department of Systems Engineering
United States Military Academy
West Point, NY 10996
845-304-6416
john.brence@usma.edu

Donald E. Brown, Ph.D.
Department Chair and Professor
Department of Systems and Information Engineering
University of Virginia
Charlottesville, VA 22903
434-924-5393
brown@virginia.edu

ABSTRACT

A comparative analysis of two forest-based regression algorithms is an in-depth investigation of the merits of Random Forest Regression (RFR) and Robust Random Forest Regression (RRFR). In previous research, we determined that RRFR was more robust than RFR to unbounded outliers and heteroscedastic datasets using a DFfits style analysis. The study's goal is to compare predictive performance of RFR and RRFR using both historic and synthetic datasets. In addition, we provide additional analysis into possible modifications of major components of these algorithms as well as provide comment on interesting results found from our experiment. We compare the performance of these algorithms using test dataset mean squared error (MSE) and mean absolute deviation (MAD).

KEY WORDS

Random Forest, Outlier, Robust Statistics, Regression, Tree-based Methods

1 INTRODUCTION

Random Forests were introduced in 2000 by Breiman [2]. This algorithm creates a classification tree forest. A single tree is created by using a bootstrap sample of the training dataset and bagging predictors in order to facilitate more efficient node-splitting. Bagging (Bootstrap Aggregation) is the generation of multiple (random) versions of a predictor in order to combine into an aggregated prediction. Bagging creates the forest prediction by combining numerous trees in order to take advantage of their predictive power [1]. Use of the Strong Law of Large Numbers indicates that the trees always converge (only if the mean is finite [exists]); therefore there is no issue with overfitting the data [1].

In April of 2001, Breiman introduced code for Random Forest Regression (RFR). This method mirrored the previous classification algorithm (Random Forests); however, it predicted and calculated error using the terminal node means and mean squared error. The thesis of this method is that complex predictors provide a wealth of interpretable scientific information about

the data and how it is predicted. In addition, among the current prediction methods, the most complex methods are the most accurate [2].

Robust Random Forest Regression embeds the methodology of Random Forest Regression [1] with a major difference ~ the introduction of robust prediction and error statistics using the median and other robust measures for prediction, and the use of mean absolute deviation (MAD) to calculate both the in-node and overall error. By adapting this new strategy, we theoretically lessen the effects of outliers and heteroscedasticity while retaining the beneficial components of RFR as well. RFR currently uses the node mean for prediction and mean squared error (MSE) to derive the in-node and overall error.

Definition: A random forest (regression) is a predictor consisting of a collection of tree-structure predictors $\left\{h(\mathbf{x},\Theta_z)_\psi, z=1...Z\right\}$ in which each tree casts an equal valued vote for the prediction at input vector $\mathbf{x}$ and where Θ_z are independently identically distributed random vectors whose elements are counts on the number of times an input row appears in the bootstrap sample [1]. z is the index for the tree in the forest and Z is the total number of trees in the forest. ψ is a statistic for central tendency, which could be the mean, median, or other measure.

For the z^{th} tree, we generate a random vector Θ_z, independent of the previous random vectors $\Theta_1,...,\Theta_{z-1}$ but with the same distribution; and we grow a tree using the training set and Θ_z, resulting in the predictor $h(\mathbf{x},\Theta_z)_\psi$ where $\mathbf{x}$ is an input vector [1].

The prediction based on the vectors $\mathbf{x},\Theta_z$ is conducted in two steps. First, at the tree level, the prediction $h(\mathbf{x},\Theta_z)_\psi$ is based on the statistic ψ chosen to determine central tendency of the observations within each terminal node ~ mean for RFR and median, trimean, broadened median or trimmed mean for RRFR. Second, all the tree predictions are combined to create a forest prediction $\left\{h(\mathbf{x},\Theta_z)_\psi, z=1,...Z\right\}$ using *bagging* (bootstrap-aggregation).

2 EXPERIMENTAL PLAN

The intent of the experimental plan is threefold: 1) to compare the RRFR algorithm to Breiman's RFR using the identical datasets to those employed by Breiman [1], 2) to examine and explore RRFR's predictive capability on the complex corrosion dataset used previously in Brence [3] and 3) to determine the capabilities and limitations of RRFR by tuning a select set of model parameters. These results are used to extend and generalize the model's capabilities to address a larger class of problems as well.

2.1 Experimental Parameters and Evaluation Criterion

There are five factors that are varied in this experiment: Prediction method, terminal node size threshold, number of trees in a forest, the percentage of independent variables tried at each split, and the dataset used to build the training and test datasets. Each run consists of ten replicates.

The prediction method factor has eight parameters for testing. The experiment includes the mean, median, broadened median, trimean, and four trimmed means (α =0.1, 0.2, 0.3, 0.4). The mean parameter is tested using Breiman's RFR which uses MSE to create node splits. The other parameters are tested using RRFR that implements MAD to create node splits. In addition, both codes output MSE and MAD for model comparison.

The terminal node threshold size, $N_{m_{threshold}}$, is a factor that sets a terminal node's population size. For example, if the threshold is five, a node is terminal on the first instance when the number of observations is five or less. This factor has three levels for testing: five, ten, and twenty. The default size used in Breiman's code was five.

The number of trees in a forest, Z, were varied in order to test algorithm performance for small, medium, and larger forest values. This is done by setting Z equal to 50, 100, and 200 respectively. Initially, the value of 300 was tested as the larger tree value, but due to the size of some of the datasets (especially those over 1,000 observations); running 300 trees per forest caused the computer to crash while doing simultaneous replicates or was very time consuming (over a week for some runs). Upon comparison of the results from a run with 200 trees versus 300 trees there was minimal improvement with the increase wait time; therefore, 200 trees were deemed reasonable for this study.

The number of independent variables tried at each split I_m determines the number of predictor variables to try for each parent node split. Since each dataset has a different number of independent variables, we chose to express this parameter in percentage at three different levels: 25%, 50%, and 75%, in order to capture low, medium, and high values for subset selection. For example, 25% indicates that a random selection of 25% of the possible predictor variables are tried for each parent node split.

The last factor identifies the different datasets used in the experiment. This factor has eleven different levels. The datasets used in this experiment are Brence1, Brence2, Brence3, Corrosion, Abalone, Boston Housing, Ozone, Servo, Friedman1, Friedman2, and Friedman3. The origins and descriptions of these datasets are set forth in section 2.2.

The five factors at various levels (unbalanced design) results in a full factorial model with 2,376 trials (23,760 overall individual runs based on 10 replications of each parameter combination). Center points, where applicable, are manually input so that each factor has at least three levels. These center points are used in order to test for non-linearities. Traditional design of experiment methods using center points (e.g. central composite) are not used since this design is unbalanced.

Our evaluation criterion is based on numerical accuracy, defined as how well, measured in terms of error rate, the algorithm predicts the true answer for each observation. A lower error rate indicates better performance. For this experiment, *all* the prediction methods are compared using both MSE and MAD. Each of these summary error rate measures are calculated based on an average of the replicates reported for each run on the test dataset.

2.2 Experimental Datasets

The datasets chosen for this study were Brence1, Brence2, Brence3, Corrosion, Abalone, Boston Housing, Ozone, Servo, Friedman1, Friedman2, and Friedman3. The Brence datasets are synthetic datasets derived especially for this study. The corrosion dataset is a scientific dataset from non-destructive tests used in Brence [3]. The remaining seven datasets correspond to those

used in Breiman's RFR study [1]. For comparison purposes, we will mirror Breiman's experiments on these datasets by using the same training and test dataset sample sizes.

Brence1 is a synthetic dataset based on a normal (Gaussian) distribution of the response variable, Y and $y_i = N(50.52,10.27) + \varepsilon_i$. This dataset contains 250 observations and eight independent (predictor) variables. The predictor variables were calculated based on deterministic functions of the normalized response (Table 1). Noise from a uniform (0, 1) distribution was superimposed on the model response values. The resulting response dataset is then post-processed so that each response value has a 0.05 of being an outlier. Notationally, we use *Y'* to represent the final response variable for this dataset. Outlier observations were created by adding or subtracting $3\sigma_Y$ (three times the response standard deviation). This dataset was used to test the prediction capability of the central tendency based on a Gaussian distribution with outliers.

Variable	Data Type	Description
x_1	continuous	$3y_i + 5.32\varepsilon_i$
x_2	continuous	$1,200 + y_i / 2 - 1.52\varepsilon_i$
x_3	continuous	$y_i^2 - 3.04\varepsilon_i$
x_4	continuous	$-500 + 2y_i - 5.32\varepsilon_i$
x_5	continuous	$219{,}298 / y_i\varepsilon_i$
x_6	continuous	$Ln(y_i^2) / 1.52\varepsilon_i$
x_7	continuous	$0.4482 y_i\varepsilon_i$
x_8	continuous	$y_i + 0.544\varepsilon_i$
Y'	continuous	Normally distributed with noise and 12 outliers

Table 1: Brence1 Variable Descriptions

This dataset has twelve synthetic outliers created in the fashion described. For this experiment the training dataset is comprised of 200 observations and the test dataset is a random sample of 50 observations representing 20% of the dataset.

The Brence2 dataset is setup in a similiar manner as the Brence1 dataset; however, the original response variable, *Y*, is generating using a Cauchy distribution or $y_i = Cauchy(49.37, 7.27) + \varepsilon_i$. This distribution is a symmetric distribution with heavy tails and a single peak at the center. The Cauchy distribution possesses an interesting property in that collecting more data does not lead to better accuracy in estimating the population mean. That is, the sampling distribution of the mean is equivalent to the sampling distribution of the original data [10]. Therefore, the mean is totally inapplicable to measure central tendency and other methods must be used [5].

This dataset is comprised of eight predictor variables, one response, and 250 observations. Calculation of the predictor variables are slightly varied from those shown in Table 1. For testing purposes, the training dataset is comprised of 200 observations and the test dataset is a random sample of 50 observations, or 20% of the dataset.

The Brence3 dataset mirrors the set-up of the two previous Brence datasets except that the original response variable, *Y*, is generated using a Double Exponential or Laplace distribution or $y_i = Laplace(16.00, 2.84) + \varepsilon_i$. The Laplace distribution was chosen in order to test predictability on a dataset whose response has a distribution with a sharp central peak and long

tails. In addition, this distribution has a unique characteristic in that the mean, median, and mode are equivalent [10]. See [5] for full description of this distribution. This dataset is comprised of eight predictor variables, one response, and 250 observations. For testing purposes, the training dataset is comprised of 200 observations and the test dataset is a random sample of 50 observations, or 20% of the dataset.

This dataset is comprised of eight predictor variables, one response, and 250 observations. Calculation of the predictor variables are slightly varied from those shown in Table 1. This dataset was built with fourteen response outliers.

The Corrosion dataset originates from a non-destructive test study done by the Institute of Aerospace Research (IAR), National Research Council, Canada [7] and modeled in a study by Brence [3]. It contains the induced voltage readings, based on different scan frequencies (predictors), from an eddy current non-destructive test and the known material loss percentages (response) of the specimen scanned.

This dataset consists of 160,608 observations and four predictor variables. The response of this data is percentage material loss, a surrogate measure for corrosion damage. Table 2 describes the variables that are used in this dataset.

Variable	Name	Data Type	Units	Description
x_1	5.5 kHz scan	continuous	volts	induced voltage from eddy current scan
x_2	8 kHz scan	continuous	volts	induced voltage from eddy current scan
x_3	17 kHz scan	continuous	volts	induced voltage from eddy current scan
x_4	30kHz scan	continuous	volts	induced voltage from eddy current scan
Y	loss	ordinal	%	Material loss in the specimen

Table 2: Corrosion Dataset Variable Descriptions

This is the largest dataset in the study. Of the 160,608 observations, a random sample of 20,000 is used to construct both the training and test datasets. The training dataset contains 10,000 observations and the test dataset incorporates 10,000 randomly selected observations.

The other datasets used in this study are discussed in detail in various other works. The Abalone, Boston Housing, Ozone, and Servo are explained in [1] and Friedman1, Friedman2, and Friedman3 are datasets used in [1] and defined in [6]. We use the same specifications for training and test datasets as Breiman [1]; however, since we used random sampling to determine the observations in the training and test datasets, the results may vary somewhat from Breiman's study.

3 EXPERIMENTAL RESULTS

In general, we found that the more trees in a forest, the greater the percentage of independent variables tried at each split, and smallest terminal node size threshold was the best combination of parameters to complement a prediction method.

Table 3 is a summary table for performance of the prediction methods on all the datasets used in this study. The areas shaded in green show the criterion where a prediction method from RRFR outperformed the mean prediction method used in the RFR algorithm. For the most part, RFR outperformed RRFR which seems counter-intuitive based on the noisy, outlier-ridden datasets presented in this study.

Best Performances Roll-up		
	MSE_Tst	MAD_Tst
Abalone	Mean	Mean
Boston	Mean	Mean
Brence1	Mean	Mean
Brence2	Mean	Trim Mean 0.3
Brence3	Mean	Mean
Corrosion	Mean	Mean
Friedman1	Mean	Mean
Friedman2	Mean	Mean
Friedman3	Mean	Trimean
Ozone	Mean	Mean
Servo	Mean	Mean

Table 3: Best Performance Roll-up

In his paper [1, pp. 23-24], Breiman provided the results of his experiment for several datasets that were used in this study. For comparison purposes, Table 4 displays our best performance results versus Breiman's report. The green-shaded cells show which study provided the better model, understanding that random selection of data-splitting for training and test set datasets may differ and Breiman was not as exhaustive in his model parameter search. This comparison is based mainly on the results of RFR since 1) it always performed best in our study for the MSE test set criterion and 2) Breiman did not report the MAD test set error.

Dataset Modeling Results (MSE_test set)		
	Brence	Breiman
Abalone	8.021	4.600
Boston	5.648	10.200
Friedman1	2.370	5.700
Friedman2	60,387.400	19,600.000
Friedman3	0.104	0.022
Ozone	11.790	16.300
Servo	0.585	0.246

Table 4: Comparison of Results with Breiman's Study

4 ADDITIONAL ANALYSIS

This section discusses interesting results and provide some explanation(s) on why these events transpired. In addition, this section provides some deeper analysis into possible areas for modeling improvement and/or comments on the how an algorithm came up with a solution.

4.1 Was a better corrosion model found based on this study?

The new baseline for the previous corrosion study [3] was a $\sqrt{MSE}$ of 3.74 for the Least Squares (LS) regression tree algorithm. This result was achieved using the original 60,000 observations training dataset and 60,000 observation test dataset. Our current models, reported in section 2.2 use a smaller training and test dataset observation size of 10,000 each.

From our research, we applied the RFR code to this dataset using the parameters: $N_{m_{threshold}} = 5$ (terminal node threshold), $Z = 500$ (number of trees in forest) and $I_m = 0.75I$ or 3 variables (number of predictor variables considered at each split). The number of trees in the forest was the only parameter that we could reasonably change beyond our current study parameters without crashing the computers. The increased observations severely hindered the computer's operation.

The result of our run improved the prediction of the corrosion dataset from a $\sqrt{MSE}$ of 3.74 to a $\sqrt{MSE}$ of 3.04 or a relative improvement of 18.8%. This result would significantly improve the capability to determine corrosion problems based on the eddy current scan and allow the technician to better perform his job. The improvement of 18.8% will improve safety, decrease the chances for catastrophic events, and potentially save lives. Money expenditures may increase or decrease depending on new policies derived to improve maintenance or change operational requirements.

4.2 Why did RFR and RRFR overfit on occasion?

Definition: Overfitting: Specifically fitting the training data thereby not achieving a general model. The outcome of an overfit model is that the training error is typically much smaller than the error result from applying a test dataset to the model.

In Breiman's paper [1] and our previous study [4], the argument of why the random forest regression algorithms do not overfit was supported by the law of large numbers and the central limit theorem. However, even Breiman found a result in his study where the training error was better than the test set error, "I believe this is an artifact caused by separate training and test sets, where the test set may have a slightly higher error rate than the training set [1]." This seems like an obvious conclusion, but does not get to the root of the error or explain why this result goes against his earlier "does not overfit" claim.

We propose that there is an issue with a tacit assumption that comments on this occurrence on three levels: In order to support this claim, the test dataset is either 1) homogeneous with the training dataset, or 2) its values at least follow a similar structure, or 3) it values fall within the bounds of the training dataset (no outliers). We found that bagging did a good job minimizing the effects of outliers in the training dataset; however, when a test dataset is applied to the current model, it is in its raw form. If there are outliers in the test dataset or abnormalities that are not included in the training dataset, the test set error will more than likely be higher than the training error.

In this study, we found that both RFR and RRFR overfit when applied to the Brence3, Corrosion, and Friedman1 datasets. Brence3 was known to have one very extreme synthetic outlier (and thirteen others), while the Corrosion and Friedman1 datasets were plagued with very noisy observations. What we found was that the Brence3 model overfits mainly because of a

random selection of several outliers for the test dataset, and we suppose that both the Corrosion and Friedman datasets are overfit due to an increased amount of noise in the test dataset or a combination of heterogeneity, structural dissimilarity, and a few out-of-bounds observations. The latter conclusion is more difficult to surmise since the test datasets are relatively large ($N \geq 2000$) and comparison-plotting does not clearly show a verifiable result.

In order to verify the result that the training datasets were not effective in their modeling, we chose to run the RFR algorithm switching the functions of the previous training and test datasets. The Corrosion model maintained 10,000 observations in both training and test dataset, the Friedman1 increased its training dataset to 2,000 observations and decreased the test dataset to 200, and the Brence3 dataset decreased its training dataset to 50 observations and increased its test dataset to 200. We comment on the dataset sizes because the general thought is better performance should come from a larger training dataset.

We found that the Corrosion model was no longer overfitting and improved in both the MSE and MAD test set criteria while the training error criteria decreased performance. From previous conjecture, this shows that the old test dataset was more general in nature and creates a better prediction model.

The Friedman1 model was very interesting. The Friedman1 model no longer overfit. For all of the twenty-seven runs, the new 2,000 observation training dataset performed better in all of the training and test set criteria. This result also validates previous suppositions.

The Brence3 model no longer overfit and revealed remarkable results. Generally a model that has 50 training observations and 200 test observations, including an extreme outlier, should perform badly. However, we found that in eighteen of the twenty-seven new runs there was improvement in the MSE test criterion. The nine exceptions came from the runs that had a terminal node threshold of twenty, where the tree models would at most include a four level tree with six terminal nodes. Even with this handicap, the model was reasonably close to the old model. This result enforces that the few outliers included in the original test dataset were more influential than the one extreme outlier, since the extreme outlier, included in the test dataset, did not hamper the model's performance. Never did the new runs outperform the old MAD test criterion results.

Table 5 shows some relative improvement in predictability from previously reported results. The Friedman1 model shows the most noteworthy improvement. In addition, with the switching of the datasets (training $\leftrightarrow$ test), there were some better predictive RFR parameter combinations. This result may be expected due to the subtle, but different characteristics of the new training datasets.

Example Predictive Improvements from Validation Exercise						MSE_Tst	
Dataset	**Run**	**Pred**	**$N_{mthresh}$**	**Z**	**I_m**	**Old Best**	**Improved**
Brence3	1665	Mean	10	200	0.75	5.140	4.935
Corrosion	1497	Mean	5	200	0.75	14.499	13.165
Friedman1	1936	Mean	5	100	0.75	2.370	0.211

Table 5: Example Predictive Improvement from Overfitting Validation Exercise

4.3 How well do we predict the extreme points (outliers)?

This section investigates how well the RFR and RRFR algorithms predict outliers in the data. The Brence1 and Brence2 datasets are used in this exploration because the outliers in each

dataset are synthetic as explained in section 2.2. The Brence1 and Brence2 datasets have twelve and fourteen synthetic outliers, respectively. There is one outlier in the Brence1 test dataset and two in the Brence2 test dataset.

The two best performances of RFR and RRFR are compared for each dataset. The method of comparison is based on whether RFR or RRFR (median) can capture the true values of the outlier's response within a 95% confidence interval (95% CI). The cells shaded in red are where the response falls outside the 95% CI. In addition, for each outlier response, a distance calculation is made from the closest 95% CI bound, where a positive distance is outside of the 95% CI and a negative distance is inside the 95% CI. These distances are summed to show a total distance for all the outlier predictions.

Table 6 shows the RFR algorithm's capability to model the twelve outliers in the Brence1 dataset. For every synthetic outlier created, the true response fell outside of the 95% CI of RFR's prediction for both training and test datasets. The total distance for all outliers from the 95% CI was 90.207.

RFR Outlier Predictions based on 95% Confidence Interval ~ Brence1					
Training Model MSE		58.133	Root_MSE	7.624	
Obs #	**Y pred**	**Y obs**	**LO_95% CI**	**HI_95% CI**	**Distance**
19	53.530	22.670	30.657	76.403	7.987
35	45.179	83.260	22.306	68.052	15.208
46	56.727	28.920	33.854	79.600	4.934
54	50.332	20.460	27.459	73.205	6.999
57	60.140	34.690	37.267	83.013	2.577
58	51.446	21.300	28.573	74.319	7.273
71	47.403	77.760	24.530	70.276	7.484
110	45.077	14.020	22.204	67.950	8.184
126	57.416	25.210	34.543	80.289	9.333
130	60.735	34.350	37.862	83.608	3.512
185	49.198	75.570	26.325	72.071	3.499
Test Model MSE		24.056	Root_MSE	4.905	
34	57.046	29.110	42.332	71.760	13.222
Total Distance outside of 95% CI					90.207

Table 6: RFR Outlier Predictions based on 95% Confidence Interval ~Brence1

Table 7 shows the RRFR result for the Brence1 dataset. First of all, notice that outlier observation 130 falls within the RRFR's predictive 95% CI. Secondly, the overall distance from the 95% CI of the RRFR predictions is 57.728, which is about 64% better than RFR. Furthermore, the RRFR algorithm does a better job adjusting its predictions to take into account the test dataset outlier ~ a distance of 6.335 < 13.222.

RRFR Outlier Predictions based on 95% Confidence Interval ~ Brence1					
Training Model MSE		64.743	Root_MSE	8.046	
Obs #	**Y pred**	**Y obs**	**LO_95% CI**	**HI_95% CI**	**Distance**
19	50.111	22.670	25.972	74.250	3.302
35	52.347	83.260	28.208	76.486	6.774
46	57.303	28.920	33.164	81.442	4.244
54	48.606	20.460	24.467	72.745	4.007
57	61.161	34.690	37.022	85.300	2.332
58	48.561	21.300	24.422	72.700	3.122
71	46.228	77.760	22.089	70.367	7.393
110	48.457	14.020	24.318	72.596	10.298
126	56.013	25.210	31.874	80.152	6.664
130	58.468	34.350	34.329	82.607	-0.021
185	48.154	75.570	24.015	72.293	3.277
Test Model MSE		37.876	Root_MSE	6.154	
34	53.908	29.110	35.445	72.371	6.335
Total Distance outside of 95% CI					57.728

Table 7: RRFR Outlier Predictions based on 95% Confidence Interval ~Brence1

The results for the Brence2 dataset for the RFR algorithm are displayed in Table 8. Again, for every synthetic outlier derived for this dataset, RFR cannot maintain the response with a 95% CI of its prediction. Of note is outlier observation 87, the extreme and only negatively-responsed outlier in this dataset. The total distance from the 95% CI for RFR with the Brence2 dataset is 67.859.

RFR Outlier Predictions based on 95% Confidence Interval ~ Brence2					
Training Model MSE		37.625	Root_MSE	6.134	
Obs #	**Y pred**	**Y obs**	**LO_95% CI**	**HI_95% CI**	**Distance**
10	48.337	69.590	29.935	66.739	2.851
13	52.018	74.170	33.616	70.420	3.750
69	53.535	72.340	35.133	71.937	0.403
87	23.344	-22.300	4.942	41.746	27.242
90	51.879	30.680	33.477	70.281	2.797
108	50.485	71.280	32.083	68.887	2.393
124	49.605	29.610	31.203	68.007	1.593
151	53.683	73.660	35.281	72.085	1.575
182	48.386	69.830	29.984	66.788	3.042
189	46.088	24.460	27.686	64.490	3.226
195	50.822	31.210	32.420	69.224	1.210
199	49.762	28.360	31.360	68.164	3.000
Test Model MSE		19.012	Root_MSE	4.360	
19	48.762	29.060	35.681	61.843	6.621
20	50.872	29.640	37.791	63.953	8.151
Total Distance outside of 95% CI					67.859

Table 8: RFR Outlier Predictions based on 95% Confidence Interval ~Brence2

RRFR is able to maintain three of the fourteen outliers within a 95% CI of it prediction (Table 9). Outlier observations 69, 151, and 195 fall just within the 95% CI. RRFR does not predict the extreme outlier 87 as well as RFR. In this case, RFR is 4.541 closer to maintaining this outlier within its 95% CI. RRFR performs overall better with a total distance value of 61.715 compared to RFR's 67.859. Another important result was that both RFR and RRFR had the same outlier prediction performance for the Brence2 test dataset, even with different Root_MSE values.

RRFR Outlier Predictions based on 95% Confidence Interval ~ Brence2					
Training Model MSE		41.024	Root_MSE	6.405	
Obs #	**Y pred**	**Y obs**	**LO_95% CI**	**HI_95% CI**	**Distance**
10	48.879	69.590	29.664	68.094	1.496
13	52.325	74.170	33.110	71.540	2.630
69	53.733	72.340	34.518	72.948	-0.608
87	28.698	-22.300	9.483	47.913	31.783
90	52.522	30.680	33.307	71.737	2.627
108	49.947	71.280	30.732	69.162	2.118
124	49.737	29.610	30.522	68.952	0.912
151	55.169	73.660	35.954	74.384	-0.724
182	48.359	69.830	29.144	67.574	2.256
189	46.128	24.460	26.913	65.343	2.453
195	50.315	31.210	31.100	69.530	-0.110
199	49.685	28.360	30.470	68.900	2.110
Test Model MSE		20.000	Root_MSE	4.472	
19	49.362	29.060	35.946	62.778	6.621
20	50.577	29.640	37.161	63.993	8.151
Total Distance outside of 95% CI					61.715

Table 9: RRFR Outlier Predictions based on 95% Confidence Interval ~Brence2

The relationship between the predictability of RFR and RRFR may be explained using the Statistics of Extremes or Extreme Value Theory [8,9]. RFR does a better job in the conventional manner of modeling; prediction of the bulk of the data distribution in order to create a general model with minimal error. RRFR, while not intentional in this study, does a better job in modeling the extreme values or tails of the distribution while trading off prediction error for the bulk of the distribution. Table 6 through Table 9, explained earlier; provide an example of this result.

The Statistics of Extremes focus on the prediction of the tails of distributions, often called outliers by modelers. The focus is important because many events such as rainfall, floods, air pollution, corrosion, the stock market, airplane and train crashes, as well as other freak weather patterns are seen as extreme events [9]. If these events could be predicted, it could save lives and/or money.

4.4 Will oversampling the bootstrap improve predictions?

In this section, we introduce and provide results of a new sampling method for the random forests algorithm. This method, in concert with the bootstrapped sample, is used to improve the predictability of the algorithm on three fronts: 1) It provides more observations, sampled from the training dataset, to create a better model, 2) it attempts to quell the effect of outliers by creating the possibility of a cluster of like observations rather than a single outlier, and 3) it increases the number of possible nodes in a tree by adding more observations to N_{Tng} in the relationship below.

$$M = 2*(N_{Tng} / N_{m_{threshold}}) + 1$$

where M is the maximum possible nodes in a tree, N_{Tng} is the number of observations used from the training dataset used to create the model, and $N_{m_{threshold}}$ is the threshold for the minimum number of observations in a parent node. The scalar "2" accounts for the binary splits of parent nodes and the (+1) represents the root node or the first (parent) node of the tree. In the tree creation process, if the sum of the tree nodes is equivalent to M , the algorithm will declare any childless parent nodes as terminal nodes.

The *oversampling* idea is closely related to bootstrapping. Essentially this method adds more training data to the dataset just prior to bootstrapping. Figure 1 shows the pseudo-code for this algorithm in it simplest form; realizing that other coding efforts was required to maintain structural integrity throughout the algorithm.

```
For a count of N_Tng +1 to N_OVS (for the oversample)
    Choose a Random number between 1 and N_Tng
    Make the random-selected observation y equal to the oversample y
        For a count of 1 to number of predictors
        Make the random-selected observation x_i equal to oversample x_i
        End (Predictor Loop)
End (Observation Loop)
    Then Call Bootstrap algorithm where N_OVS replaces N_Tng .
```

Figure 1: Oversampling Pseudo-Code

An oversampling approach was taken to model the Ozone and Servo datasets in order to explore the effectiveness of this approach with RFR and RRFR (median). Table 10 through Table 13 display the results of specific runs based on 10% and 25% oversamples. Note that the values in the MSE_tst and MAD_tst columns are calculated differences from the original data runs. For ease of reading the table, any cell highlighted in green shows improvement. In addition, the OF column indicates where overfitting occurred. For the OF column: A = MAD overfit, S = MSE overfit, and B = Both MAD and MSE overfit.

4.4.1 Ozone Results

Table 10 shows how RRFR (median) algorithm performed with the oversampling technique. It is clear that this algorithm did not benefit with this technique when applied to the ozone dataset, especially when increasing the oversample from 10% to 25%.

Comparison of Training Dataset Oversample Results to Orginal Runs ~ Ozone								
RRFR (median prediction) Run Results								
			10% Tng Oversample			25% Tng Oversample		
$N_{mthresh}$	Z	l_m	MSE_tst	MAD_tst	OF	MSE_tst	MAD_tst	OF
5	50	0.25	8.089	0.593		45.648	2.915	
5	50	0.5	4.375	0.360		39.367	2.346	
5	50	0.75	5.702	0.458		40.207	2.431	
5	100	0.25	7.092	0.581		62.802	3.689	
5	100	0.5	4.912	0.385		29.764	2.126	
5	100	0.75	4.767	0.347		30.363	1.753	
5	200	0.25	6.287	0.474		61.775	3.777	
5	200	0.5	6.956	0.546		46.587	2.750	
5	200	0.75	6.147	0.473		35.779	2.208	
10	50	0.25	6.301	0.470		36.148	2.326	
10	50	0.5	6.209	0.395		27.698	1.390	
10	50	0.75	4.449	0.381		28.990	1.693	
10	50	0.75	4.540	0.314		8.337	0.397	
10	100	0.25	4.964	0.363		26.435	1.810	
10	100	0.5	5.808	0.421		26.215	1.487	
10	100	0.75	5.287	0.394		28.378	1.763	
10	200	0.25	5.138	0.364		39.922	2.498	
10	200	0.5	5.416	0.388		22.028	1.430	
10	200	0.75	5.533	0.394		17.703	0.994	
20	50	0.25	4.041	0.283		13.870	0.886	
20	50	0.5	3.620	0.202		13.789	0.755	
20	50	0.75	4.326	0.313		8.319	0.368	
20	100	0.25	4.140	0.273		17.326	1.119	
20	100	0.5	4.834	0.325		8.989	0.398	
20	100	0.75	4.796	0.355		10.801	0.592	
20	200	0.25	4.124	0.278		15.438	0.910	
20	200	0.75	4.664	0.334		15.294	0.746	

Table 10: RRFR (Median) Training Dataset Oversample (10% vs. 25%) ~ Ozone

The RFR algorithm, ozone results shown in Table 11, has a limited increased performance when the oversampling technique is utilized. There is no apparent general statement of improved performance based on a 10% or 25% oversample. In some cases the 10% outperforms the 25% oversample and vice versa. One possible conclusion is that as you increase $N_{m_{threshold}}$ you are more probable in better performance; however, this is not a validated conclusion.

Comparison of Training Dataset Oversample Results to Orginal Runs ~ Ozone								
RFR (mean prediction) Run Results								
			10% Tng Oversample			25% Tng Oversample		
$N_{mthresh}$	Z	l_m	MSE_tst	MAD_tst	OF	MSE_tst	MAD_tst	OF
5	50	0.25	-0.220	0.003		-0.187	0.034	
5	50	0.5	0.502	0.775		1.251	-0.111	
5	50	0.75	-0.179	-0.010		0.382	0.058	
5	100	0.25	0.538	0.074		1.307	0.249	
5	100	0.5	0.142	0.010		0.108	0.036	
5	100	0.75	-0.384	-0.012		-0.039	0.021	
5	200	0.25	0.252	0.003		0.334	0.053	
5	200	0.5	0.084	0.005		0.011	0.022	
5	200	0.75	-0.084	0.022		0.234	0.045	
10	50	0.25	0.346	-0.018		0.000	-0.015	
10	50	0.5	0.528	0.077		-0.016	-0.013	
10	50	0.75	0.455	0.069		0.935	0.161	
10	100	0.25	-0.369	-0.031		0.069	0.067	
10	100	0.5	-0.244	-0.016		-0.349	-0.014	
10	100	0.75	-0.056	-0.019		0.096	-0.005	
10	200	0.25	0.029	0.033		1.414	0.257	
10	200	0.5	0.003	0.001		0.257	0.070	
10	200	0.75	-0.046	-0.014		-0.013	0.007	
20	50	0.25	0.068	0.058		0.072	0.041	
20	50	0.5	0.388	0.034		0.131	-0.055	
20	50	0.75	-0.152	-0.054		-0.263	-0.055	
20	100	0.25	-0.272	-0.022		-0.118	0.033	
20	100	0.5	-0.174	-0.029		0.112	-0.011	
20	100	0.75	-0.204	-0.041		-0.219	-0.044	
20	200	0.25	0.010	0.007		0.006	-0.002	
20	200	0.5	-0.151	-0.056		0.018	-0.033	
20	200	0.75	0.200	-0.001		0.314	-0.010	

Table 11: RFR (Mean) Training Dataset Oversample (10% vs. 25%) ~ Ozone

Both RRFR and RFR did not overfit any model. No overfitting transpired because this dataset is a good general example of the test dataset. This occurrence is not always the case, nor is overfitting a negative result in some instances, especially in the case of the Servo dataset in the next section.

4.4.2 Servo

Table 12 exhibits the results of oversample the servo dataset with the RRFR (median) algorithm. This result is much different than reported earlier in Table 10. We found that it is beneficial to oversample in all cases at the 10% rate. For the 25% oversample, we improve our MSE_tst every time, while the MAD_tst showed improvement only 60% of the time. In addition, there is no significant benefit of increasing the oversample from 10% to 25% until the latter half of the table; otherwise a 10% oversample does just fine.

The benefit in the latter half the table may be a function of the increased number of nodes allowed in each tree. This outcome is a benefit because once the algorithm reaches the allowable number of nodes in the tree; it labels the remaining childless parent nodes as terminal. This is where a significant amount of error may be encountered due to the order of parent nodes are examined for splitting. This is one of the main reasons why oversampling was explored.

This dataset and algorithm combination provides the first instance of overfitting found in this section. Overfitting, often thought of a negative by product of modeling, in this case may assist to create better models. Of the six events, we still see improvement in one form or another. In four of the instances, we see a decreased performance in the MAD_tst, for which that run was overfit. Within these results, the comparative result of overfitting positively impacts our modeling effort.

Comparison of Training Dataset Oversample Results to Orginal Runs ~ Servo							
RRFR (median prediction) Run Results							
					25% Tng Oversample		
$N_{mthresh}$	Z	I_m	MAD_tst	OF	MSE_tst	MAD_tst	OF
5	50	0.25	-0.235		-0.680	0.250	A
5	50	0.5	-0.145		-0.829	0.120	
5	50	0.75	-0.102		-1.259	0.004	
5	100	0.25	-0.224		-1.024	0.147	
5	100	0.5	-0.147		-1.151	0.092	
5	100	0.75	-0.122		-1.873	-0.109	
5	200	0.25	-0.240		-1.079	0.155	A
5	200	0.5	-0.159		-1.573	-0.010	A
5	200	0.75	-0.138		-1.494	0.007	
10	50	0.25	-0.146		-0.892	0.100	
10	50	0.5	-0.081		-1.428	-0.038	
10	50	0.75	-0.076		-1.659	-0.115	
10	100	0.25	-0.169		-1.191	0.073	A
10	100	0.5	-0.079		-1.138	0.057	
10	100	0.75	-0.057		-1.879	-0.149	
10	200	0.25	-0.178		-0.841	0.137	A
10	200	0.5	-0.102		-1.629	-0.104	
10	200	0.75	-0.057		-2.120	-0.225	
20	50	0.25	-0.051		-1.134	-0.017	
20	50	0.5	-0.039		-1.322	-0.076	
20	50	0.75	-0.012	A	-2.113	-0.316	
20	100	0.25	-0.070		-1.411	-0.067	
20	100	0.5	-0.032		-2.091	-0.314	
20	100	0.75	-0.026		-2.006	-0.299	
20	200	0.25	-0.065		-1.393	-0.089	
20	200	0.5	-0.038		-1.820	-0.232	
20	200	0.75	-0.029		-2.250	-0.370	

Table 12: RRFR (Median) Training Dataset Oversample (10% vs. 25%) ~ Servo

Oversampling RFR at the 10% and 25 % level for the servo dataset reveals an interesting result (Table 13). Generally we find better performance with the larger oversample with some exceptions. Most of these exceptions occur in the upper third of the table where oversample does not positively impact the modeling effort. However, in the bottom two-thirds, we show that the larger the oversample we take, the better result in both the MSE_tst and MAD_tst criteria. A possible explanation is that for the upper third of the table the $N_{m_{threshold}} = 5$, where the number of nodes in the dataset is generally sufficient and oversampling provides no lift to the predictive capability. However, when we only use 25% of the variables to split ($I_m = 0.25I$), there is a benefit to oversampling, more likely due to the enhanced opportunity to evaluate the predictor variables provided by the increased number of nodes.

The lower two-third's performance may described as the "more data = more tree nodes" effect. With a larger $N_{m_{threshold}}$ variable, the algorithm often gets into the predicament where many of the parent nodes are not considered for a split which most likely increases model error. Therefore, oversampling provides a positive result.

Comparison of Training Dataset Oversample Results to Orginal Runs ~ Servo							
RFR (mean prediction) Run Results							
					25% Tng Oversample		
$N_{mthresh}$	Z	I_m	MAD_tst	OF	MSE_tst	MAD_tst	OF
5	50	0.25	-0.036	B	-0.198	-0.072	S
5	50	0.5	0.023	B	0.070	0.029	B
5	50	0.75	0.037	B	0.121	0.030	A
5	100	0.25	-0.050	B	-0.135	-0.060	S
5	100	0.5	0.009	B	0.017	0.008	B
5	100	0.75	0.039	B	0.071	0.013	A
5	200	0.25	-0.016	B	-0.078	-0.029	B
5	200	0.5	0.026	B	0.067	0.010	
5	200	0.75	0.043	B	0.087	0.029	
10	50	0.25	-0.014	B	-0.194	-0.102	B
10	50	0.5	-0.016	B	-0.086	-0.044	
10	50	0.75	0.010	B	-0.004	-0.013	A
10	100	0.25	-0.019	B	-0.152	-0.068	S
10	100	0.5	-0.014	B	-0.050	-0.034	
10	100	0.75	0.017	B	-0.011	-0.006	S
10	200	0.25	-0.040	B	-0.153	-0.082	S
10	200	0.5	-0.012	B	-0.060	-0.035	B
10	200	0.75	0.016	B	0.007	-0.009	B
20	50	0.25	-0.102	B	-0.351	-0.129	S
20	50	0.5	-0.034	B	-0.248	-0.111	B
20	50	0.75	-0.018	B	-0.175	-0.061	S
20	100	0.25	-0.027	B	-0.271	-0.108	B
20	100	0.5	-0.049	B	-0.185	-0.100	B
20	100	0.75	-0.008	B	-0.165	-0.045	B
20	200	0.25	-0.054	B	-0.289	-0.103	B
20	200	0.5	-0.045	B	-0.173	-0.101	B
20	200	0.75	-0.018	B	-0.171	-0.055	B

Table 13: RFR (Mean) Training Dataset Oversample (10% vs. 25%) ~ Servo

Another result displayed in Table 13 is the increased event of overfitting. In the 10% oversample, we see that both MSE and MAD overfit every experimental run, with a predominant improvement to the model. In the 25% oversample, we run the gamut of overfitting. In this table we see that every time we overfit with MAD only, the model does not improve, while when we overfit with MSE only we get better results. When both overfit, more often than not we show improvement. From this overfitting result; however, no general modeling statements are made. This result seems to be inherent to this specific dataset.

4.4.3 What happens to the error if we allow single observation terminal nodes?

We chose to use the Boston Housing and Brence2 datasets to examine the results of setting $N_{m_{threshold}} = 2$ or allowing a single observation to inhabit a terminal node. The Boston dataset was used because it is a pretty generic dataset and the Brence2 was chosen because of the interesting result where RFR performed best with the MSE test criterion while RRFR performed best with the MAD test criterion. For the RRFR runs, we used the median prediction method for two reasons, 1) it did well with these two datasets and 2) some of the other robust prediction methods have trouble with calculations with less than five observations in a node.

There were three possible conclusions that we figured into this analysis. The first was unlikely, but possible that we could maintain the same models and error status quo. Second, and more feasible, is that the training model would overfit and perform much worse on the test dataset then the original runs. A third possible result, from allowing single observations to populate a terminal node, would be increased efficiency with respect to outliers and abnormal observations, resulting in an improvement of the test dataset error.

We chose to compare the new runs to the original run combinations that used an $N_{m_{threshold}} = 5$. This comparison seemed reasonable since it was the closest relative run. Using an $N_{m_{threshold}} = 10$ or 20 did not seem like a fair comparison. Table 14 and Table 15 show a comparison of the error results for change of $N_{m_{threshold}}$. The green shaded cells show where there was improvement when the runs incorporated $N_{m_{threshold}} = 2$, simply calculated by subtracting the $N_{m_{threshold}} = 5$ results from these new runs.

Table 14 shows the results for the modeling the Boston Housing dataset. From this table, we see mixed results from this exploration. Using the median prediction method or RRFR, the results show that improvement occurs about 50% of the time in both the training and test set errors, with no distinguishable pattern-explanation. It is quite clear, however, that the mean prediction or RFR does not benefit from the further observation breakdown beyond $N_{m_{threshold}} = 5$ for the Boston Housing dataset. Finally, it was remarkable that the runs with $N_{m_{threshold}} = 2$ never caused a model to overfit the dataset for both RRFR and RFR.

Comparison of Terminal Node Threshold: 5 vs. 2 ~ Boston Housing					
RRFR (median prediction) Run Results					
Z	**l_m**	**MSE_tng**	**MAD _tng**	**MSE_tst**	**MAD_tst**
50	0.25	7.441	0.152	0.690	0.075
50	0.5	199.911	5.362	-1.014	-0.041
50	0.75	14.172	0.293	-0.181	-0.010
100	0.25	-21.301	-0.449	-0.764	-0.050
100	0.5	22.347	0.463	-0.145	-0.024
100	0.75	-5.328	-0.111	0.202	0.012
200	0.25	-16.778	-0.358	0.051	-0.002
200	0.5	-211.450	-5.620	0.067	0.030
200	0.75	3.235	0.066	0.442	0.049
RFR (mean prediction) Run Results					
Z	**l_m**	**MSE_tng**	**MAD _tng**	**MSE_tst**	**MAD_tst**
50	0.25	1.549	0.156	0.393	0.069
50	0.5	0.887	0.107	0.174	0.038
50	0.75	1.185	0.141	0.340	0.091
100	0.25	1.415	0.143	0.604	0.115
100	0.5	0.873	0.096	0.469	0.102
100	0.75	0.742	0.101	0.485	0.131
200	0.25	0.720	0.086	0.535	0.095
200	0.5	0.971	0.104	0.394	0.089
200	0.75	0.761	0.088	0.368	0.087

Table 14: Error Performance Comparison: *Nthsize* 2 vs. 5 ~ Boston

The experimental result from the Brence2 dataset shows that reducing $N_{m_{threshold}}$ to 2 enhances the overall test dataset error performance of RFR and RRFR. Only in one instance, did RRFR not improve its test dataset performance for MSE; however, in the same run it did improve the MAD test error. RRFR did improve its training error performance 66% of the time, which is not as important as improving the test error performance. RFR, with its mean prediction, showed a marked improvement across the board with the test error, with a resounding decrease in performance of the training error. In all cases presented for the Brence2 dataset, the models never overfit the dataset.

Comparison of Terminal Node Threshold: 5 vs. 2 ~ Brence2					
RRFR (median prediction) Run Results					
Z	**l_m**	**MSE_tng**	**MAD _tng**	**MSE_tst**	**MAD_tst**
50	0.25	-29.400	-0.295	-0.160	-0.016
50	0.5	-116.840	-1.204	-0.513	-0.117
50	0.75	-4.880	-0.051	0.085	-0.008
100	0.25	250.600	2.611	-0.643	-0.068
100	0.5	810.610	11.580	-0.425	-0.056
100	0.75	-1051.310	-14.066	-0.559	-0.074
200	0.25	-39.610	-0.410	-0.034	-0.013
200	0.5	205.590	2.253	-0.539	-0.087
200	0.75	-100.940	-1.021	-0.522	-0.054
RFR (mean prediction) Run Results					
Z	**l_m**	**MSE_tng**	**MAD _tng**	**MSE_tst**	**MAD_tst**
50	0.25	2.213	0.100	-0.242	-0.029
50	0.5	0.064	0.034	-0.445	-0.004
50	0.75	1.920	0.084	-0.630	-0.045
100	0.25	1.004	0.060	-0.089	-0.068
100	0.5	0.569	0.025	-0.317	-0.070
100	0.75	0.884	0.066	-0.225	-0.026
200	0.25	1.756	0.052	-0.073	-0.075
200	0.5	1.073	0.018	-0.403	-0.055

Table 15: Error Performance Comparison: *Nthsize* 2 vs. 5 ~ Brence2

Overall, it is worth the time and minimal effort to conduct runs with the $N_{m_{threshold}} = 2$, since even when an algorithm decreased in test error performance, the change was not very significant. Timing may be a problem because the lower the value of $N_{m_{threshold}}$, the longer the CPU runtime because the number of nodes in a tree is calculated as a function of the terminal threshold and training sample size. Running the code in this fashion may be deceptive if a test dataset is not used. In many of the cases, the training error increased, but improved the test error; a result not found without a test dataset.

The results reported in this section imply this experiment is more worthwhile when modeling a noisy dataset (Brence2). In this manner, running the additional code is well worth the time and effort where the results seem unanimous in improving the models. The reduced $N_{m_{threshold}}$ seems to segregate many of the abnormalities or outliers in the dataset and provide and improvement of the test error regardless of the result of the training error.

5 CONCLUSION

Based on the results of our main experiment, we found that RFR's predictive performance dominates RRFR in terms of minimizing test dataset error. This outcome seems counter-intuitive based on the results in [4] and the number of datasets that included outliers. We suppose that this performance conclusion is related to whether the outlier is unbounded or not. There are other indications of why RFR performed better; however, these clues require more analysis and future work.

We explored the reason behind why RRFR did not perform as well with the datasets used in this study. The datasets selected were generally biased towards using robust measures which further intrigued this analyst. Our hypothesis, explained at a very high level with simulation validation is that RRFR did a better job of modeling the outliers while accepting increased error on the predicting the bulk of the data. This result relates to the study of statistics of extremes or extreme value theory which focuses on extreme values often found in catastrophic accidents. In relation to corrosion modeling, the RRFR result is important to this study because the existence or sizable depth of corrosion is typically an extreme event. In addition the combined effort of the

RFR and RRFR models may provide excellent guidance to the technician responsible for detecting and repairing corrosive materials

We found many interesting results from our additional analysis that may provide a better understanding of the properties of these random forest regression algorithms. Through our additional analysis, we provided a means to seek model improvement and a discussion of what methods uncover additional data information.

Through our continued study of RRFR and RFR we have found specific instances where one method performed better than the other. In future work, we will show why the mean prediction method (RFR) dominated our experiment, why RRFR did not perform as well with these datasets, as well as a proposed solution to improve the prediction of RRFR when dealing with the datasets used in this study.

6 REFERENCES

1. Breiman, L. [2001] *Random Forests*, Statistics Department, University of California, ftp://ftp.stat.berkeley.edu/pub/users/breiman/.
2. Breiman, L. [2000] *Understanding Complex Predictors*, Statistics Department, University of California. ftp://ftp.stat.berkeley.edu/pub/users/breiman/.
3. Brence, John R. & Brown, Donald E. [2002] Data Mining Corrosion from Eddy Current Non-Destructive Tests, *Computers and Industrial Engineering: Data Mining and Knowledge Discovery in Industrial Engineering*. 43, pp. 821- 840.
4. Brence, John R. & Brown, Donald E. [2006] Analysis of Robust Measures in Random Forest Regression, Technical Report sie06_0002, Department of Systems and Information Engineering, University of Virginia.
5. Christensen, Ronald [1989] Data Distributions: A Statistical Handbook. 2 ed., Entropy Limited, Lincoln, MA.
6. Friedman, J. [1991] Multivariate Adaptive Regression Splines, *The Annals of Statistics*, 19, 1, pp. 1-67.
7. Forsyth, D.S. [2000] *Nondestructive Inspections of Calibration Specimens and KC 135 Aircraft Specimens*. Institute for Aerospace Research, Canada.
8. Gumbel, E.J. [1958] Statistics of Extremes, Columbia University Press, New York, NY.
9. Kotz, S, & Nadarajah, S. [2001] Extreme Value Distributions: Theory and Applications, Imperial College Press, World Scientific Publishing Company, Portland, Oregon.
10. NIST, *Engineering Statistics Handbook*. http://www.itl.nist.gov .