8-2017

Tree-based Regression for Interval-valued Data

Chih-Ching Yeh
Utah State University

Tree-based Regression for Interval-valued Data

A Project Presented to the
Department of Mathematics and Statistics

Utah State University

In Partial Fulfillment
of the Requirements for the Degree

Master of Statistics

Utah State University

by

Chih-Ching Yeh

June 2017

Approved:

Yan Sun Major Professor	Adele Cutler Committee Member
John Stevens Committee Member	

SUMMARY

Regression methods for interval-valued data have been increasingly studied in recent years. As most of the existing works focus on linear models, it is important to note that many problems in practice are nonlinear in nature and therefore development of nonlinear regression tools for interval-valued data is crucial. In this project, we propose a tree-based regression method for interval-valued data, which is well applicable to both linear and nonlinear problems. Unlike linear regression models that usually require additional constraints to ensure positivity of the predicted interval length, the proposed method estimates the regression function in a nonparametric way, so the predicted length is naturally positive without any constraints. A simulation study is conducted that compares our method to popular existing regression models for interval-valued data under both linear and nonlinear settings. Furthermore, a real data example is presented where we apply our method to analyze price range data of the Dow Jones Industrial Average index and its component stocks.

Keywords and phrases: random forests; nonparametric; regression tree; kernel regression; nonlinearity; prediction accuracy

AMS Classification: 62G08; 62J05

JEL Classification:

1. Introduction

Regression with interval-valued data has been attracting increasing interest among researchers. There are various models built upon set arithmetic, which typically view interval-valued data as realizations of one-dimensional random sets (Diamond (1990), Korner and Näther (1998), Gil et al. (2002), Gil et al. (2007), González-Rodríguez et al. (2007), Blanco et al. (2011), Cattaneo and Wiencierz (2012)). Separately, there are also plentiful models developed under symbolic data analysis (SDA) that treat an interval as a bivariate vector (Carvalho et al. (2004), Billard (2007), Lima Neto and Carvalho (2008), Lima Neto and Carvalho (2010)). A detailed review of these models will be given in Section 2. Generally speaking, models from both domains have their distinct advantages and disadvantages, but they all share the common drawback of being too restrictive. In other words, all of these models are rigid to some degree due to the constraints they need to impose on their parameters. This is rooted in the fundamental fact that the space of intervals $\mathcal{K}_{\mathcal{C}}(\mathbb{R})$ as a metric space is not linear (see Appendix A). In addition, many data in practice are nonlinear, for which linear models are insufficient. Motivated by all of these, we propose to study regression of interval-valued data by nonparametric approaches. Without any distribution assumption, nonparametric methods usually do not require additional constraints and therefore are more flexible. Furthermore,

they can take care of both linearity and nonlinearity. Particularly in this project, we propose a tree-based regression method based on random forests.

Random forests (Breiman (2001)) are ensembles of classification or regression trees created using bootstrap samples of the training data and random feature selection in tree induction. The method is among the most popular algorithms and widely used in data science and machine learning in appreciation of its high prediction performance. Unlike other ensemble methods, random forests involve additional randomness by selecting a random subset of the predictor variables at each splitting node, which makes it robust to overfitting and high-dimensionality. For classification, the prediction is made by the majority vote from all the trees. For regression, the prediction is instead determined by the average of the tree predictions. For interval-valued data, comparing to other regression methods, random forests automatically overcome the difficulty of mathematical coherence, i.e., the predicted interval range/radius must be nonnegative, due to their nonparametric nature. Thus, it is expected to perform better, especially when data do not fulfill the restrictions of other regression methods. Furthermore, the fact that random forests can deal with both linearity and nonlinearity makes them more flexible methods.

The rest of the project is organized as follows. Section 2 provides a review of major (linear and nonlinear) regression methods for interval-valued data in the literature, where in particular an extension of the kernel method for nonlinear interval regression is discussed. Section 3 describes random forests as classical methods of classification and regression in machine learning, and proposes their interval-valued adaptation. Results of a systematic simulation study that compares random forest regression to typical existing methods are reported in Section 4, and real data applications are presented in Section 5. Section 6 concludes with remarks for future research. Preliminaries of random sets theory are deferred to the Appendix.

Throughout the project, we denote by $[x] \in \mathcal{K}_{\mathcal{C}}(\mathbb{R})$ a bounded closed interval, whose lower and upper bounds are denoted by x^L and x^U, respectively. Alternatively, $[x]$ can also be represented by its center and radius, denoted by x^C and x^R, respectively. A random interval which takes values in $\mathcal{K}_{\mathcal{C}}(\mathbb{R})$ is denoted by $[X]$. Bolded letters denote vectors. For example, $[\boldsymbol{x}] = [[x_1], \cdots, [x_p]]^T$ denotes a p-dimensional hyper interval, and its random version is denoted by $[\boldsymbol{X}]$. There are various metrics defined for the space $\mathcal{K}_{\mathcal{C}}(\mathbb{R})$ (see Appendix A), which usually can be used for the same model with the choice up to the user. We adopt a unifying notation $d(\cdot, \cdot)$ for all of the metrics.

2. Review of Regression Methods

2.1. Linear regression

Linear regression for interval-valued data has been extensively studied in the past decades. Existing models have been developed mainly in the two domains of random

sets and symbolic data analysis (SDA). In the framework of random sets, an interval is viewed as a compact convex set in the one-dimensional real space $\mathbb{R}$, and the linear relationships between intervals are modeled according to set arithmetic. Separately, the aim of SDA is to extend classical data analysis techniques to nontraditional data formats, such as lists, intervals, histograms and distributions. In this framework, linear regression is often developed by fitting separate point-valued models to the center and radius (or the lower and upper bounds), essentially treating an interval as a bivariate vector. In the following, we briefly review major models from both domains.

The foundation of linear regression in the random sets framework is the linear structure in the space $\mathcal{K}_{\mathcal{C}}(\mathbb{R}^d)$ defined by the Minkowski addition and scalar multiplication, i.e.

$$\begin{aligned} &A + B = \{a + b : a \in A, b \in B\}, \\ &\lambda A = \{\lambda a : a \in A\}, \\ &A, B \in \mathcal{K}_{\mathcal{C}}, \quad \lambda \in \mathbb{R}. \end{aligned}$$

For $[x], [y] \in \mathcal{K}_{\mathcal{C}}(\mathbb{R})$, this means

$$[x] + [y] = [x^L + y^L, x^U + y^U], \tag{2.1}$$

$$\lambda[x] = \begin{cases} [\lambda x^L, \lambda x^U], & \lambda \geq 0, \\ [\lambda x^U, \lambda x^L], & \lambda < 0. \end{cases} \tag{2.2}$$

Although different researchers may have different notation and degrees of restrictions (Gil et al. (2001), Gil et al. (2002), Gil et al. (2007), González-Rodríguez et al. (2007), Sinova et al. (2012)), linear regression models in $\mathcal{K}_{\mathcal{C}}(\mathbb{R})$ generally have the form

$$[Y] = a[X] + [\mathcal{E}], \tag{2.3}$$

where $a \in \mathbb{R}$ and $[\mathcal{E}]$ is an interval-valued random error with a fixed expectation $E([\mathcal{E}]) = [b] \in \mathcal{K}_{\mathcal{C}}(\mathbb{R})$. Here the addition and multiplication are in the sense of (2.1) and (2.2), respectively. It follows that

$$\widehat{[Y]} = E([Y] \mid [X]) = a[X] + [b], \tag{2.4}$$

Equivalently, the equation (2.4) can be written in terms of the center and radius as

$$\hat{Y}^C = aX^C + b^C, \tag{2.5}$$

$$\hat{Y}^R = |a|X^R + b^R. \tag{2.6}$$

This leads to the following equation that shows linearity in $\mathcal{K}_{\mathcal{C}}$:

$$d\left(\widehat{[Y_1]}, \widehat{[Y_2]}\right) = |a| d([X_1], [X_2]). \tag{2.7}$$

That is, a constant change in the metric of $[X]$ results in a constant change in the metric of $[Y]$. This reveals the essence of model (2.3) as an analogous result of the fundamental property of classical linear regression. However, such a property has to be achieved at the price of reduced model flexibility from practical point of view. Notice from (2.5)-(2.6) that the slopes for the center and radius equations must have the same absolute value. This reduction of flexibility is the price we pay to achieve linearity in $\mathcal{K}_{\mathcal{C}}$.

The multivariate extension of model (2.3) is given as

$$[Y] = \sum_{i=1}^{p} a_i[X_i] + [\mathcal{E}], \tag{2.8}$$

or equivalently, in the center-radius form

$$Y^C = \sum_{i=1}^{p} a_i X_i^C + \mathcal{E}^C, \tag{2.9}$$

$$Y^R = \sum_{i=1}^{p} |a_i| X_i^R + \mathcal{E}^R, \tag{2.10}$$

where $a_i \in \mathbb{R}$, $i = 1, \cdots, p$, and $\mathcal{E}^C, \mathcal{E}^R$ are random variables with $E(\mathcal{E}^C) = b^C \in \mathbb{R}$ and $E(\mathcal{E}^R) = b^R > 0$. Estimation of the model parameters is generally performed by the least squares method that minimizes the mean squared errors with respect to the metric in $\mathcal{K}_{\mathcal{C}}(\mathbb{R})$. Assuming observing a sample of size n, the least squares estimates are given by

$$\left\{\hat{a}_1, \cdots, \hat{a}_p; \widehat{[b]}\right\} = \arg\min \left\{ \frac{1}{n} \sum_{j=1}^{n} d^2 \left([y_j], \sum_{i=1}^{p} a_i[x_{ij}] + [b] \right) \right\}, \tag{2.11}$$

subject to the constraint $b^R > 0$.

In the separate framework of SDA, interval linear regression has been studied mainly by treating the intervals as bivariate vectors. Models belonging to this category typically specify separate linear equations for the center and radius (or upper and lower bound), respectively. For example, the MinMax method by Billard and Diday (2002) has the model equations

$$Y^L = \beta_0^L + \sum_{i=1}^{p} \beta_i^L X_i^L + \epsilon_1, \quad \beta_i^L \in \mathbb{R},\ i = 0, \cdots, p,$$

$$Y^U = \beta_0^U + \sum_{i=1}^{p} \beta_i^U X_i^U + \epsilon_2, \quad \beta_i^U \in \mathbb{R},\ i = 0, \cdots, p.$$

Alternatively, Lima Neto and Carvalho (2008) proposed the Center and Range Method (CRM) as

$$\begin{aligned} Y^C &= \beta_0^C + \sum_{i=1}^{p} \beta_i^C X_i^C + \epsilon_1, \quad \beta_i^C \in \mathbb{R},\ i = 0, \cdots, p, \\ Y^R &= \beta_0^R + \sum_{i=1}^{p} \beta_i^R X_i^R + \epsilon_2, \quad \beta_i^R \in \mathbb{R}.\ i = 0, \cdots, p. \end{aligned}$$

Here, both ϵ_1 and ϵ_2 are zero-mean random errors. There is no interval structure between them. Obviously, without any constraint, these two methods suffer from the problem of mathematical coherence, i.e., the predicted upper bound may be smaller than the lower bound, or the predicted radius may be negative. This leads to the proposal of the Constrained Center and Range Method (CCRM) (Lima Neto and Carvalho (2010)), which takes the same form of CRM, with the additional constraints $\beta_0^R, \beta_1^R \geq 0$ to ensure the nonnegativity of $\hat{Y}^R$. There have been further developments based on this type of modeling. For examples, Domingues et al. (2010) considered a parametric inferential method in replacement of the least squares to deal with the issue of outliers, and Lima Neto et al. (2011) proposed a generalized linear model to allow for additional model flexibility. Although the bivariate representation of an interval could result in loss of geometric information and less interpretability (e.g., the linearity property (2.7) does not hold anymore), this type of models generally has improved flexibility, and therefore are preferred in some practical situations.

There is an intrinsic difficulty for linear regression with interval-valued data, because $\mathcal{K}_{\mathcal{C}}(\mathbb{R})$ is not a linear space (see, e.g., Sun (2016) for a detailed discussion). More specifically, although the Minkowski addition and scalar multiplication define a linear structure in $\mathcal{K}_{\mathcal{C}}(\mathbb{R})$, there is no inverse element of addition. This results in the biggest problem for most models to impose non-negative constraints. Technically, such constraints usually mean that underestimation is more heavily penalized than overestimation, leading to biased estimators. They also introduce significant computational complication, making it difficult to draw inferences.

2.2. Kernel method for nonlinear regression

Although linear regression is often preferred for its simplicity, there are many situations in practice where nonlinearity does exist and linear methods alone are not sufficient to tackle those problems. Therefore, developing nonlinear regression methods for interval-valued data is important. However, compared to linear regression, there are very few papers in the literature where nonlinear regression is rigorously studied. Among those, we are particularly interested in the kernel approach (Jeon et al. (2015)). It is essentially an extension of the multivariate kernel density

estimator to interval-valued data. The main purpose of the method is to estimate the multivariate density function based on interval-valued data. Nevertheless, it can also be used to perform nonlinear regression in an indirect way. After a brief review of the kernel approach proposed in Jeon et al. (2015), we will discuss an immediate extension of this method to directly address nonlinear regression with interval-valued data.

The classical multivariate kernel density estimator has the form

$$\hat{f}(\boldsymbol{x}) = \frac{1}{n}\sum_{j=1}^{n} K_h\left(\boldsymbol{x} - \boldsymbol{x}_j\right)/h,$$

where $K_h(\cdot)$ is a multivariate kernel function, $\boldsymbol{x} = [x_1, \cdots, x_p]^T$ denotes a p-dimensional vector, and h is the bandwidth parameter. There is a range of popularly used kernel functions, including uniform, triangular, Epanechnikov, and Gaussian. The bandwidth parameter h controls the smoothness of the estimated density function: smaller values of h can reflect more local structures resulting in a wiggly function, while larger values of h make the function smoother but with possible loss of local details. In Jeon et al. (2015), the interval-valued observation is viewed as a histogram, to which a kernel distribution is fitted. Precisely, their proposed density estimator is defined as

$$\hat{f}(\boldsymbol{x}) = \frac{1}{n}\sum_{j=1}^{n} \phi\left(\boldsymbol{x}|\hat{\boldsymbol{\mu}}_j, \hat{\boldsymbol{\Sigma}}_j\right),\ \boldsymbol{x} \in \mathbb{R}^p, \tag{2.12}$$

where $\phi(\cdot|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the multivariate Gaussian kernel (i.e., multivariate normal density). For the j^{th} interval observation, the correponding mean $\hat{\boldsymbol{\mu}}_j$ and covariance $\hat{\boldsymbol{\Sigma}}_j$ are estimated from the data with a unique set of weights. In the regression setting with outcome variable Y and predictor variables X_i, $i = 1, \cdots, p$ (for which interval-valued data is observed), one computes the conditional density of Y given $\boldsymbol{X} = [X_1, \cdots, X_p]^T$ as

$$\hat{f}_{Y|\boldsymbol{X}}(y|\boldsymbol{x}) = \frac{\hat{f}_{Y,\boldsymbol{X}}(y, \boldsymbol{x})}{\hat{f}_{\boldsymbol{X}}(\boldsymbol{x})},\ \boldsymbol{x} \in \mathbb{R}^p,\ y \in \mathbb{R}. \tag{2.13}$$

The drawback of this method is that, although it makes good use of the interval-valued data, it does not develop an interval-valued model. Namely, the densities (2.12) and (2.13) estimated from the interval-valued data are still in the point-valued context. Consequently, prediction for the interval-valued output is somewhat less straightforward. For example, Jeon et al. (2015) proposed a rather complicated prediction method, of which the key is to calculate the conditional cumulative distribution function

$$P(Y \leq y|\boldsymbol{x}^L \leq \boldsymbol{X} \leq \boldsymbol{x}^U)$$

based on the estimated conditional density $\hat{f}$. Then, the center is predicted as the conditional expectation, and the lower and upper bounds are predicted by the $\hat{q}_L^{th}$ and $\hat{q}_U^{th}$ quantiles of the conditional distribution, respectively. The values of $\hat{q}_L$ and $\hat{q}_U$ are estimated by the relative positions of $\boldsymbol{x}^L$ and $\boldsymbol{x}^U$ in the sample.

In fact, in a regression context, it is more natural to directly estimate the conditional mean as a function of the input variables, without the need of the conditional density. For example, we may consider using the kernel approach to estimate the following two regression function for the center and radius, respectively,

$$m^C\left(x_i^C, x_i^R\right) = E\left(Y^C | X_i^C = x_i^C, X_i^R = x_i^R\right), \tag{2.14}$$

$$m^R\left(x_i^C, x_i^R\right) = E\left(Y^R | X_i^C = x_i^C, X_i^R = x_i^R\right). \tag{2.15}$$

It is in a similar spirit to CCRM, except that we add the radii and centers of the predictor intervals in both the center and radius equations, to make the model more flexible. The kernel estimators for the regression functions at $[\boldsymbol{X}] = [\boldsymbol{x}^*]$ are

$$\hat{m}^C\left([\boldsymbol{x}^*]\right) = \frac{\sum_{j=1}^n K\left(\frac{d(\boldsymbol{x}^*, \boldsymbol{x}_j)}{h}\right) Y_j^C}{\sum_{j=1}^n K\left(\frac{d(\boldsymbol{x}^*, \boldsymbol{x}_j)}{h}\right)}, \tag{2.16}$$

$$\hat{m}^R\left([\boldsymbol{x}^*]\right) = \frac{\sum_{j=1}^n K\left(\frac{d(\boldsymbol{x}^*, \boldsymbol{x}_j)}{h}\right) Y_j^R}{\sum_{j=1}^n K\left(\frac{d(\boldsymbol{x}^*, \boldsymbol{x}_j)}{h}\right)}, \tag{2.17}$$

where $d(\boldsymbol{x}^*, \boldsymbol{x}_j)$ is the generalized distance between two p-dimensional hyper intervals defined by

$$d^2(\boldsymbol{x}^*, \boldsymbol{x}_j) = \sum_{i=1}^p \left\{ \left[(x_i^*)^C - x_{i,j}^C\right]^2 + \left[(x_i^*)^R - x_{i,j}^R\right]^2 \right\}.$$

As for the kernel density estimator, here $K(\cdot)$ is a kernel function and h is the bandwidth parameter. As we can see from (2.16)-(2.17), the kernel regression function is essentially a weighted average, so the predicted radius $\hat{m}^R(\cdot)$ is automatically positive without any constraints.

3. Random Forest Regression for Interval-valued Data

As we discussed previously, linear regression for interval-valued data suffers from the intrinsic problem that $\mathcal{K}_{\mathcal{C}}(\mathbb{R})$ is not a linear space. Consequently, linear models typically have to impose non-negativity constraints. For example, model (2.9)-(2.10) needs to have a positive expectation for $\mathcal{E}^R$. In CCRM, all of the coefficients in the radius equation $\beta_i^R, i = 0, 1, \cdots, p$, are constrained to be nonnegative. Therefore,

much of classical linear regression theory, such as least squares, does not apply. Instead, a constrained optimization algorithm is needed, making inference difficult. In the preceding section, we discussed the possibility of kernel regression for interval-valued data. Given its nonparametric nature, it has the advantage of solving the problem of mathematical coherence automatically. Furthermore, it can handle nonlinearity. However, the kernel method suffers from the curse of dimensionality, i.e., when the number of predictors gets large, the method tends to be slow and inaccurate (e.g., Linton and Nielsen (1995)). In addition, the performance of the kernel method depends heavily on the choice of the kernel function and the bandwidth parameter, so tuning is a big issue. In order to address these drawbacks of the kernel method, while still keeping its advantages, we propose a random forests regression for interval-valued data.

The random forest algorithm is an ensemble method developed by Leo Breiman (Breiman (2001)). A random forest is an ensemble of a collection of trees. Before the introduction of random forests, ensembles had already attracted lots of attention, as they were observed to achieve much more accurate predictions than individual trees. Many authors contributed various techniques for constructing ensembles. Among these, popular examples include Bagging (Breiman (1996)), Adaboost (Freund and Schapire (1996)), and Randomization (Dietterich (2000)). In Bagging, each tree is constructed from a bootstrap sample drawn with replacement from the training data. The original version of Adaboost resamples observations with weights that are successively adjusted to give higher weight to "difficult" observations. Randomization generates an ensemble by randomizing the interval decisions made by the base algorithm. After the ensembles are constructed, the majority vote of the individual classifiers is taken for classification and the average is taken for regression. It is well understood that averaging results in variance reduction, and reducing the correlation between individual classifiers further enhances the variance gains (Breiman (2001)). Motivated by this principle, random forests injects another layer of randomness by changing the structure of each tree. Instead of optimizing the response by evaluating all the predictors, as is done with single-tree methods or bagging, a subset of the predictors, drawn at random independently for each node in each tree, is employed. This strategy turns out to perform very well and is robust against overfitting.

One of the main advantages random forests have over other estimation methods is that they are fully non-parametric, including the effects of the predictors and response variables. So they well handle the issues of nonlinearity and mathematical coherence. Precisely, because the radius is predicted essentially by an average of the terminal nodes, which contain all positive elements, i.e., radii of the observed intervals, the prediction is automatically positive without any constraints. Situations where there are as many or more predictor variables than observations may also be problematic for traditional methods, but random forests tackle the issue of

dimensionality automatically, largely due the use of decision trees as the base learners (a building block for an ensemble process). Like all tree-based methods, random forests automatically fit interactions without the interacting predictors needing to be specified a priori. Finally and importantly, the random forests algorithm only has three tuning parameters: the size of the subset at each node, the number of trees in the forest, and the depth of the trees, which makes it conveniently applicable in practice. In fact, the number of trees can be chosen as large as desired without risk and for classification, the trees can be grown to the deepest possible depth. The results are not particularly sensitive even to the size of the subset at each node (Breiman (2001)), so tuning is relatively straightforward.

For random forests regression with interval-valued data, we propose the same regression equations (2.14)-(2.15) as in the kernel method. For completeness, we describe the basic random forests regression algorithm in the following. Readers are referred to Liaw and Wiener (2002)) for more details.

1. For each of the N desired regression trees, draw a bootstrap sample from the original data.
2. For each bootstrap sample, grow a tree as described in steps 3 and 4.
3. At each node, select *p/3* predictor variables and find all possible splits on these predictors. The "best split" is determined by minimizing the residual sum of squares over all splits.
4. Continue splitting until each node has no less than 5 observations. Once a node reaches this criterion, the node is said to be a terminal node.
5. To make a prediction for a new observation, drop the observation down each of the N trees and average the predictions.
6. The prediction accuracy is further enhanced by using the data not in the bootstrap as the test data. The estimate of the error rate is acquired by aggregating this out-of-bag (OOB) predictions, and is called the OOB estimate of error rate.

4. Simulation

In this section, we examine the empirical performance of random forest regression and compare it to CCRM and kernel estimator. For these purposes, we simulate four settings of linear and three settings of non-linear data. For each setting, three datasets with n = 500, 1000 and 2000 rectangles are generated. Of each dataset, 10% is used as the training set and the remaining 90% is used as the test set. For example, the dataset of size n = 500 is split into a training set and test set with n = 50 and 450, respectively. The models are fit to the training sets, and then the prediction errors are computed on the test sets. The predictive accuracy is measured by three popularly used criteria: coefficient of determination (R^2), mean squared error (MSE) and mean absolute error (MAE). The process is repeated for each

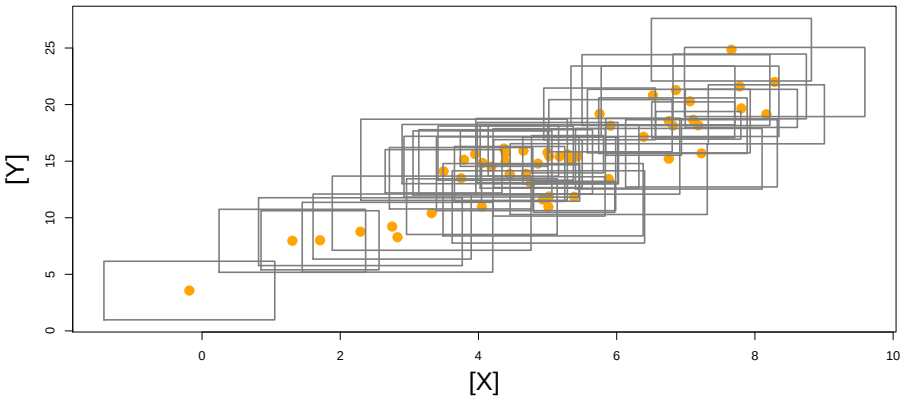

Figure 1: Plot of a simulated data set from setting 1. The gray rectangles denote the interval-valued data, and the yellow dots are the corresponding centers (same for figures 2-6 in the following).

setting/sample size combination 100 times independently, and the average results are reported in tables 1-4. All of our analysis is performed in R. The random forests algorithm is implemented using the *randomForest* package, and the kernel regression and CCRM are implemented using the packages *np* and *iRegression*, respectively. Details of our simulation settings are described below.

- Setting 1: Linear model (2.5)-(2.6) with $r > 0$. The center of the predictor interval X^C is generated from $N(5, 2^2)$, and the radius X^R is generated from $U(0.5, 1.5)$, both independently. The center and radius of the response interval are determined by the equations

$$Y_i^C = 2X_i^C + 5 + \epsilon_i^C,$$
$$Y_i^R = 2X_i^R + \epsilon_i^R,$$

  where $\epsilon_i^C \sim N(0, 2^2)$ and $\epsilon_i^R \sim N(0.5, 0.3^2)$, $i = 1, \cdots, n$. A simulated data set of $n = 200$ from setting 1 is shown in figure 1.

- Setting 2: Linear model (2.5)-(2.6) with $r < 0$. The center of the predictor interval X^C is generated from $U(0, 20)$, and the radius X^R is generated from $U(10, 11)$, both independently. The center and radius of the response interval follow the same equations as in Setting 1, but $\epsilon_i^C \sim N(0, 5^2)$ and $\epsilon_i^R \sim N(-15, 0.5^2)$, $i = 1, \cdots, n$. A simulated data set of $n = 200$ from setting 2 is shown in figure 2.

- Setting 3: Linear center and radius relationships. The center and radius of the predictor interval are independently generated from $N(5, 5^2)$ and $U(10, 15)$,

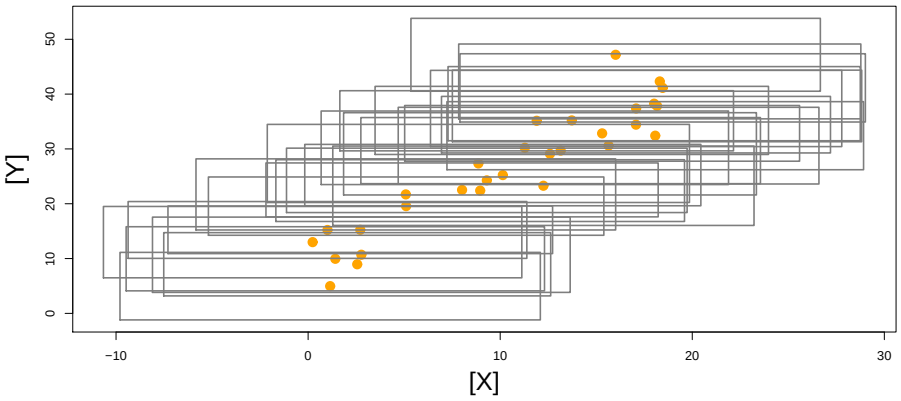

Figure 2: Plot of a simulated data set from setting 2.

respectively. The response intervals are generated according to the equations

$$Y_i^C = 10X_i^C + 20X_i^R + \eta + \epsilon_i^C,$$
$$Y_i^R = 2X_i^R + \theta + \epsilon_i^R,$$

where $\eta \sim U(0,4)$, $\epsilon_i^C \sim N(-5,\sigma^2)$, $\sigma \sim U(3,4)$ and $\frac{n}{50}\theta \sim U(0,2)$, $\epsilon_i^R \sim N(-15,1^2)$, $i = 1,\cdots,n$. A simulated data set of $n = 200$ from setting 3 is shown in figure 3.

- Setting 4: Close-to-linear center and radius relationships. The center and radius of the predictor interval are independently generated from $N(5,0.9^2)$ and $N(5,10^2)$, respectively. The response intervals are determined by

$$Y_i^C = 0.22e^{X_i^C} + \epsilon_i^C,$$
$$Y_i^R = \Phi(X_i^R;2,2) + \epsilon_i^R,$$

where $\Phi(\cdot;\mu,\sigma)$ is the CDF (cumulative distribution function) of $N(\mu,\sigma^2)$, and the random errors are generated by $\epsilon_i^C \sim N(0,\sigma_C^2)$, $\sigma_C^2 \sim U(15,20)$, $\epsilon_i^R \sim N(1,\sigma_R^2)$, $\sigma_R^2 \sim U(0,1)$, $i = 1,\cdots,n$. A simulated data set of $n = 200$ from setting 4 is shown in figure 4.

- Setting 5: Nonlinear center relationship and linear radius relationship. The center and radius of the predictor interval are independently generated from $N(5,2^2)$ and $U(0.5,1.5)$, respectively. The center and radius of the response interval follow the equations

$$Y_i^C = 6 + 4\sin\left(0.25\pi X_i^C\right) + \epsilon_i^C,$$
$$Y_i^R = X_i^R + 0.5 + \epsilon_i^R,$$

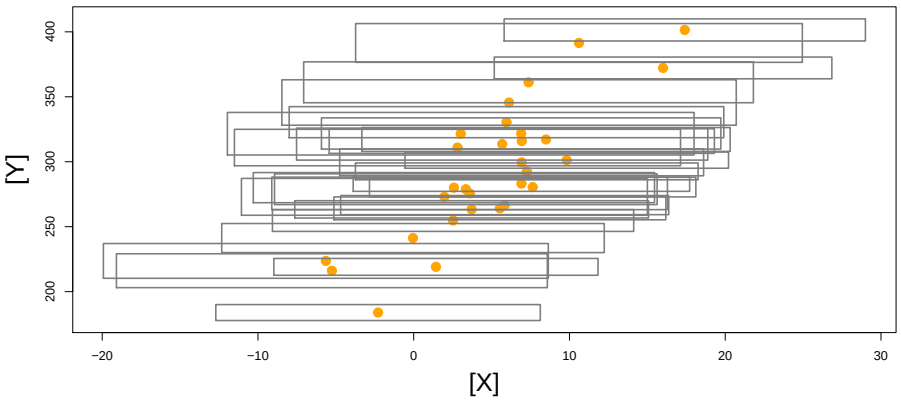

Figure 3: Plot of a simulated data set from setting 3.

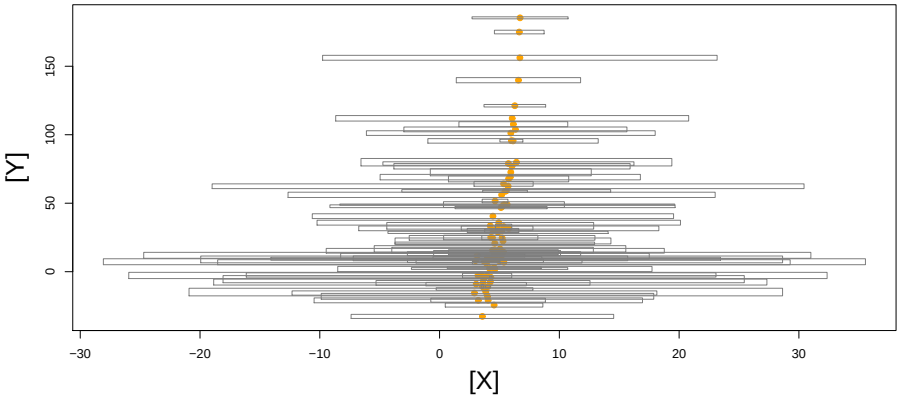

Figure 4: Plot of a simulated data set from setting 4.

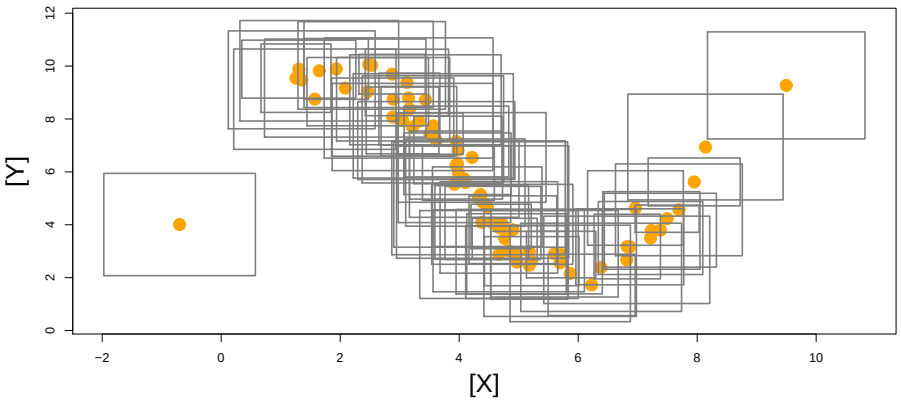

Figure 5: Plot of a simulated data set from setting 5.

where $\epsilon_i^C \sim N(0, 0.5^2)$ and $\epsilon_i^R \sim N(0, 0.2^2)$, $i = 1, \cdots, n$. A simulated data set of $n = 200$ from setting 5 is shown in figure 5.

- Setting 6: Nonlinear center and radius relationships. The center and radius of the predictor interval are independently generated from $N(5, 2^2)$ and $U(0.25, 0.5)$, respectively. The response intervals are generated by

$$\begin{aligned} Y_i^C &= 6 + 2X_i^R + \sin\left(0.253\pi X_i^C\right) + \epsilon_i^C, \\ Y_i^R &= |-0.3X_i^C X_i^R + 0.5| + \epsilon_i^R, \end{aligned}$$

where $\epsilon_i^C \sim N(0, 0.5^2)$ and $\epsilon_i^R \sim N(0, 0.1^2)$, $i = 1, \cdots, n$. A simulated data set of $n = 200$ from setting 6 is shown in figure 6.

- Setting 7: Multivariate nonlinear center and radius relationships with noise variables. There are a total of 5 predictor intervals. Their centers are generated by $X_1^C \sim N(5, 3^2)$, $X_2^C \sim \beta(0.5, 0.5)$, $X_3^C \sim N(10, 3.5^2)$, $X_4^C \sim U(0.5, 1.5)$, $X_5^C \sim N(8, 3.5^2)$, all independently. Define

$$\begin{aligned} V_1 &= u_1 + e^{-0.5\gamma(3,2)+\tau_1}, \\ V_2 &= u_2 + e^{-0.5\beta(1,3)+\tau_2}, \end{aligned}$$

where $\tau_1, \tau_2 \sim N(0, 0.2^2)$ and $u_1, u_2 \sim U(0, 0.5)$, all independently. The radii of the predictor intervals are generated by $X_1^R = \frac{2V_1}{1+V_1}$, $X_2^R = \frac{3V_2}{1+V_2}$, $X_3^R \sim N(10, 3^2)$, $X_4^R \sim U(2.5, 3.5)$, $X_5^R \sim \beta(2, 5)$, all independently. The center and

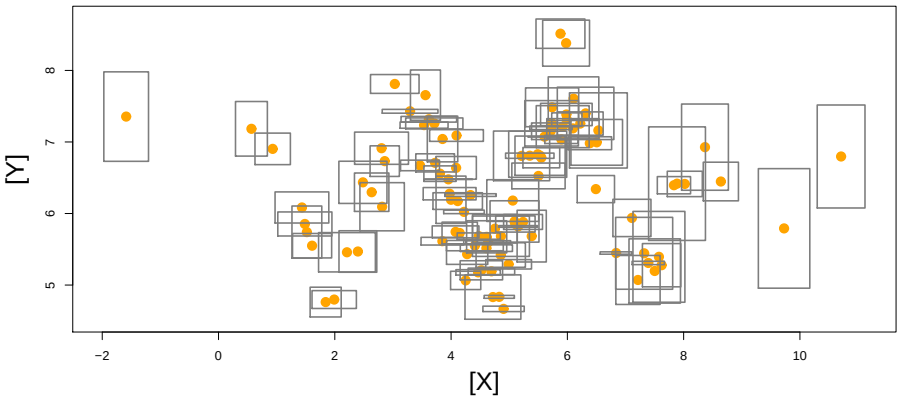

Figure 6: Plot of a simulated data set from setting 6.

radius of the response interval are determined by the equations

$$Y_i^C = (X_{1i}^C + (X_{1i}^C)^2)(X_{2i}^C + (X_{2i}^C)^2) - (X_{3i}^C + (X_{3i}^C)^2)(X_{4i}^C + (X_{4i}^C)^2) - X_{5i}^C + \epsilon_i^C,$$
$$Y_i^R = \frac{(X_{2i}^R)^2}{5} + 0.1X_{3i}^R - 5(X_{1i}^R X_{4i}^R + X_{5i}^R) + 4 + \epsilon_i^R,$$

where $\epsilon_i^C \sim N(0, 1^2)$ and $\epsilon_i^R \sim N(-3, 0.15^2)$, $i = 1, \cdots, n$.

The predictive accuracies for the center of the linear models (settings 1-4) are displayed in table 1 and plotted in figures 8 - 11. The performances of both CCRM and random forests generally improve for all of the four settings as the sample size increases. In the first two settings where the data show a strong linear association, random forest regression tree does not have any advantages over CCRM, - in fact, CCRM performs slightly better than random forests for these two settings. In setting 3, the response center is linearly associated with both the predictor center and radius, which CCRM is incapable of capturing but random forests can. Therefore, in this setting, random forest regression yields higher predictive accuracy over CCRM for the center. For the slightly nonlinear model in setting 4, the random forest regression outcompetes CCRM, due to its ability to handle nonlinearity.

Table 2 shows the predictive accuracies for the radius of the linear models. As for the center, the random forest regression is slightly worse than CCRM in setting 1 where the data follow a linear pattern with a strictly positive intercept. However, for linear models with a negative intercept in setting 2 and 3, CCRM returns poor predictive accuracy due to its constraint that the intercept must be positive, but random forests need not have this constraint and therefore perform much better. Also for the slightly nonlinear model in setting 4, random forests yield much better predictive accuracy than CCRM.

Table 1: Predictive Accuracy for Linear Models: Center (Settings 1-4)

		R^2		MSE		MAE	
Setting	n	CCRM	RF	CCRM	RF	CCRM	RF
1	50	**0.7928**	0.7161	**4.110**	5.650	**1.620**	1.890
	100	**0.7954**	0.7259	**4.050**	5.450	**1.610**	1.860
	200	**0.7977**	0.7278	**4.040**	5.430	**1.600**	1.860
2	50	**0.8349**	0.7868	**26.08**	33.62	**4.080**	4.630
	100	**0.8395**	0.7878	**25.26**	33.45	**4.010**	4.630
	200	**0.8410**	0.7894	**25.16**	33.31	**4.000**	4.610
3	50	0.7309	**0.8662**	883.6	**446.4**	25.53	**14.81**
	100	0.7385	**0.9276**	866.7	**241.6**	25.35	**10.55**
	200	0.7449	**0.9599**	854.6	**133.9**	25.22	**7.610**
4	50	0.5606	**0.7157**	1468.4	**981.9**	24.01	**18.46**
	100	0.5447	**0.7650**	1474.1	**834.4**	24.38	**17.50**
	200	0.5798	**0.7973**	1383.7	**661.6**	23.52	**17.29**

Table 2: Predictive Accuracy for Linear Models: Radius (Settings 1-4)

		R^2		MSE		MAE	
Setting	n	CCRM	RF	CCRM	RF	CCRM	RF
1	50	**0.7768**	0.7095	**0.0900**	0.1200	**0.2400**	0.2800
	100	**0.7829**	0.7139	**0.0900**	0.1200	**0.2400**	0.2800
	200	**0.7853**	0.7131	**0.0900**	0.1200	**0.2400**	0.2800
2	50	0.2671	**0.4144**	0.4300	**0.3400**	0.5300	**0.4700**
	100	0.2720	**0.4182**	0.4300	**0.3400**	0.5300	**0.4700**
	200	0.2766	**0.4178**	0.4200	**0.3400**	0.5200	**0.4600**
3	50	0.5901	**0.8126**	3.910	**1.790**	1.630	**1.070**
	100	0.5905	**0.8139**	3.970	**1.790**	1.640	**1.070**
	200	0.5940	**0.8144**	3.930	**1.790**	1.630	**1.070**
4	50	0.5685	**0.7434**	0.1100	**0.0700**	0.2700	**0.1900**
	100	0.5982	**0.7810**	0.1000	**0.0600**	0.2500	**0.1700**
	200	0.6098	**0.7970**	0.0900	**0.0500**	0.2500	**0.1600**

The center and radius results for nonlinear models are shown in table 3 and table 4, respectively. The predictions against the original data are plotted in figures 12 and 13. As for the linear models, as the sample size increases, both the kernel estimator and random forests improve with larger R^2s and smaller MSEs and MAEs. For the models with fewer predictors, such as setting 5 and 6, kernel estimator has competitive performance compared to random forest regression. For the multivariate model that has 5 predictors (setting 7), random forest regression has higher predictive accuracy than the kernel estimator, because the kernel-based estimator generally suffers from the "curse of dimensionality". Also, it is worth noting that, as the number of predictors rises, the kernel estimator rapidly becomes much more time consuming. For setting 7, the kernel estimator takes 28.12 seconds to run one simulation (for $n = 50, 100, 200$ totally), while random forests only take 1.08 seconds. (Shown in figure 7)

Table 3: Predictive Accuracy for Nonparametric Models: Center (Settings 5-7)

		R^2		MSE		MAE	
Setting	n	KE	RF	KE	RF	KE	RF
5	50	0.6185	**0.9030**	2.880	**0.7300**	0.7800	**0.5900**
	100	0.8288	**0.9366**	1.300	**0.4800**	0.5600	**0.5100**
	200	0.9010	**0.9480**	0.7500	**0.3900**	0.4800	**0.4800**
6	50	-0.0534	**0.2644**	0.8100	**0.5600**	0.7100	**0.6000**
	100	-0.2152	**0.3872**	0.9200	**0.4700**	0.6500	**0.5400**
	200	-0.0209	**0.4746**	0.7800	**0.4000**	0.5200	**0.5000**
7	50	**0.5679**	0.5276	**17838**	20370	**63.76**	99.59
	100	0.6254	**0.6403**	15742	**15380**	**55.82**	84.88
	200	0.6713	**0.7340**	13982	**11410**	**47.19**	71.20

5. Real Data Application

A real dataset is analyzed by the random forest regression to show its applicability. The dataset contains daily [min, max] stock price ranges for three companies, namely Boeing Aircraft Manufacturing Company (BA), General Electric (GE), and JPMorgan Chase (JPM), and the Down Jones Industrial Average index (DJIA),

Table 4: Predictive Accuracy for Nonlinear Models: Radius (Settings 5-7)

		R^2		MSE		MAE	
Setting	n	KE	RF	KE	RF	KE	RF
5	50	**0.6341**	0.5527	**0.0400**	0.0500	**0.1700**	0.1900
	100	**0.6497**	0.5610	**0.0400**	0.0500	**0.1700**	0.1900
	200	**0.6608**	0.5600	**0.0400**	0.0500	**0.1600**	0.1900
6	50	0.3425	**0.4546**	0.0200	**0.0200**	0.1000	**0.1000**
	100	0.4764	**0.5743**	0.0200	**0.0100**	0.0900	**0.0900**
	200	0.5419	**0.6328**	0.0100	**0.0100**	0.0800	**0.0800**
7	50	0.1321	**0.6065**	7.430	**3.520**	**1.250**	1.410
	100	0.1561	**0.7033**	9.440	**2.690**	1.230	**1.220**
	200	0.2326	**0.7900**	6.970	**1.900**	**0.9600**	1.000

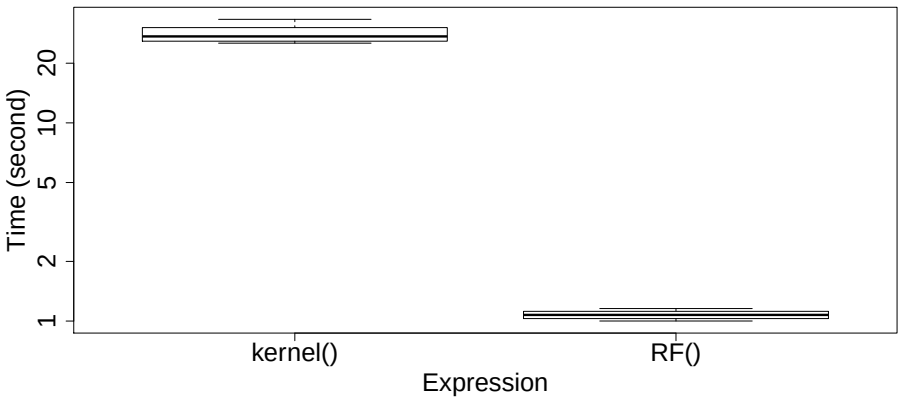

Figure 7: The Time Consumed for Each Function - Random Forest (RF) and Kernel Estimator (Kernel). The average time that the random forest function and kernel function for one analysis consumes is 1.08 and 28.12 seconds.

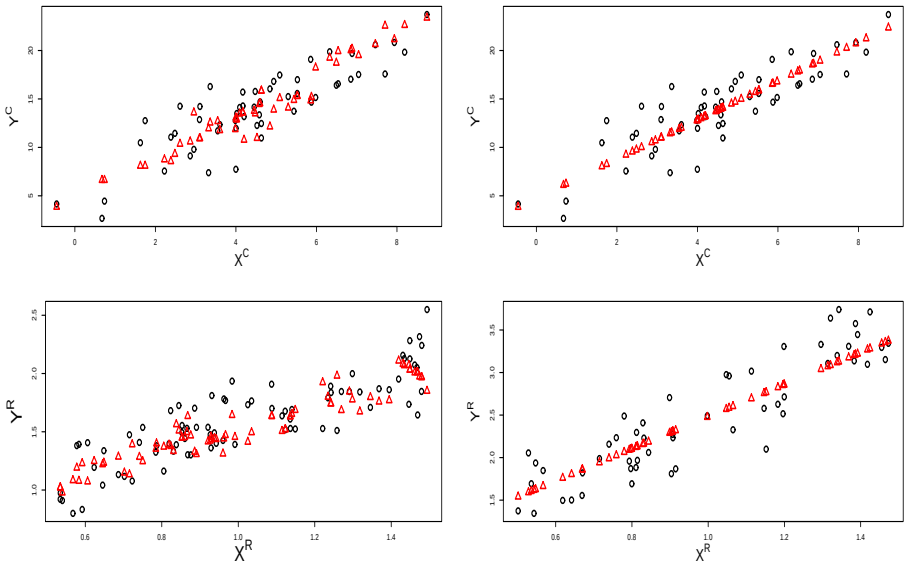

Figure 8: Overlaid plots of predicted centers and radii against the observed values from setting 1. The left two plots are for random forests and the right two are for CCRM. Observed data from the test set are represented by black circles and the predicted data are represented by red triangles (same for figures 9-13 in the following).

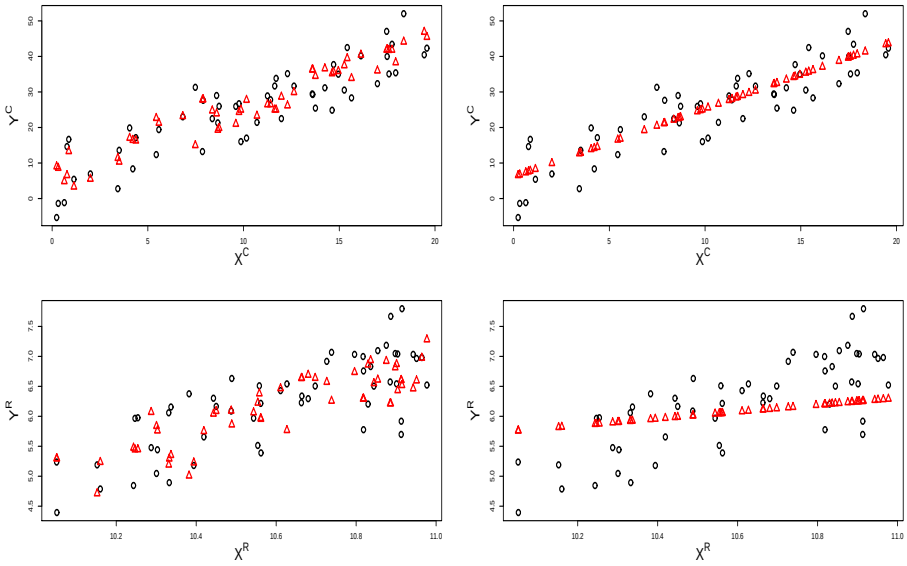

Figure 9: Overlaid plots of predicted centers and radii against the observed values from setting 2. The left two plots are for random forests and the right two are for CCRM.

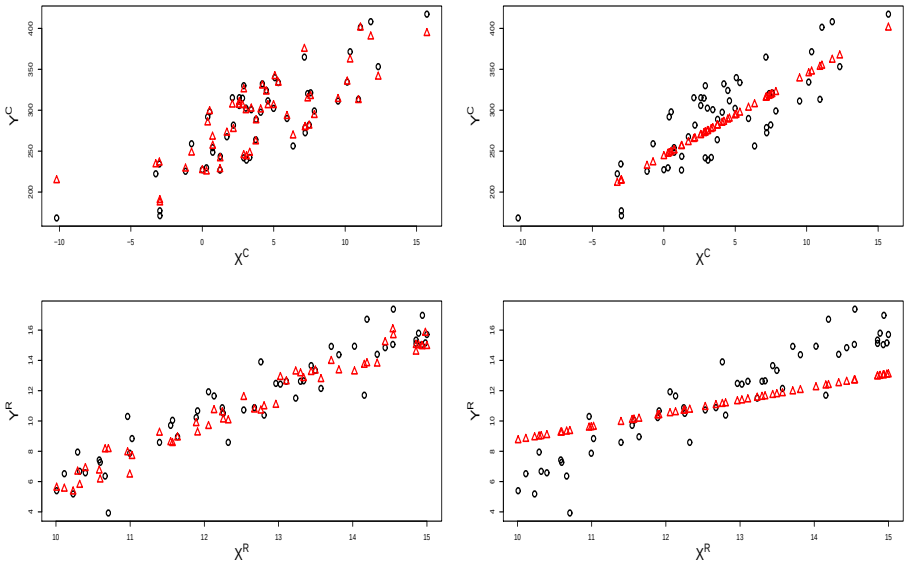

Figure 10: Overlaid plots of predicted centers and radii against the observed values from setting 3.The left two plots are for random forests and the right two are for CCRM.

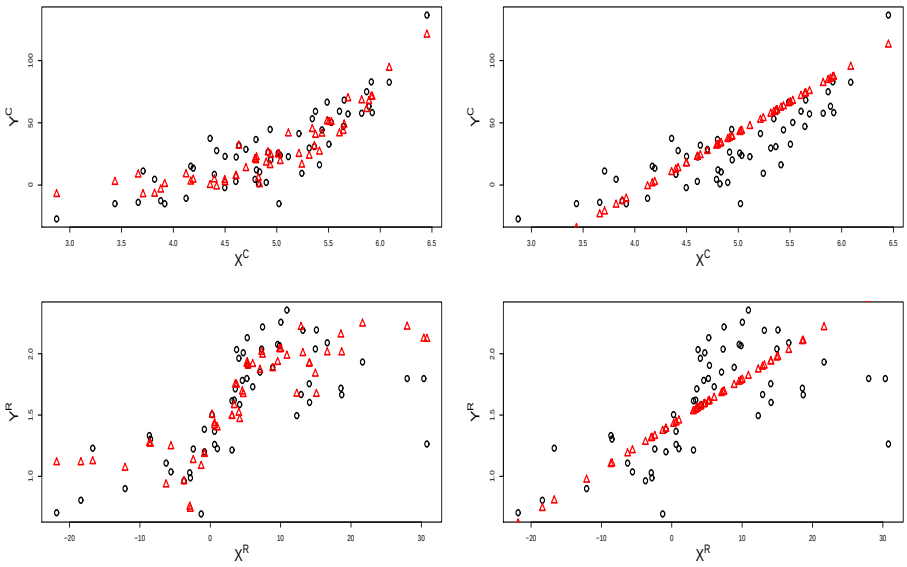

Figure 11: Overlaid plots of predicted centers and radii against the observed values from setting 4.The left two plots are for random forests and the right two are for CCRM.

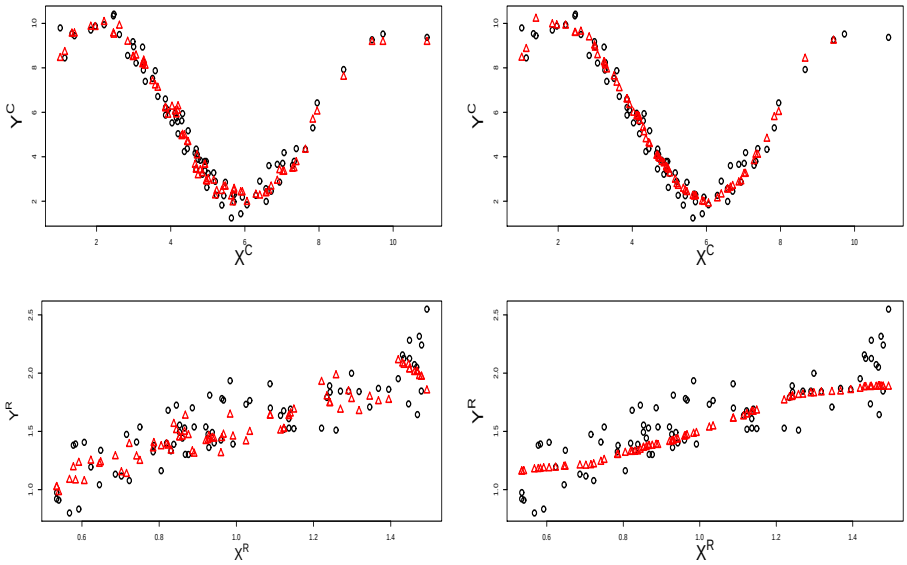

Figure 12: Overlaid plots of predicted centers and radii against the observed values from setting 5. The left two plots are for random forests and the right two are for kernel estimator.

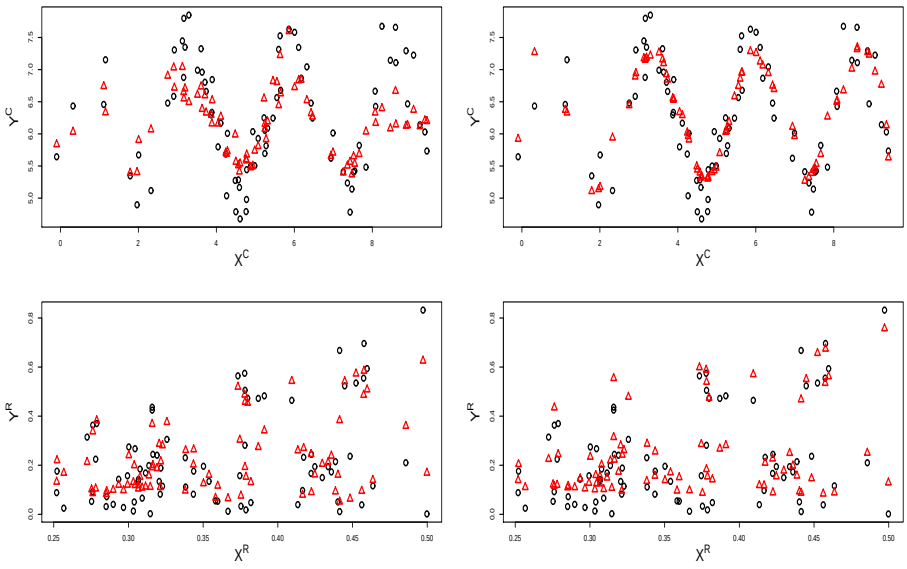

Figure 13: Overlaid plots of predicted centers and radii against the observed values from setting 6. The left two plots are for random forests and the right two are for kernel estimator.

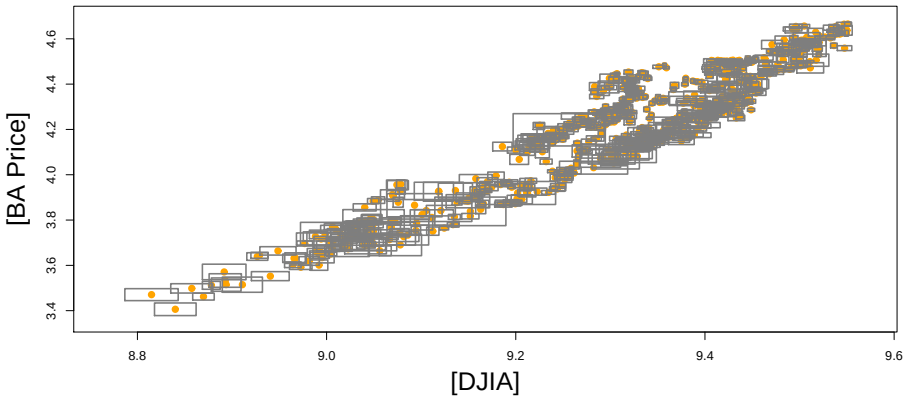

Figure 14: Plot of the Stock Price Interval for BA. The gray rectangles denote the interval-valued data, and the yellow dots are the corresponding centers.

with 1511 price intervals for each asset from Jan 3, 2006 to Dec 30, 2011. We split the data into a training set of 1208 (80%) and a test set of 303 (20%) intervals.

DJIA is a stock market index created by Wall Street Journal editor and Dow Jones & Company co-founder Charles Dow, to show how 30 large, publicly owned, companies based in the United States have traded during a standard trading session in the stock market. In our analysis, the DJIA is first used as the sole variable to predict each of the three individual stocks. As seen in figure 16, JPM (as well as other stocks) represents a pretty linear relationship with the DJIA index. So CCRM is first utilized as the baseline model. Then, random forest regressions are built for the centers and radii of the stocks, using both the center and radius of the DJIA index as predictors. Predictive accuracies are assessed in terms of the R^2, MSE and MAE on the test sets, and are reported in tables 5 and 6.

Predictive results for JPM price ranges are plotted in figure 19 for illustration. For the predicted centers of BA, GE and JPM, random forests achieve a consistently better performance compared to CCRM. As for the radii, CCRM obtains slightly better accuracy by 0.04 for the R^2 of BA due to its relatively strong linear relationship with the DJIA index. For GE and JPM, random forests predict the data more accurately, compared to CCRM. Based on these results, the DJIA index is capable of predicting the three stock price ranges, accounting for a variance from 63% to 97% for the center, and from 64% to 82% for the radius, by random forest regression. The comparisons to CCRM show that random forests generally outperform CCRM in terms of predictive accuracy.

The three selected stocks, namely BA, GE and JPM, are leading companies in aviation, manufacturing and financial industries, so these stocks together may explain a good portion of variability of the DJIA index. In the second part of our

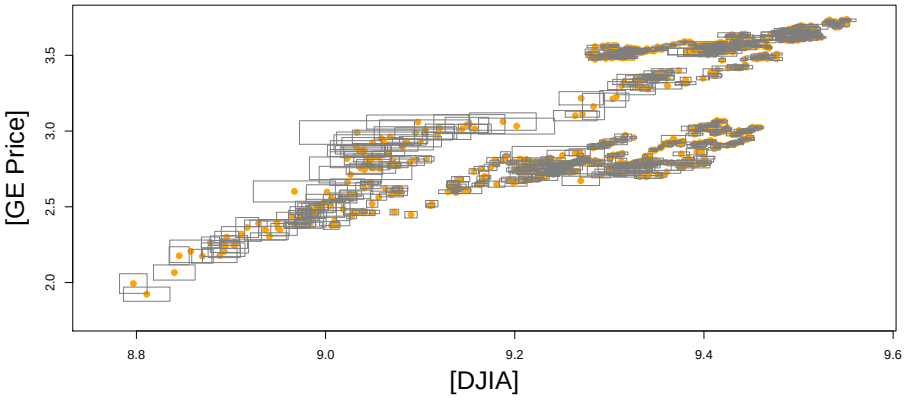

Figure 15: Plot of the Stock Price Interval for GE. The gray rectangles denote the interval-valued data, and the yellow dots are the corresponding centers.

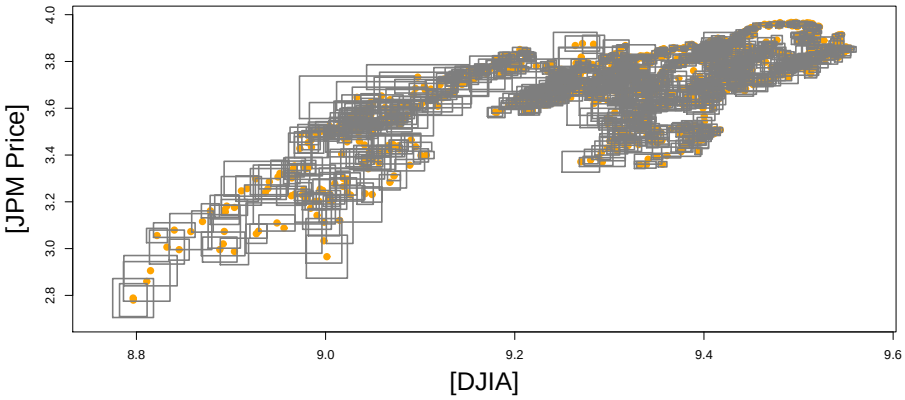

Figure 16: Plot of the Stock Price Interval for JPM. The gray rectangles denote the interval-valued data, and the yellow dots are the corresponding centers.

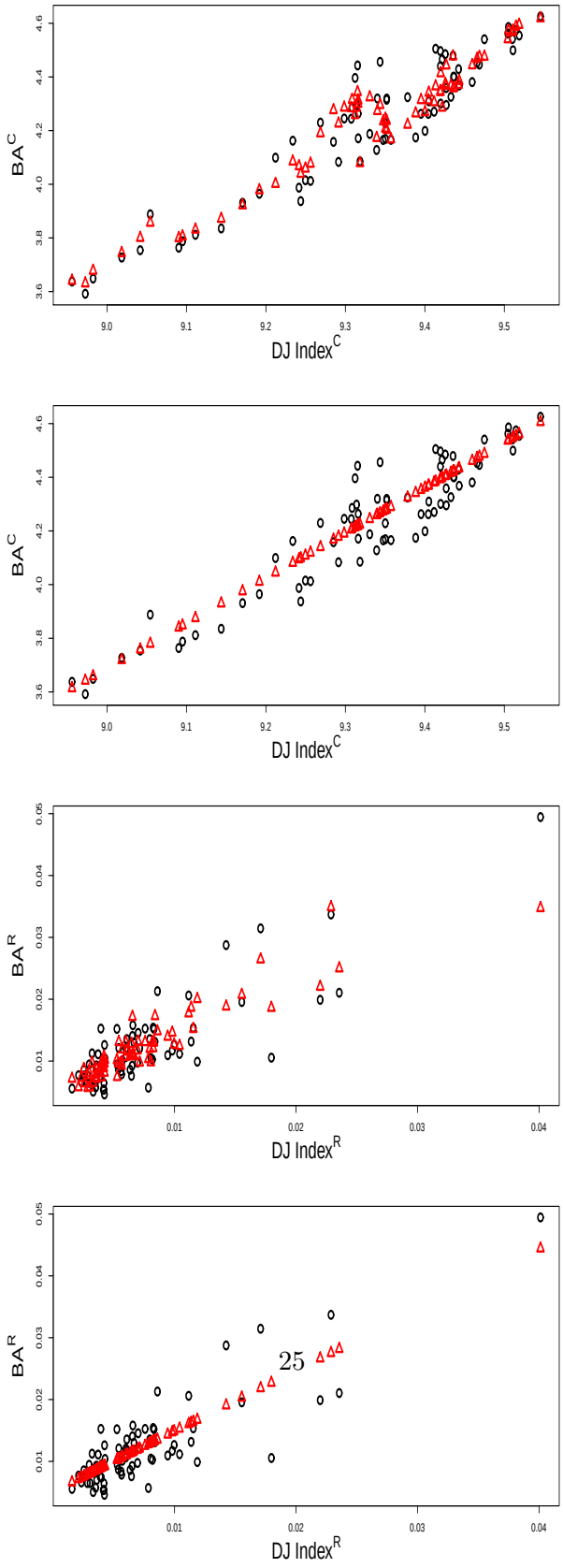

Figure 17: Overlaid plots of predicted centers and radii against the observed values for BA. Observed data from the test set are represented by black circles and the predicted data are represented by red triangles.

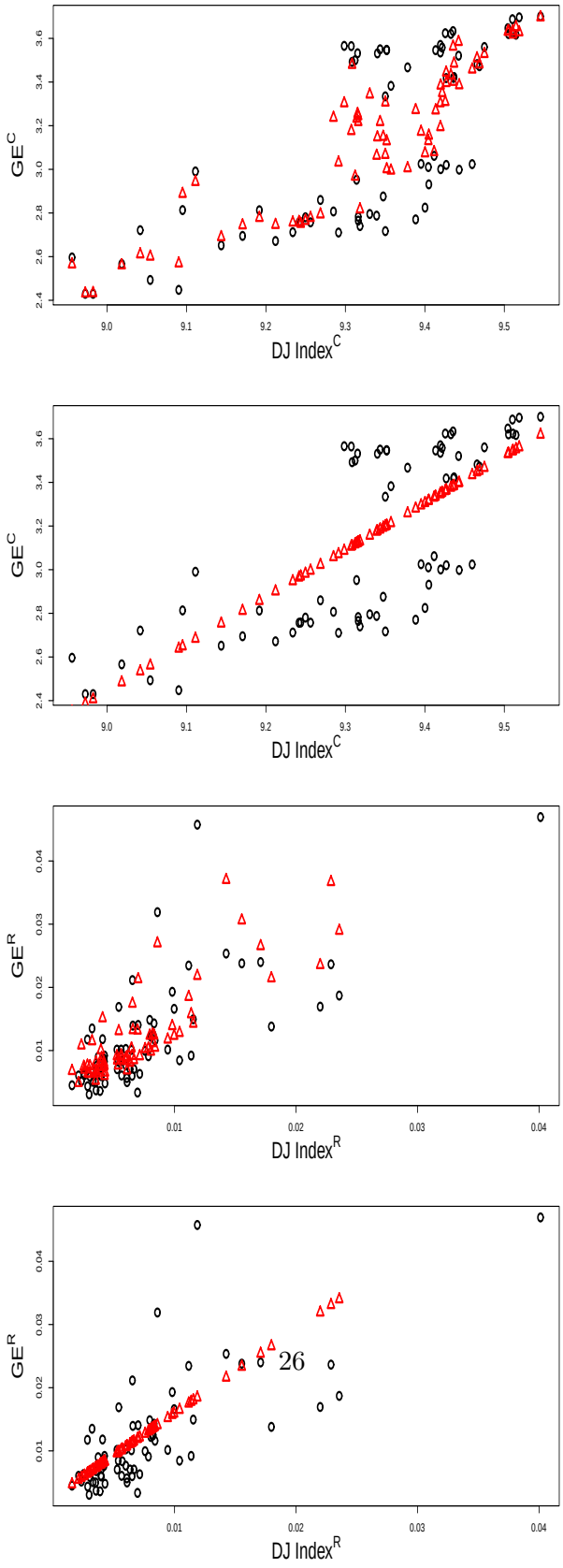

Figure 18: Overlaid plots of predicted centers and radii against the observed values for GE. Observed data from the test set are represented by black circles and the predicted data are represented by red triangles.

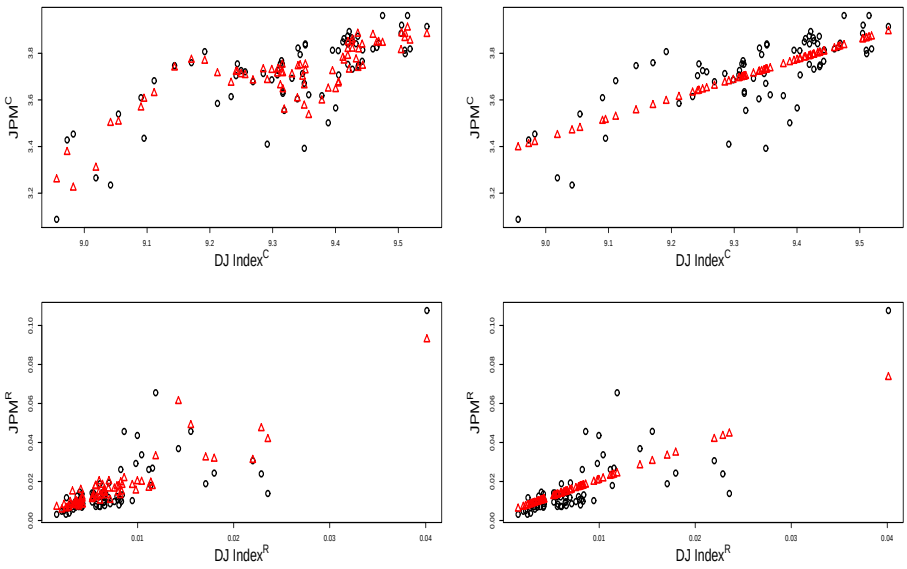

Figure 19: Overlaid plots of predicted centers and radii against the observed values for JPM. Observed data from the test set are represented by black circles and the predicted data are represented by red triangles.

analysis, we build predictive models for the DJIA based on the three stocks jointly. The multivariate analysis is run using both random forests and CCRM. As in the previous part, random forests include both the centers and radii of the stocks as the predictors for both the center and radius of the DJIA index. As the results show in table 5 and table 6, for this multivariable analysis, random forests achieve better performance than CCRM. This is partly due to the extra predictor variables included in the random forests regression equations. In addition, the ability of random forests to account for nonlinearity is an advantage over CCRM.

6. Conclusion

We proposed random forest regression for interval-valued data. Its nonparametric nature automatically ensures mathematical coherence without any constraints, making it more flexible than most of the existing regression methods such as CCRM. In addition to linear problems, it also handles nonlinearity very well. As most nonparametric methods require extensive tuning to achieve optimal performance, random forests are especially user-friendly in the sense that very little tuning is needed. Furthermore, they are robust against overfitting and high-dimensionality. All these advantages of random forest regression make it a potentially very promising method to be extended to interval-valued data. In this project, we adopted

Table 5: Predictive Accuracy for Asset Prices: Center

	R^2		MSE		MAE	
	CCRM	RF	CCRM	RF	CCRM	RF
BA	0.8655	**0.9005**	0.0089	**0.0066**	0.0806	**0.0655**
GE	0.5946	**0.6259**	0.0699	**0.0644**	0.2353	**0.1880**
JPM	0.5739	**0.6904**	0.0109	**0.0079**	0.0818	**0.0655**
DJIA Index	0.8682	**0.9665**	0.0026	**0.0007**	0.0441	**0.0185**

Table 6: Predictive Accuracy for Asset Prices: Radius

	R^2		MSE		MAE	
	CCRM	RF	CCRM	RF	CCRM	RF
BA	**0.6785**	0.6357	**1.87E-05**	2.11E-05	**0.0031**	0.0032
GE	0.5311	**0.7196**	7.44E-05	**4.45E-05**	0.0043	**0.0038**
JPM	0.6522	**0.7150**	7.91E-05	**6.48E-05**	0.0059	**0.0054**
DJIA Index	0.7708	**0.8164**	1.08E-05	**8.69E-06**	0.0021	**0.0020**

bivariate regression. Simulation results confirmed its advantages over CCRM and the kernel estimator. Future research includes regression with multivariate response, i.e., center and radius jointly, to account for their potential correlation.

Appendix A. Random Sets Preliminaries

Denote by $\mathcal{K}\left(\mathbb{R}^d\right)$ or $\mathcal{K}$ the collection of all non-empty compact subsets of $\mathbb{R}^d$. The Hausdorff metric

$$\rho_H(A,B) = \max\left(\sup_{a\in A}\rho(a,B), \sup_{b\in B}\rho(b,A)\right),\ \forall A,B\in\mathcal{K},$$

where ρ denotes the Euclidean metric, defines a metric in $\mathcal{K}\left(\mathbb{R}^d\right)$. As a metric space, $\mathcal{K}\left(\mathbb{R}^d\right)$ is complete and separable (Debreu (1967)). In the space $\mathcal{K}$, a linear structure can be defined by Minkowski addition and scalar multiplication as

$$A+B=\{a+b: a\in A, b\in B\}, \qquad \lambda A=\{\lambda a: a\in A\}, \tag{A.1}$$

$\forall A,B\in\mathcal{K}$ and $\lambda\in\mathbb{R}$. However, $\mathcal{K}\left(\mathbb{R}^d\right)$ is not a linear space (or vector space), as there is no inverse element of addition.

Let $(\Omega,\mathcal{L},P)$ be a probability space. A random compact set is a Borel measurable function $A:\Omega\to\mathcal{K}$, $\mathcal{K}$ being equipped with the Borel σ-algebra induced by the Hausdorff metric. If $A(\omega)$ is convex almost surely, then A is called a random compact convex set. (Molchanov (2005), p.21, p.102.) The collection of all compact convex subsets of $\mathbb{R}^d$ is denoted by $\mathcal{K}_{\mathcal{C}}\left(\mathbb{R}^d\right)$ or $\mathcal{K}_{\mathcal{C}}$. Much of the random sets theory has focused on compact convex sets (Artstein and Vitale (1975), Aumann (1965), and Lyashenko (1982), Lyashenko (1983)). Especially, when $d=1$, $\mathcal{K}_{\mathcal{C}}(\mathbb{R})$ contains all the non-empty bounded closed intervals in $\mathbb{R}$. A measurable function $[X]:\Omega\to\mathcal{K}_{\mathcal{C}}(\mathbb{R})$ is called a random interval. The expectation of a random compact convex random set A is defined by the Aumann integral of set-valued function (see Aumann (1965), Artstein and Vitale (1975)) as

$$E(A)=\{E\xi:\xi\in A \text{ almost surely}\}.$$

In particular, the Aumann expectation of a random interval $[X]$ is given by

$$E([X])=[E\left(X^L\right),E\left(X^U\right)]. \tag{A.2}$$

For each $X\in\mathcal{K}\left(\mathbb{R}^d\right)$, the function defined on the unit sphere S^{d-1}:

$$s_X(u)=\sup_{x\in X}\langle u,x\rangle, \quad \forall u\in S^{d-1}$$

is called the support function of X. Let $\mathcal{S}$ be the space of support functions of all non-empty compact convex subsets in $\mathcal{K}_{\mathcal{C}}$. Then, $\mathcal{S}$ is a Banach space equipped with the L_2 metric

$$\|s_X(u)\|_2=\left[\int_{S^{d-1}}|s_X(u)|^2\mu(\mathrm{d}u)\right]^{\frac{1}{2}},$$

where μ is the normalized Lebesgue measure on S^{d-1}. According to various embedding theorems (Radstrom (1952); Hormander (1954)), $\mathcal{K}_{\mathcal{C}}$ can be embedded isometrically into the Banach space $C(S)$ of continuous functions on S^{d-1}, and $\mathcal{S}$ is the image of $\mathcal{K}_{\mathcal{C}}$ into $C(S)$. Therefore, $\delta(X,Y) := \|s_X - s_Y\|_2$, $\forall X, Y \in \mathcal{K}_{\mathcal{C}}$, defines an L_2 metric on $\mathcal{K}_{\mathcal{C}}$. It is well known that ρ_H and δ are equivalent metrics, but ρ_H is less preferred for statistical analysis for several reasons. As an important one, $E\rho_H^2(X, h(X))$ is not minimized at $h(X) = E(X)$. Particularly the δ-metric for an interval $[x] \in \mathcal{K}_{\mathcal{C}}(\mathbb{R})$ is

$$\|[x]\|_2 = \|s_{[x]}(u)\|_2 = \frac{1}{2}\left(x^L\right)^2 + \frac{1}{2}\left(x^U\right)^2 = \left(x^C\right)^2 + \left(x^R\right)^2,$$

and the δ-distance between two intervals $[x], [y] \in \mathcal{K}_{\mathcal{C}}(\mathbb{R})$ is

$$\begin{aligned} \delta([x],[y]) &= \left[\frac{1}{2}\left(x^L - y^L\right)^2 + \frac{1}{2}\left(x^U - y^U\right)^2\right]^{\frac{1}{2}} \\ &= \left[\left(x^C - y^C\right)^2 + \left(x^R - y^R\right)^2\right]^{\frac{1}{2}}. \end{aligned}$$

Gil et al. (2001) generalized the δ-distance to the W-distance as

$$d_W([x],[y]) = \left\{\int_{[0,1]} \left[f_{[x]}(\lambda) - f_{[y]}(\lambda)\right]^2 dW(\lambda)\right\}^{\frac{1}{2}}, \tag{A.3}$$

where $f_{[x]}(\lambda) = \lambda x^U + (1-\lambda)x^L$, $\forall \lambda \in [0,1]$, and W is any non-degenerate symmetric measure on $[0,1]$. The advantage of the W-distance lies in its flexibility to assign weights to the points in the interval. In particular, this can be interpreted as a probability distribution for a random point inside the interval. On the other hand, it can be shown that

$$d_W^2([x],[y]) = \left(x^C - y^C\right)^2 + \left(x^R - y^R\right)^2 \int_{[0,1]} (2\lambda - 1)^2 dW(\lambda). \tag{A.4}$$

Notice that $\int_{[0,1]} (2\lambda - 1)^2 dW(\lambda) \in [0,1]$ is a constant determined by W. So the W-distance can also be interpreted as choosing a weight for $\left(X^R - Y^R\right)^2$ in calculating the L_2 distance.

Appendix B. R Implementation

- makeset: Given the centers and the ranges of X and Y, the function calculates and returns the endpoints of X and Y intervals.

```
makeset <- function(Xc, Xr, Yc, Yr) {
    set <- data.frame(Xc, Xr, Yc, Yr)
    set$xL <- set$Xc - set$Xr
    set$xR <- set$Xc + set$Xr
    set$yL <- set$Yc - set$Yr
    set$yU <- set$Yc + set$Yr
    return(set)
}
```

- recplot: Given a list of datasets with different sample size, the function plots a rectagle composed of the X intervel and Y interval in gray with a center point of (Xc, Yc) in orange.

```
recplot <- function(dataset, AXES = TRUE, p = 1, ...) {

    datasetsample <- dataset[sample(nrow(dataset), p * nrow(dataset),
        replace = FALSE), ]

    # Generate rectangle
    plot(datasetsample$Xc, datasetsample$Yc, col = "orange",
        axes = AXES, ylim = c(min(datasetsample$yL) - 0.01, max(v$yU) +
            0.01), xlim = c(min(datasetsample$xL) - 0.01, max(datasetsample$xR) +
            0.01), pch = 19, ...)


    rect(datasetsample$xL, datasetsample$yL, datasetsample$xR,
        datasetsample$yU, density = 0, col = "grey50", ...)
}
```

- RFLoop: Apply *randomforest* function to a list of datasets with different sample size.

```
RFLoop <- function(formula, datalist, ...) {
    RFlist <- lapply(1:length(datalist), function(i) {
        randomForest(formula, data = datalist[[i]][[1]], ntree = 1000,
            ...)
    })
    names(RFlist) <- names(datalist)
    return(RFlist)
}
```

- RsquareNON: Calculate the R-squared (coefficient of determination) for the test sets, using the models trained with the training sets. The predictions are aquired from the training models.
  $R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{\sum (Yc - \hat{Yc})^2}{\sum (Yc - \bar{Yc})^2}$.

```
RsquareNON <- function(set, Cpredmodel, Rpredmodel = NULL){

  # Replicate set[[i]] if there is only one dataset in the list
  if (length(set) != length(Cpredmodel)){
    set <- lapply(1: length(Cpredmodel), function(i){
    set
    })
  }


  # For CCRM
  if (all(lapply(Cpredmodel, class)== "ccrm" && is.null(Rpredmodel) )){
    cat("CCRM", "\n")
    Rlist <- lapply(1: length(set), function(i){
        setpred <- predsetccrm(set[[i]][[2]], Cpredmodel[[i]])

        #R-square for Yc
        STc <- (set[[i]][[2]]$Yc - mean(set[[i]][[2]]$Yc))^2
        #SSTc <- var(set[[i]]$Yc) * (nrow(set[[i]])-1)
        SEc <- (set[[i]][[2]]$Yc - setpred$Yc)^2
        Rsqc <- 1 - sum(SEc) / sum(STc)

        #R-square for Yr
        STr <- (set[[i]][[2]]$Yr - mean(set[[i]][[2]]$Yr))^2
        #SSTr <- var(set[[i]]$Yr) * (nrow(set[[i]])-1)
        SEr <- (set[[i]][[2]]$Yr - setpred$Yr)^2
        Rsqr <- 1 - sum(SEr) / sum(STr)


        Rmatrix <- matrix(c(round(Rsqc*100, 2), round(Rsqr*100, 2)), , 2)
        colnames(Rmatrix) <- c("center", "radius")
        Rmatrix
      })
  }
  # For Kernal or RF
  else{
    cat("Kernal/RF", "\n")
    Rlist <- lapply(1:length(set), function(i) {
```

```
    STc <- var(set[[i]][[2]]$Yc) * (nrow(set[[i]][[2]])-1)
    SEc <- (set[[i]][[2]]$Yc - predict(Cpredmodel[[i]],
                                       newdata = set[[i]][[2]]))^2
    Rsqc <- 1 - sum(SEc) / sum(STc)

    # R-square for Yr
    STr <- var(set[[i]][[2]]$Yr) * (nrow(set[[i]][[2]])-1)
    SEr <- (set[[i]][[2]]$Yr - predict(Rpredmodel[[i]],
                                       newdata = set[[i]][[2]]))^2
    Rsqr <- 1 - sum(SEr) / sum(STr)

    Rmatrix <- matrix(c(round(Rsqc*100, 2), round(Rsqr*100, 2)), , 2)
        colnames(Rmatrix) <- c("center", "radius")
        Rmatrix
})
  }

     # lapply(Rmatrix, setNames, c("R^2 for c %", "R^2 for r %"))
     names(Rlist) <- names(Cpredmodel)
     return(Rlist)

}
```

- Predregplot: The function plots the predicted intervals on top of the originial interva rectangles.

```
predregplot <- function(datalist, Cpredmodel, Rpredmodel = NULL, p = 1,  ...)
  {
  #Only draw p*100% of rectangles
  par(mfrow = c(1, length(datalist)))
    # For CCRM
  if (all(lapply(Cpredmodel, class)== "ccrm" && is.null(Rpredmodel) )){
    cat("CCRM", "\n")
    lapply(1: length(datalist), function(i){
      setpred <- predsetccrm(datalist[[i]], Cpredmodel[[i]])

      set.seed(13543)
      setpredsample <- setpred[sample(nrow(setpred), p*nrow(setpred),
                                      replace = FALSE),]

      recplot(datalist[[i]], p = p   , main = "Predicted vs. Actual Data")
      mtext(names(datalist)[i])
```

```
      points(datalist[[i]]$Xc, setpred$Yc, col = "blue", bg = TRUE, pch = 24)

      rect(setpredsample$xL, setpredsample$yL, setpredsample$xR,
           setpredsample$yU, density = 20, col = "blue", angle = -30, ...)

      lines(datalist[[i]]$Xc, setpred$Yc, lty = 1, col = "green")
    })
  }
  # For Kernal or RF
  else{
  lapply(1: length(datalist), function(i){
    setpred <- makeset(datalist[[i]]$Xc, datalist[[i]]$Xr,
                       predict(Cpredmodel[[i]]), predict(Rpredmodel[[i]]))


    set.seed(13543)
    setpredsample <- setpred[sample(nrow(setpred), p*nrow(setpred),
                                    replace = FALSE),]

    recplot(datalist[[i]], p = p   , main = "Predicted vs. Actual Data")
    mtext(names(datalist)[i])
    points(setpred$Xc, setpred$Yc, col = "blue", bg = TRUE, pch = 24)

    rect(setpredsample$xL, setpredsample$yL, setpredsample$xR,
         setpredsample$yU, density = 20, col = "blue", angle = -30, ...)
  })
  }

  on.exit(par(mfrow = c(1, length(datalist))))
}
```

- predsetccrm: The function generates a dataset with the original centers and ranges of the predictor (X) and the predicted centers and ranges of the response (Y), using CCRM models trained with the training sets.

```
predsetccrm <- function(set, ccrmmodel) {
    # Yc
    coefc <- ccrmmodel$coefficients.C[-1]
    varc <- set[names(ccrmmodel$coefficients.C)[-1]]
    Yc <- ccrmmodel$coefficients.C[[1]] + coefc %*% t(varc)
    # Yr Extract variables' names
    varnames <- as.character(sapply(strsplit(gsub("\\+", "",
        gsub("~", "", as.character(ccrmmodel$call)[3])), " "),
```

```
        function(x) {
            x[!x == ""]
        }))[-1]

    coefr <- ccrmmodel$coefficients.R[-1]
    varR <- set[varnames]
    Yr <- ccrmmodel$coefficients.R[[1]] + coefr %*% t(varR)

    predccrmset <- makeset(set$Xc, set$Xr, as.numeric(Yc), as.numeric(Yr))
    return(predccrmset)
}
```

- MAE: Calculate the mean absolute error for the test sets, using the models trained with the training sets. The predictions are aquired from the training models. $MAE = \frac{\sum |Yc - \hat{Yc}|}{n}$.

```
MAE <- function(set, Cpredmodel, Rpredmodel = NULL) {
    # Replicate set[[i]] if there is only one dataset in the list
    if (length(set) != length(Cpredmodel)) {
        set <- lapply(1:length(Cpredmodel), function(i) {
            set
        })
    }

    # For CCRM
    if (all(lapply(Cpredmodel, class) == "ccrm" && is.null(Rpredmodel))) {
        cat("CCRM", "\n")
        MAElist <- lapply(1:length(set), function(i) {

            setpred <- predsetccrm(set[[i]][[2]], Cpredmodel[[i]])

            # MAE for YC
            AEc <- abs(set[[i]][[2]]$Yc - setpred$Yc)
            MAEc <- sum(AEc)/nrow(set[[i]][[2]])

            # MAE for YC
            AEr <- abs(set[[i]][[2]]$Yr - setpred$Yr)
            MAEr <- sum(AEr)/nrow(set[[i]][[2]])


            MAE <- matrix(c(MAEc, MAEr), 1, 2)
            colnames(MAE) <- c("center", "radius")
            MAE
```

```
        })
    } else {
        MAElist <- lapply(1:length(set), function(i) {
            cat("Kernal/RF", "\n")
            # MAE for Yc
            AEc <- abs(set[[i]][[2]]$Yc - predict(Cpredmodel[[i]],
                newdata = set[[i]][[2]]))
            MAEc <- sum(AEc)/nrow(set[[i]][[2]])

            # MAE for Yc
            AEr <- abs(set[[i]][[2]]$Yr - predict(Rpredmodel[[i]],
                newdata = set[[i]][[2]]))
            MAEr <- sum(AEr)/nrow(set[[i]][[2]])

            MAE <- matrix(c(MAEc, MAEr), 1, 2)
            colnames(MAE) <- c("center", "radius")
            MAE
        })
    }
    # colnames(MAE) <- c('MAEc', 'MAEr')
    names(MAElist) <- names(set)
    return(MAElist)
}
```

- MSE: Calculate the mean squared error for the test sets, using the models trained with the training sets. The predictions are aquired from the training models. $MSE = \frac{\sum (Yc - \hat{Y}c)^2}{n}$.

```
MSE <- function(set, Cpredmodel, Rpredmodel = NULL) {
    # Replicate set[[i]] if there is only one dataset in the list
    if (length(set) != length(Cpredmodel)) {
        set <- lapply(1:length(Cpredmodel), function(i) {
            set
        })
    }

    # For CCRM
    if (all(lapply(Cpredmodel, class) == "ccrm" && is.null(Rpredmodel))) {
        cat("CCRM", "\n")
        MSElist <- lapply(1:length(set), function(i) {

            setpred <- predsetccrm(set[[i]][[2]], Cpredmodel[[i]])
```

```
            # MSE for YC
            SEc <- (set[[i]][[2]]$Yc - setpred$Yc)^2
            MSEc <- sum(SEc)/nrow(set[[i]][[2]])

            # MSE for Yr
            SEr <- (set[[i]][[2]]$Yr - setpred$Yr)^2
            MSEr <- sum(SEr)/nrow(set[[i]][[2]])


            MSE <- matrix(c(MSEc, MSEr), 1, 2)
            colnames(MSE) <- c("center", "radius")
            MSE
        })
    } else {
        MSElist <- lapply(1:length(set), function(i) {
            # MSE for Yc
            SEc <- (set[[i]][[2]]$Yc - predict(Cpredmodel[[i]],
                newdata = set[[i]][[2]]))^2
            MSEc <- sum(SEc)/nrow(set[[i]][[2]])

            # MSE for Yc
            SEr <- (set[[i]][[2]]$Yr - predict(Rpredmodel[[i]],
                newdata = set[[i]][[2]]))^2
            MSEr <- sum(SEr)/nrow(set[[i]][[2]])

            MSE <- matrix(c(MSEc, MSEr), 1, 2)
            colnames(MSE) <- c("center", "radius")
            MSE
        })
    }
    # colnames(MAE) <- c('MAEc', 'MAEr')
    return(MSElist)
}
```

- Simulation example: Setting 1

```
require(randomForest)
require(iRegression)
```

```
nvector <- c(50, 100, 200)
totalsize <- nvector/0.1

set.seed(6458)
set1data <- lapply(1:length(nvector), function(i) {
```

```
    # Total Dataset


    # Xc
    Xc1 <- rnorm(totalsize[i], 5, 2)

    # Yc
    epsilon1_1 <- rnorm(totalsize[i], 0, 2)
    Yc1 <- 2 * Xc1 + 5 + epsilon1_1

    # Xr
    Xr1 <- runif(totalsize[i], 0.5, 1.5)

    # Yr

    epsilon1_2 <- rnorm(totalsize[i], 0.5, 0.3)
    Yr1 <- 2 * Xr1 + epsilon1_2


    while (any(Yr1 < 0)) {
        epsilon1_2 <- rnorm(totalsize[i], 0.5, 0.3)
        Yr1 <- 2 * Xr1 + epsilon1_2
    }

    dataset1 <- makeset(Xc1, Xr1, Yc1, Yr1)

    # Trainingset
    trainingsample <- sample(nrow(dataset1), nvector[i])

    training1 <- dataset1[trainingsample, ]

    # Testset
    test1 <- dataset1[-trainingsample, ]

    # c(training1, test1)
    return(list(training1, test1))



})
names(set1data) <- paste("n =", nvector)
```

2. Visulization

```
recplot(set1data$`n = 200`[[2]], main = "Simulation Setting 1",
    p = 0.3, position = "bottomright")
```

3. RF

```
# Yc
set.seed(5432)
set1RFc <- RFLoop(Yc ~ Xc, set1data)

# Yr
set.seed(458)
set1RFr <- RFLoop(Yr ~ Xr, set1data)
```

4. CCRM

```
ccrm1 <- lapply(1:length(set1data), function(i) {
    ccrm("Yc ~ Xc", "Yr ~ Xr", data = set1data[[i]][[1]])
})
names(ccrm1) <- names(set1data)
```

4.1 Generate predicted dataset containing Xc, Xr, $\hat{Y}c$ and $\hat{Y}r$

```
predset1 <- predsetccrm(set1data$`n = 200`[[2]], ccrm1[[3]])
```

5. R^2

```
# RF
RsquareNON(set1data, set1RFc, set1RFr)

# CCRM
RsquareNON(set1data, ccrm1)
```

- Simulation example: Setting 7

```
nvector <- c(50, 100, 200)
totalsize <- nvector/0.1

set.seed(55133)
set7data <- lapply(1:length(nvector), function(i) {

    # Total Dataset

    Xc7_1 <- rnorm(totalsize[i], 5, 3)
    Xc7_2 <- rbeta(totalsize[i], 0.5, 0.5)
    Xc7_3 <- rnorm(totalsize[i], 10, 3.5)
    Xc7_4 <- runif(totalsize[i], 0.5, 1.5)
    Xc7_5 <- rnorm(totalsize[i], 8, 3.5)
```

```
    # U
    u1 <- runif(totalsize[i], 0, 0.5)
    u2 <- runif(totalsize[i], 0, 0.5)

    # tao
    tao1 <- rnorm(totalsize[i], 0, 0.2)
    tao2 <- rnorm(totalsize[i], 0, 0.2)

    # V
    V1 <- u1 + exp(-0.5 * rgamma(totalsize[i], 3, 2) + tao1)
    V2 <- u2 + exp(-0.5 * rbeta(totalsize[i], 1, 3) + tao2)


    # Xr
    Xr7_1 <- 2 * V1/(1 + V1)
    Xr7_2 <- 3 * V2/(1 + V2)
    Xr7_3 <- rnorm(totalsize[i], 10, 3)
    Xr7_4 <- runif(totalsize[i], 2.5, 3.5)
    Xr7_5 <- rbeta(totalsize[i], 2, 5)

    # Yc
    epsilon7_1 <- rnorm(totalsize[i], 0, 1)
    Yc7 <- (Xc7_1 + Xc7_1^2) * (Xc7_2 + Xc7_2^2) - (Xc7_3 + Xc7_3^2) *
        (Xc7_4 + Xc7_4^2) - Xc7_5 + epsilon7_1

    # Yr
    epsilon7_2 <- rnorm(totalsize[i], -3, 0.1)
    Yr7 <- (Xr7_2)^2/5 + (0.1 * Xr7_3) - 5 * (Xr7_1 * Xr7_4 +
        Xr7_5) + 4 + epsilon7_2

    while (length(Yr7) != totalsize[i]) {
        epsilon7_2_com <- rnorm((totalsize[i] - length(Yr7)),
            -3, 0.1)
        Yr7 <- c(Yr7, ((Xr7_2)^2/5 + (0.1 * Xr7_3) - 5 * (Xr7_1 *
            Xr7_4 + Xr7_5) + 4 + epsilon7_2_com))
        Yr7 <- Yr7[which(Yr7 >= 0)]
    }
    dataset7 <- makeset(Xc7_1, Xr7_1, Yc7, Yr7)
    dataset7 <- cbind(dataset7, Xc7_2, Xc7_3, Xc7_4, Xc7_5, Xr7_2,
        Xr7_3, Xr7_4, Xr7_5)
    # Trainingset
    trainingsample <- sample(nrow(dataset7), nvector[i])

    training7 <- dataset7[trainingsample, ]
```

```
    # Testset
    test7 <- dataset7[-trainingsample, ]

    # c(training7, test7)
    return(list(training7, test7))



})
names(set7data) <- paste("n =", nvector)
```

3. Random Forest

a. Yc ~ Xc_1 + Xc_2 + Xc_3

```
set.seed(6325)
set7RFc <- RFLoop(Yc ~ Xc + Xc7_2 + Xc7_3 + Xc7_4 + Xc7_5, set7data)
```

b. Yr ~ Xc_1 + Xc_2

```
set.seed(6532)
set7RFr <- RFLoop(Yr ~ Xr + Xr7_2 + Xr7_3 + Xr7_4 + Xr7_5, set7data)
```

4. Kenel Estimator

- Bandwidth selection

```
# Yc
bwYc7k <- lapply(1:length(set7data), function(i) {
    npregbw(Yc ~ Xc + Xc7_2 + Xc7_3 + Xc7_4 + Xc7_5, data = set7data[[i]][[1]],
        bwmethod = "cv.aic", ckertype = "uniform")
})
names(bwYc7k) <- names(set7kdata)

# Yr
bwYr7k <- lapply(1:length(set7data), function(i) {
    npregbw(Yr ~ Xr + Xr7_2 + Xr7_3 + Xr7_4 + Xr7_5, data = set7data[[i]][[1]],
        bwmethod = "cv.aic", ckertype = "uniform")
})
names(bwYr7k) <- names(set7kdata)
```

- KE

a. Yc: Kernal Univariate Regression

```
# Local Constant
modelYc.np7k <- lapply(1:length(set7data), function(i) {
    npreg(bwYc7k[[i]], data = set7data[[i]][[1]], y.eval = TRUE)
```

```
})
```

b. Yr: Kernal Univariate Regression

```
# Local Constant
modelYr.np7k <- lapply(1:length(set7data), function(i) {
    npreg(bwYr7k[[i]], data = set7data[[i]][[1]], y.eval = TRUE)
})
```

5. R^2

```
# RF
RsquareNON(set7data, set7RFc, set7RFr)

# Kernel
RsquareNON(set7data, modelYc.np7k, modelYr.np7k)
```

References

[1] Artstein, Z., Vitale, R.A. (1975). A strong law of large numbers for random compact sets. *Annals of Probability*, 5, 879–882

[2] Aumann, R.J. (1965). Integrals of set-valued functions. *J. Math. Anal. Appl.*, 12, 1–12.

[3] Billard, L., Diday, E. (2000). Regression analysis for interval-valued data. In *Data Analysis, Classification and Related Methods*, Proceedings of the Seventh Conference of the International Federation of Classification Societies (IFCS'00) (pp. 369–374). Springer, Belgium.

[4] Billard, L., Diday, E. (2002). Symbolic regression analysis. In *Classification, Clustering and Data Analysis*, Proceedings of the Eighth Conference of the International Federation of Classification Societies (IFCS'02) (pp. 281–288). Springer, Poland.

[5] Billard, L. (2007). Dependencies and variation components of symbolic interval-valued data. In *Selected Contributions in Data Analysis and Classification* (pp. 3–12). Springer, Berlin Heidelberg.

[6] Blanco-Fernández, A., Corral, N., González-Rodríguez, G. (2011). Estimation of a flexible simple linear model for interval data based on set arithmetic. *Computational Statistics and Data Analysis*, 55, 2568–2578.

[7] Blanco-Fernández, A., Colubi, A., González-Rodríguez, G. (2012). Confidence sets in a linear regression model for interval data. *Journal of Statistical Planning and Inference*, 142, 1320–1329.

[8] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 26, 2, 123–140.

[9] Breiman, L. (2001). Random forests. *Machine Learning*, 45, 5–32.

[10] Carvalho, F.A.T., Neto, E.A.L., Tenorio, C.P. (2004). A new method to fit a linear regression model for interval-valued data. *Lecture Notes in Computer Sciences*, 3238, 295–306.

[11] Cattaneo, M.E.G.V., Wiencierz, A. (2012). Likelihood-based imprecise regression. *International Journal of Approximate Reasoning*, 53, 1137–1154.

[12] Debreu, G. (1967). Integration of Correspondences. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability II*, Berkeley, California, 351-372.

[13] Diamond, P. (1990). Least squares fitting of compact set-valued data. *J. Math. Anal. Appl.*, 147, 531–544.

[14] Dietterich, T. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 40, 139–157.

[15] Domingues, M.A.O., Souza, R.M.C.R., and Cysneiros, F.J.A. (2010). A robust method for linear regression of symbolic interval data. *Pattern Recognition Letters*, 31, 1991–1996.

[16] Freund, Y. and Schapire, R.E. (1996) Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, 148–164.

[17] Gil, M.A., Lopez, M.T., Lubiano, M.A., Montenegro, M. (2001). Regression and correlation analyses of a linear relation between random intervals. *Test*, 10, 1 183–201.

[18] Gil, M.A., Lubiano, M.A., Montenegro, M., Lopez, M.T. (2002). Least squares fitting of an affine function and strength of association for interval-valued data. *Metrika*, 56, 97–111.

[19] Gil, M.A., González-Rodríguez, G., Colubi, A., and Montenegro, M. (2007). Testing linear independence in linear models with interval-valued data. *Computational Statistics & Data Analysis*, 51, 3002–3015.

[20] González-Rodríguez, G., Blanco, A., Corral, N., and Colubi, A. (2007). Least squares estimation of linear regression models for convex compact random sets. *Advances in Data Analysis and Classification*, 1, 67–81.

[21] Hörmander, H. (1954). Sur la fonction d'appui des ensembles convexes dans un espace localement convexe. *Arkiv för Mat*, 3, 181–186.

[22] Jeon, Y., Ahn, J., and Park, C. (2015). A nonparametric kernel approach to interval-valued data analysis. *Technometrics*, 57, 4, 566–575.

[23] Kendall, D.G. (1974). Foundations of a theory of random sets. In: *Stochastic Geometry*, eds. Harding, E.F. and Kendall, D.G., John Wiley & Sons, New York.

[24] Körner, R., Näther, W. (1998). Linear regression with random fuzzy variables: extended classical estimates, best linear estimates, least squares estimates. *Information Sciences*, 109, 95–118.

[25] Lyashenko, N.N. (1982). Limit theorem for sums of independent compact random subsets of Euclidean space. *Journal of Soviet Mathematics*, 20, 2187–2196.

[26] Lyashenko, N.N. (1983). Statistics of random compacts in Euclidean space. *Journal of Soviet Mathematics*, 21, 76–92.

[27] Liaw, A. and Wiener, M. (2002). Classification and Regression by randomForest. *R News*, 2, 3, 18–22.

[28] Manski, C.F., Tamer, T. (2002). Inference on regressions with interval data on a regressor or outcome. *Econometrica*, 70, 519–546.

[29] Matheron, G. (1975) *Random Sets and Integral Geometry*. John Wiley & Sons, New York.

[30] Molchanov, I. (2005) *Theory of Random Sets*. Springer-Verlag, London.

[31] Lima Neto, E.A., Carvalho, F.A.T. (2008). Centre and range method for fitting a linear regression model to symbolic interval data. *Computational Statistics & Data Analysis*, 52, 1500–1515.

[32] Lima Neto, E.A., Carvalho, F.A.T. (2010). Constrained linear regression models for symbolic interval-valued variables. *Computational Statistics & Data Analysis*, 54, 333–347.

[33] Lima Neto E.A., Cordeiro, G.M. and Carvalho, F.A.T. (2011). Bivariate symbolic regression models for interval-valued variables. *Journal of Statistical Computing and Simulation*, 81, 11, 1727-1744.

[34] Rådström, H. (1952). An embedding theorem for spaces of convex sets. *Proc. Amer. Math. Soc.*, 3, 165–169.

[35] Seber, G.A.F. (1977). Linear regression analysis. John Wiley & Sons, New York.

[36] Sinova, B., Colubi, A., Gil, M.A., and González-Rodŕiguez, G. (2012). Interval arithmetic-based simple linear regression between interval data: discussion and sensitivity analysis on the choice of the metric. *Information Sciences*, 199, 109–124.

[37] Stoyan, D. (1998). Random sets: models and statistics. *International Statistical Review*, 66, 1, 1-27.

[38] Sun, Y. and Li, C. (2014). On linear regression for interval-valued data in $\mathcal{K}_{\mathcal{C}}(\mathbb{R})$. preprint. arXiv: 1401.1831.

[39] Sun, Y. (2016). Linear regression with interval-valued data. *WIREs Computational Statistics*, 8, 54-60.