Noname manuscript No.
(will be inserted by the editor)

Kernel Methods for Software Effort Estimation

Effects of different kernel functions and bandwidths on estimation accuracy

Ekrem Kocaguneli · Tim Menzies · Jacky W. Keung

Received: date / Accepted: date

Abstract Analogy based estimation (ABE) generates an effort estimate for a new software project through an adaptation of similar past projects (a.k.a analogies). A majority of ABE methods follow uniform weighting in adaptation procedure. Non-uniform weighting has also been proposed as an alternative adaptation method, yet has not been thoroughly investigated. In this research we investigate kernel density estimation for non-uniform weighting in the context of ABE. We used different kernel methods to compute non-uniform weights of analogies. In an extensive experimentation of 330 ABE variants, we found that it is vital to select the right number of neighbors for estimation accuracy. However, non-uniform weighting induced by a kernel method could never outperform uniformly weighted ABE. Hence, we speculate that kernel density estimation in ABE is not a very productive research area to explore.

Keywords Effort estimation, data mining, kernel function, bandwidth

1 Introduction

Software effort estimates are reported to be often wrong by a factor of four [7] or even more [20]. The critical results of wrong estimates for a company are obvious: 1) promising projects that would stay within budget may be rejected, 2) accepted projects may over-run their planned resources and worst of all 3) over-running projects may be cancelled thereby wasting the entire effort. Therefore, effort estimation is an active research area [6, 18, 22, 42] that constantly explores more variations with each model being developed or improved. For example, Auer et al. [4] proposed an extensive

E. Kocaguneli and T. Menzies
Lane Department of Computer Science and Electrical Engineering
West Virginia University
Morgantown, WV 26505, USA
E-mail: ekocagun@mix.wvu.edu, tim@menzies.us

Jacky W. Keung
Department of Computing
The Hong Kong Polytechnic University
Kowloon, Hong Kong
E-mail: Jacky.Keung@comp.polyu.edu.hk

search to learn the best weights for different project features in 2006. Menzies et al.'s COSEEKMO tool explored thousands of discretizer combinations, data pre-processors, feature subset selectors, and inductive learners in the same year [31]. In 2007, Baker proposed an exhaustive search of all possible project features, learners and other variables [5]. Pendharkar et. al. used Bayesian Network (BN) for effort estimation and incorporated BN into decision making procedure aginst risks [37]. Mendes and Mosley employed data-driven and hybrid BN models for web effort estimation [28]. Li et. al. investigated the feature weighting as well as instance selection in analogy based estimation (ABE) domain to address the memory and computation costs in their 2009 study [25]. All these work contributed narrowing down the possible space we need to discover to really understand software effort estimation. Future studies will continue to narrow down this space and investigate other variations of software effort estimation methods.

ABE is based on the premise that effort of a future project can be estimated by adapting the effort values of k similar past projects (adapted k projects are also called *analogies*) [19, 25, 30]. Among proposed adaptation methods we can name choosing closest analogy [8,14], taking mean or median of k analogies [30,45]. In both mean and median approach the influence of analogies are equal, in other words, the low ranked analogies have just as much influence as the high ranked analogies. Equal influence of analogies is an example of uniform weighting ABE methods (from now on U-ABE). Another method for weighting is to let different analogies have different influence on the final estimate. For example Mendes et. al. proposes a method called inverse rank weighted mean (IRWM) that allows higher ranked analogies to have greater influence than the lower ranked ones [29, 30]. Such methods that let different analogies have unequal influence are examples of non-uniform weighting ABE methods (from now on N-ABE).

In this research, we contribute narrowing down the search space in software effort estimation by investigating different weighting strategies in ABE. We investigate the impact of non-uniform weighting and use the concept of kernel density estimation [41] in the context of N-ABE. Kernel-based methods are reported to be one of the most popular non-parametric estimators that can uncover structural features in the data [49]. Furthermore, in various different contexts different researchers benefit from kernel density estimation and report successful results [13, 15, 36].

To guide us in this paper, we have identified the following research questions:

RQ1 Is there any evidence that non-uniform weighting improves the performance of ABE?
RQ2 What is the effect of different kernels for non-uniform weighting ABE?
RQ3 What is the effect of different bandwidths when used for non-uniform weighting ABE?
RQ4 How do the characteristics of software effort datasets influence the performance of kernel weighting in non-uniform weighting ABE?
RQ5 Which parameters among kernel, bandwidth and selected analogy number matter the most in terms of estimation accuracy?

In our research, after an extensive study of 330 variants of ABE, it is clear that selecting the right number of neighbors is vital for effort estimation. However, U-ABE methods are *never enhanced by a particular kernel method*. Hence, we do not believe kernel selection to be a productive area of research to explore in software effort

estimation. We therefore advise researchers in this area to explore other aspects of effort estimation.

Although results of N-ABE are negative, these negative results have at least three positive consequences. Firstly, we can assert that there is nothing inherently wrong with intuition-based weighting schemes like IRWM (since all the weighting schemes we explored had similar results). Secondly, we can better focus future research. Our reading of the results presented in this paper is that in smaller data sets, nuanced explorations of some neighborhood is less useful than the size of that neighborhood. In other words, the value of k appears to be a more important factor in ABE rather than non-uniform weighting strategies. Lastly, unlike studies in other domains that claim "kernel does not matter but the bandwidth does", we cannot offer supportive evidence for a statistical heuristic that leads to the same conclusion. Our results point to the conclusion that neither kernel nor the bandwidth has a considerable impact for improvement in software effort estimation. The benefit of our last result is a reduction in search space for future explorations.

The rest of the paper is organized as follows: In Section 2 we provide background information regarding software effort estimation in general as well as ABE and kernel density estimation. We continue with Section 3, in which we provide the details of the methodology we adopted in this research: Weighting strategies, datasets, experimental settings and performance criteria. In Section 5 we provide a brief discussion of our pre-experimental intuition of N-ABE over some intuitive examples. Then in Section 4 we give the results of our research and continue with Section 6, where we summarize the possible threats the validity of our results. We discuss the conclusions of our research in Section 7 and present our answers to the research questions we followed. Finally in Section 8 we list some of the likely future directions of this research and conclude.

2 Background

In this section, we will provide general background information about software effort estimation and ABE. We will also address how kernel methods have been utilized in the literature and discuss how they can be adapted to software effort estimation domain as a weighting strategy.

2.1 Software Effort Estimation

We can divide software effort estimation into at least two groups [42]: *expert judgment* and *model-based* techniques. *Expert judgment* methods are widely used in software effort estimation practices [16]. Expert judgment can be applied either explicitly (following a method like Delphi [6]) or implicitly (informal meetings among experts). Regardless of the method expert judgment is applied, it is prone to some pitfalls. One possible pitfall in expert-based methods is the fact that they are open to clashes of interest. For instance a faulty estimation of a senior expert may be taken over the more accurate estimation made by a junior expert. Another pitfall is that expert-based methods can be as good as your experts. Jorgensen et al. report that human experts are very poor at evaluating and improving their estimation skills [17].

Unlike expert-based methods, *model-based techniques* do not rely heavily on human judgment. Model based techniques are products of:

1) Algorithmic and parametric approaches or
2) Induced prediction systems.

The first approach is in simplest terms the adaptation of an expert-proposed model to local data. A widely known example to such an approach is Boehm's COCOMO method [7]. The second approach is particularly useful in the case where local data does not conform to the specifications of the expert's method. A few examples of induced prediction systems are linear regression, neural nets, model trees and analogy based estimation [31, 44]. Regardless of the categorization of models, they are all built on inherent assumptions. For example, linear regression assumes that the effort data fits a straight line while model trees assume that the data fits a set of straight lines. In the cases where data violates these assumptions, patches are applied, e.g. take the logarithm of exponential distributions before linear regression [7,21]. However, choosing the appropriate patch again requires qualified experts.

2.2 ABE

Analogy based estimation (ABE) or estimation by analogy (EBA) is a form of case based reasoning (CBR). According to the taxonomy presented in Section 2.1 ABE is grouped together with induced prediction systems. In their 2005 study Myrtveit et. al. follow a different categorization than the one presented in this paper [35]. They group estimation models into sparse-data and many-data categories. Sparse-data methods are defined to be estimation methods that need few or no historical data. Examples to sparse-data methods are Analytical Hierarcy Process (AHP) [43], expert judgment and case-based reasoning. Many-data methods are identified in the form of a function and are subdivided into: 1) functions, 2) Arbitrary function approximators (AFA). The functions may be in the form of $y = Ax^B$, where a mathematical relationship exists between the variables of the expression (e.g. linear regression models). Unlike functions, AFA make no assumption between predictor and response variables. EBA, classification and regression trees (CART) and artificial neural networks (ANN) belong to this class [35].

According to the taxonomy presented by Myrtveit et. al. CBR may belong to both sparse-data or many data category [35]. If one uses CBR to reason from and already selected case then it is identified to be a single-data method. However, if CBR is used to identify the closest case, then it is categorized as a many-data method. ABE is an example of this use of CBR [35].

ABE in the simplest terms, generates its estimate for a test project by gathering evidence from the effort values of similar past projects in some training set. When we analyze the previous research of experts on the domain of ABE such as Shepperd et. al. [46], Mendes et. al. [30] and Li et. al. [25], we can see a baseline technique lying under all ABE methodologies. The baseline technique is composed of the following steps:

- Form a table whose rows are completed past projects (this forms the training set).
- The columns of this table are composed of *independent* variables (the features that define projects) and a *dependent* variable (the recorded effort value).
- Decide on the number of similar projects (*analogies*) to use from the training set when examining a new test instance , i.e. decide on the k-value.
- For each test instance, select those k analogies out of the training set.

  - While selecting analogies, use a similarity measure (for example the Euclidean distance).
  - Before calculating similarity, apply a scaling measure on independent features to equalize their influence on this similarity measure.
  - Use a feature weighting scheme to reduce the effect of less informative features.
- Adapt the effort values of the k nearest analogies to come up with an effort estimate.

Following the steps of this baseline technique, we will define a framework called ABE0. ABE0 uses the Euclidean distance as a similarity measure, whose formula is given in Equation 1.

$$Distance = \sqrt{\sum_{i=1}^{n} w_i(x_i - y_i)^2} \tag{1}$$

In Equation 1, w_i corresponds to feature weights applied to independent features. ABE0 framework does not favor any features over the others, therefore each feature has equal importance in ABE0, i.e. $w_i = 1$.

Following the selection of projects in a CBR system, the next step is deciding on how to adapt them. There is a wide variety of adaptation strategies in the literature [27]. Using effort value of the nearest neighbor [8], taking mean [34] or median [3] of closest analogies, inverse distance and inverse rank weighted mean of closest analogies are among the commonly used adaptation methods in CBR literature [27]. The adaptation of effort suggested by baseline approach does not have to be a a complex process. ABE0 simply returns the median effort values of the k nearest analogies. Since ABE0 implicitly assigns equal weights to k nearest analogies, it turns out to be an U-ABE method.

Angelis et. al. suggests that as the number of the closest projects increase, median is a robust solution [3]. They have found that taking median instead of mean decreases the estimation error. The reason why we chose ABE0 framework to use median instead of mean in our research is due to the fact that we also make use of high as well as low k values and using mean could have let extreme effort values have a strong influence on the estimation. However, we want the estimates of ABE0 framework to represent the majority of selected instances and not greatly affected by extreme values. Therefore, ABE0 uses median instead of mean.

In this research we will compare the results of ABE0 framework with different non-uniform weighting strategies, i.e. with different N-ABE methods. N-ABE methods have been previously adressed in literature. For example inverse rank weighted mean (IRWM) was proposed by Mendes et. al. [30]. IRWM method enables higher ranked analogies to have greater influence than the lower ones. Assuming that we have *3* analogies, the closest analogy (CA) gets a weight of 3, the second closest (SC) gets a weight of 2 and the weight assigned to the last analogy (LA) is 1. With this weighting approach, IRWM would calculate the estimation as in Equaiton 2. Note that we can generalize IRWM to handle more than 3 neighbors as follows: In the case of n closest analogies, the closest neighbor would have the weight of n, the next one would have the weight of $n-1$ and so on. The weighted sum would then be divided by the sum of all weights: $\sum_{i=1}^{n} i$

$$Effort = (3 * CA + 2 * SA + 1 * LA)/(3 + 2 + 1) \tag{2}$$

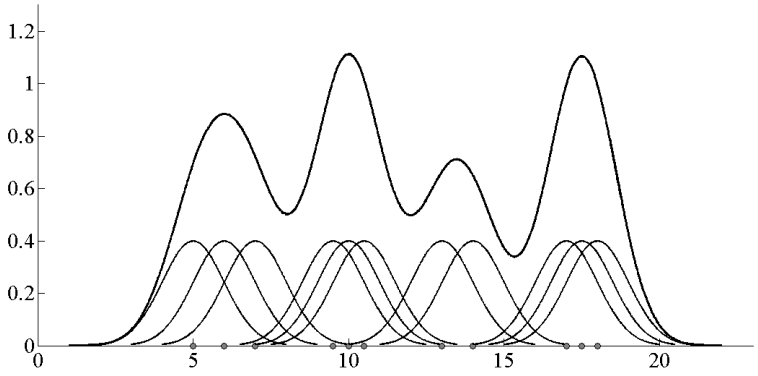


Fig. 1 We see a Gaussian kernel density estimate built on individual data points. Each point is in the center of a kernel and its effect is distributed to its neighborhood. The sum of all kernels make up the final Gaussian kernel density estimate.

IRWM has its root in expert judgment. In other words, in the lack of valuable experts, such a weighting strategy would be almost impossible to apply to the needs of a particular dataset.

2.3 Kernel Density Estimation

IRWM is one example of a broad class of statistical reasoning called kernel density estimation, where IRWM acts like a triangular kernel assigning weights to analogies on the basis of their distance. Kernel density estimation is a non-parametric estimation method that is used to uncover the underlying structures of data, which a parametric approach may fail to reveal [49]. Since we used the univariate kernel density estimation, we will suffice to mention the univariate case in this paper. However, the same approach can be easily adapted to higher dimensionalities [41, 49].

The kernel function is usually chosen to be unimodal and symmetric about zero [49]. A probability distritibution function can be chosen as the kernel function (for instance Gaussian kernel). In a kernel estimation method, the center of the kernel is placed right on each data point and the influence of each data point is distributed to the overall neighborhood. To reach the final density function, contributions coming from each data point are summed up. For example, note how individual Gaussian curves add up to generate the final density estimate in Figure 1 (one kernel is added for each observation along the x-axis [40]).

Kernel density estimation has been successfully used for different type of datasets. For instance Palpanas et. al. use kernel density estimation to address the problem of deviation detection in environment of sensor networks [36]. Frank et. al. use kernel estimation for locally weighting the attributes of Naive Bayes [13]. Furthermore John et. al. use kernel estimation to tackle the normality assumption regarding continuous datasets [15]. They replace single Gaussian distribution that is used to model continuous data with non-parametric kernel density estimation and they report considerable improvements on real and artificial datasets. Although kernel density estimation is used

in different areas for modeling different types of data, to the best of our knowledge it was not previously used in the context of ABE. In this research we propose using kernel density estimation as a N-ABE method for assigning weights to selected analogies.

The kernels we use in our research are: Uniform, triangular, Epanechnikov and Gaussian. We can use a generic formula for some kernels, which is given in Equation 3, where $\mathbf{1}_{(|x|<1)}$ is the indicator function. Furthermore, Equation 4 and Equation 5 explain for the calculation of other functions in Equation 3. Depending on the value of p in Equation 3, we can derive different kernels. For example for $p = 0$ we elicit the uniform kernel, for $p = 1$ we elicit Epanechnikov kernel etc.

$$K(x,p) = \frac{\left(1-x^2\right)^p}{2^{2p+1}B\left(p+1,p+1\right)}\mathbf{1}_{(|x|<1)} \tag{3}$$

$$B\left(p+1,p+1\right) = \frac{\Gamma\left(a\right)\Gamma\left(b\right)}{\Gamma\left(a+b\right)} \tag{4}$$

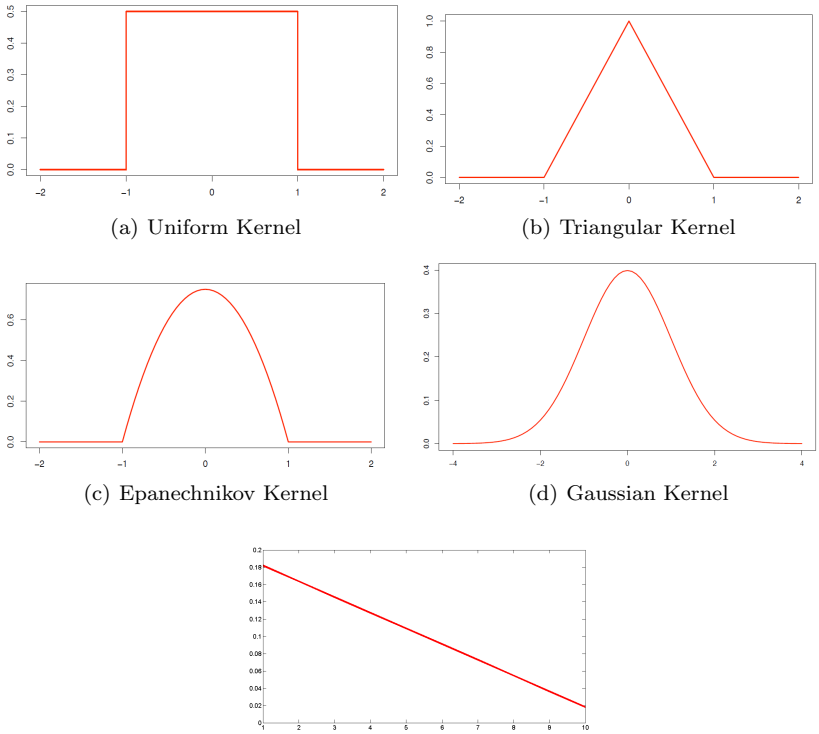
$$\Gamma\left(n\right) = (n-1)! \tag{5}$$

This paper explores the kernels of Figure 2 as well as IRWM [29, 30]. The general shapes of these kernels are given in Figure 3 [40]. IRWM is not actually proposed as a kernel method and it does not fully conform to the definition of standard kernel methods (being unimodal and symmetric about zero and so on). However, due to the weighting strategy it proposes we can read it as an expert proposed kernel, whose shape would look like the right part of a triangular kernel as in Figure 3(e).

Initially, we planned to explore many more kernels than those listed above. However, to our surprise, we found that estimation accuration was not influenced by the kernel method. A literature review revealed that a similar effect has been reported in domains other than effort estimation. A standard result is that the selection of bandwidth (h) for kernels is more influential than the kernel types [10, 41]. Bandwidth h is fundamentally a scaling factor that controls how wide probability density function will spread around a point and the choice of h determines how smooth or rough density estimation will be, i.e. appropriate choice of h is critical to avoid under and over-smoothing [40, 49]. To avoid both under and over-smoothing conditions we used various bandwidth values

Kernel Type	**Formula**
Uniform Kernel	$K(\rho) = \frac{1}{2}\mathbf{1}_{(\lvert\rho\rvert<1)}$
Triangular Kernel	$K(\rho) = (1-\lvert\rho\rvert)\,\mathbf{1}_{(\lvert\rho\rvert<1)}$
Epanechnikov Kernel	$K(\rho) = \frac{3}{4}\left(1-\rho^2\right)\mathbf{1}_{(\lvert\rho\rvert<1)}$
Gaussian Kernel	$K(\rho) = \frac{1}{\sqrt{2\pi}}e^{\frac{-1}{2}\rho^2)}$

Fig. 2 The formulas for different kernels used in this study. In formulas $\rho = \frac{x-X_i}{h}$. Note that IRWM kernel has different characteristics and its calculation details were provided in Section 2.2.



(a) Uniform Kernel

(b) Triangular Kernel

(c) Epanechnikov Kernel

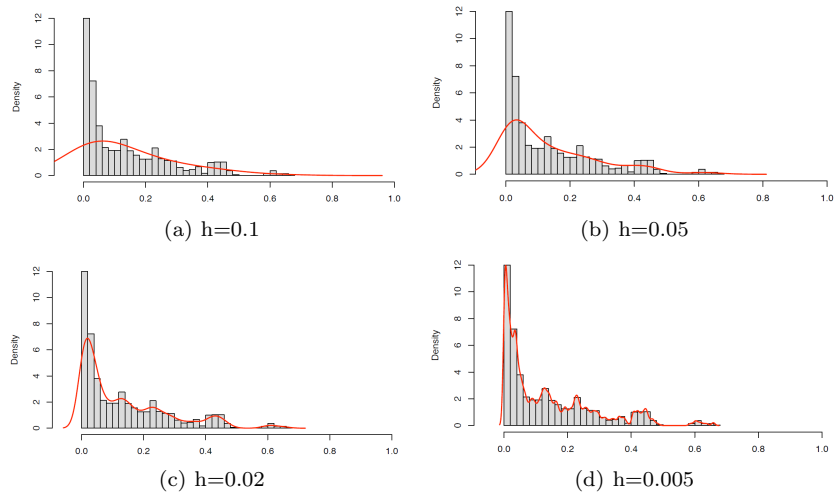
(d) Gaussian Kernel

(e) IRWM Kernel for k=10

Fig. 3 General shapes of the kernel types used in this research. Note that IRWM is not symmetric unlike other kernels, that is due to the fact IRWM has a different weighting procedure than other kernels (see Equation 2).

in our research. One of the bandwidths we used is suggested by John et. al., which is $h = 1/\sqrt{n}$ where h is the bandwidth and n is the size of dataset [15]. The other bandwidth values we used are: 2, 4, 8 and 16.

We can see how choosing different bandwidth values affect kernel density estimation in Figure 4. Figure 4 from [40] is suggestive, but not convincing evidence that effort estimation may not be improved by the selection of the kernel method. Prior work (reporting that bandwidth h values were more important than choice of kernel) comes from (e.g.) the signal processing literature [10,41]. Data sets in those domains are very different to the data sets seen in effort estimation: thousands to billions of instances (for signal processing) versus dozens to hundreds of instances (for effort estimation). In data-started domains like effort estimation, it may be the case that an intelligent selection of the kernel could compensate for data scarcity. Note that other authors in the field of effort estimation have also shared this opinion [29, 30]. However, we are unaware of any effort estimation publications that explore this issue in a systematic manner. Hence, the rest of this paper.



(a) h=0.1 (b) h=0.05

(c) h=0.02 (d) h=0.005

Fig. 4 We see the effect of bandwidth on kernel density estimation. From Figure 4(a) to Figure 4(d), the bandwidth gets smaller and smaller and we observe a transition from over-smoothing to under-smoothing.

3 Methodology

In this section we provide the methodology that we adopted in our research. We discuss how we use kernel density estimation as a non-uniform weighting method as well as which kernels we use for this purpose. Furthermore, we provide information regarding the datasets we used in this research and discuss their characteristics. Also we provide information regarding the experimental settings we adopted. Finally we discuss the performance criteria according to which we compare the performance of N-ABE methods to basic U-ABE method ABE0.

3.1 Weighting Method

Here we summarize how kernel density estimation is employed as a non-uniform weighting method in this research. Assume that our dataset is divided into two sets: $A = \{x_1, ..., x_k\}$ (selected **A**nologies) and $R = \{t_1, ..., t_{n-k}\}$ (**R**est of the dataset). We build the kernel density estimation on R and evaluate the resulting function at instances of A. Equation 6 shows the probability calculation with kernel density estimation. In Equation 6 the kernel K is built on training data $t_i \in R$ and is evaluated at k^{th} analogy x_k for a bandwidth of h.

$$f(x_k, h) = \frac{1}{nh} \sum_{t_i \in R} K\left(\frac{x_k - t_i}{h}\right) \tag{6}$$

The general idea of this approach is that selected k analogies for a test instance come from a distribution and this distribution is specific to the dataset. Furthermore, according to this specific distribution we get different probability values for each analogy. In other words, we have different $f(x_k, h)$ values for each analogy. We use these

probability values as weights for analogies. Note that before using a probability value as a weight, we need to scale it to 0-1 interval according to Equation 7, where x_k represents all the analogies in A except x_i.

$$weight_{x_i} = \frac{f(x_i, h) - max(f(x_k, h))}{max(f(x_k, h)) - min(f(x_k, h))} \tag{7}$$

After calculating $weight_{x_i}$ for each analogy, we update their actual effort values according to their weights. Updating the actual effort values simply means multiplying an effort value with its weight. Equation 8 shows how actual effort value of an analogy, x_i, is updated.

$$updatedEffort_{x_i} = actualEffort_{x_i} * weight_{x_i} \tag{8}$$

3.1.1 Uniform vs. Non-Uniform Weighting

In this research we compare two different weighting strategies in ABE: Uniform weighting and non-uniform weighting. For the ease of naming we use abbreviations to refer to these strategies. **U**niform weighting ABE is abbreviated by U-ABE, whereas **N**on-unform weighting ABE is abbreviated by N-ABE.

For both strategies we can come up with a number of different methods. As for U-ABE, we defined a base method that we call ABE0 (see Section 2.2). ABE0 assumes that each one of the k analogies have equal importance: It assigns equal weights to all analogies. Therefore, ABE0 uses the actual effort values without applying any change during adaptation for the final estimate. In terms of N-ABE methods, we use different kernel types in the context of kernel density estimation. Depending on the underlying characteristics of a particular dataset, kernel density estimation assigns certain probability values to analogies. These probability values define the importance of each analogy: Higher probability means higher weight. Different kernel types used as N-ABE methods in our research are: Uniform kernel, triangular kernel, Epanechnikov kernel, Gaussian kernel and IRWM.

The fundamental difference between N-ABE methods and ABE0 is that in ABE0 analogies are given uniform weights and their actual effort values are used in an *as is* manner, whereas in N-ABE methods analogies are assigned different weights and their actual effort values are multiplied by these weight values.

One point that needs further clarification is the use of uniform kernel as a N-ABE method. Since uniform kernel also follows a uniform weighting scheme, it may be confused with ABE0. However, the uniform weighting scheme of ABE0 and that of uniform kernel are quite different from one another. In uniform kernel, depending on how we select bandwidth some of the instances get uniform and non-zero probability values, whereas some others get a probability value of zero. Therefore, uniform kernel in fact turns out to be a N-ABE method.

Figure 5 succinctly illustrates the difference between uniform kernel being a N-ABE method and ABE0 being a U-ABE method. In Figure 5 x-axis shows the instance IDs, whereas y-axis shows probabilities of instances simulated by a normal probability distribution function (PDF). As we can see from Figure 5, ABE0 would assume equal importance of all instances and hence assign equal probabilities. On the other hand, a uniform kernel assigns equal probabilities to only a certain portion of the instances.

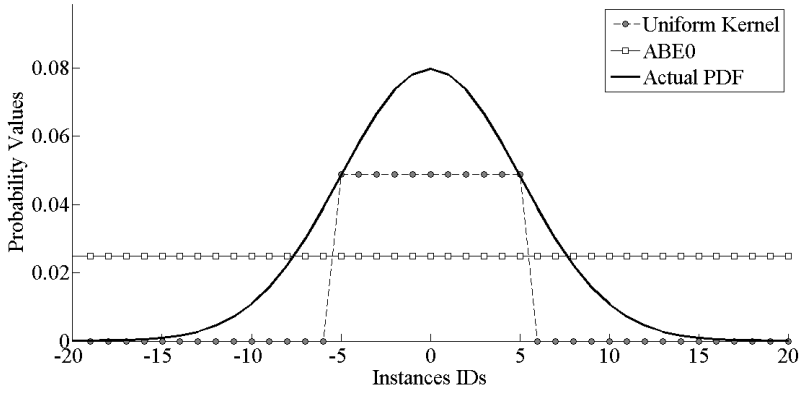


Fig. 5 In the case of ABE0 all instances are given equal probability values, hence equal weights. However, uniform kernel prefers some instances over the others: Only a certain portion of the instances are given equal weights.

3.2 Data

In our research, we have used three commonly used datasets in software effort estimation research: Nasa93, the original Cocomo81 [7], and Desharnais [11]. Cocomo81 and Nasa93 datasets contain projects developed in NASA, whereas Desharnais dataset contains projects developed by Canadian software houses. A number of studies using these datasets subject them to pre-processing steps such as feature and instance selection. Since observing the impact of instance and/or feature selection is not one of our aims, in our research we did not resort to these pre-processing steps and used all features as well as all the instances in the datasets. Furthermore, studies making use of Desharnais dataset exclude 4 of the project instances due to the fact that some of the project properties are missing. Yet, the datasets are already sparsely populated and there are ways to handle missing entries. Simple mean imputation is one of these methods, in which missing properties are replaced with the mean values of other projects for the same property [2]. To handle instances with missing entries in Desharnais dataset, we used simple mean imputation.

Apart from selecting commonly used datasets, we took the quality of the datasets into consideration. In order to evaluate the goodness of datasets, Kitchenham and Mendes propose a quality scoring that consists of four values: poor (less than ten projects), fair (between ten to twenty projects), good (between twenty to forty projects) and excellent (more than forty projects) [23]. Following this quality criteria all the datasets we use in our research rank as excellent quality. The details regarding these datasets can be found in Figure 6.

Dataset	**Features**	$T = \|Projects\|$	**Content**	**Units**
Cocomo81	17	63	NASA projects	months
Nasa93	17	93	NASA projects	months
Desharnais	12	81	Canadian software projects	hours
		Total: 237		

Fig. 6 We used 237 projects coming from 3 datasets. Datasets have different characteristics in terms of the number of attributes as well as the measures of these attributes.

3.3 Experiments

Our experimental settings aim at comparing the performance of standart U-ABE (ABE0) to that of N-ABE (non-uniform weighting through different kernel types). To separate train and test sets we use leave-one-out method, which entails selecting 1 instance out of a dataset of size n as the test set and using the remaining $n-1$ instances as the training set. For each test instance, we run ABE0 and store the estimated effort for that test instance. Then we run different N-ABE methods for the same test instance and store their estimates as well. As the analogy number is reported to play a critical role in estimation accuracy [19], both for U-ABE and N-ABE methods, we tried different numbers of analogies. Furthermore, to hinder any particular bias that would come from the settings of a single experiment, we repeated the afore mentioned procedure 20 times.

In this research we use 2 forms of ABE methods (uniform and non-uniform weighting) induced on 3 datasets (Cocomo81, Nasa93 and Desharnais) for 5 different numbers of analogies, a.k.a k values. The k values we used in our research are: $k \in \{1, 3, 5, 7, 9, dynamicK\}$. $dynamicK$ is selected for each individual test instance through a process, in which we randomly pick up 10 instances from the training set as the validation set and select the lowest error yielding k value as the $dynamicK$. Furthermore, we use 5 different kernels (Uniform, triangular, Epanechnikov, Gaussian and IRWM) with 5 bandwidth values in N-ABE experiments. Therefore, to further explore field of software effort estimation, we investigate a total of 330 different settings in this research:

- U-ABE Experiments: 15 settings
  - *3 datasets * 5 k values = 15*
- N-ABE Experiments: 315 settings
  - Standard Kernels: *3 datasets * 5 k values * 4 kernels * 5 bandwidths = 300*
  - IRWM: *3 datasets * 5 k values = 15*

3.4 Performance Criteria

To observe the effect of non-uniform weighting in ABE, we use the following performance measures: the magnitude of relative error (MRE), median magnitude of relative error (MdMRE), prediction within 25% error rate (Pred(25)) and win-tie-loss values generated by a statistical test (Mann-Whitney U Test). MRE is used by the authors because it is the most commonly used performance criterion for assessing the performance of competing software effort estimation methods [8, 12, 33]. Furthermore, as we can see from Formula 9, MRE value is a direct measure of the absolute difference between the prediction and actual value [48] and hence it gives a per-instance based performance evaluation.

$$MRE = \frac{|actual_i - predicted_i|}{actual_i} \tag{9}$$

MdMRE has emerged as one of the de facto standard evaluation criterion for cost estimation models [47]. Often software effort studies make use of mean magnitude of relative error (MMRE) as the performance criterion as well. MMRE is the mean of all MRE values. However, the mean approach considers every observation and is sensitive to individual predictions that have high MREs [30]. One way to address this problem

is the median approach via MdMRE. Median also gives information about central tendency, but it is less sensitive to extreme MRE values. Therefore, while we comment on the results of MRE-based measures in Section **??**, we provide MdMRE values. The formulas of MdMRE is given in Equation 10, where n is the test set size.

$$MdMRE = median(MRE_1, MRE_2, ..., MRE_n) \tag{10}$$

Pred(l) is also a very widely used performance criterion in software effort estimation studies [30]. Pred(l) is defined as the prediction at level l, which measures the percentage of estimates that are within l percent of the actual effort values [28]. The suggestion regarding the value of l is that it shall be set at 25 percent [9] and that a good prediction system should offer this accuracy level 75 percent of the time [9, 30]. Therefore, we will also use Pred(l) at 25 25% (i.e. Pred(25)).

$win_i = 0$, $tie_i = 0$, $loss_i = 0$
$win_j = 0$, $tie_j = 0$, $loss_j = 0$
if MANN-WHITNEY($MRE's_i$, $MRE's_i$) says they are the same **then**
  $tie_i = tie_i + 1$;
  $tie_j = tie_j + 1$;
else
  if median($MRE's_i$) < median($MRE's_j$) **then**
    $win_i = win_i + 1$
    $loss_j = loss_j + 1$
  else
    $win_j = win_j + 1$
    $loss_i = loss_i + 1$
  end if
end if

Fig. 7 Pseudocode for Win-Tie-Loss Calculation Between Method i and j

Note that, MRE related measures are subject to many pitfalls. If MRE is used a stand-alone performance evaluation criterion (i.e. not combined with appropriate statistical tests), it may lead to biased or even false conclusions [12]. To prevent us from falling into MRE-related pitfalls, we use another performance criterion called win-tie-loss calculation. A win-tie-loss calculation tells that comparison between two methods i and j makes sense only if they are statistically different. If there is no statistically significant difference between two methods, say method i and method j, then it indicates that results are observations coming from the same distribution. Therefore, methods are said to '*tie*" and their *tie* values (tie_i and tie_j) are incremented. However, if there is a statistical difference between two methods, then the method with a lower median MRE score, say i, is said to have a "win" and the one with the lower MRE, say j, is said to have a "lose". The related values win_i and $loss_j$ are incremented by one. The pseudocode for a win-tie-loss calculation is given in Figure 7. For the comparison of methods in win-tie-loss calculation, a non-parametric statistical test (the Mann-Whitney rank-sum test) is used at a significance level of 95%.

4 Results

To see the effect of kernel weighting, rigorous experimentation on 19 datasets and 3 different performance measures was performed. Also, for each performance measure we

tried 4 different kernels subject to 5 different bandwidths, plus the IRWM kernel (which does not have a bandwidth concept) and reported associated $win, tie, loss$ statistics.

Figure 8 is a sample table, where we report the $win, tie, loss$ statistics of Desharnais dataset for ABE0 and N-ABE through Gaussian kernel. For each dataset we have 4 such tables (one for each kernel), so for all datasets there is *19 Datasets* $\times$ *4 tables = 76 tables*. For the IRWM kernel there will be another *19 datasets* $\times$ *1 kernel = 19 tables*. So in total, we have *76 + 19 = 95 tables* to report, which is impractical to put in a summary report such as this one. Therefore, we provide the sample table of Figure 8 as a pointer and manual to other 94 tables, which are given under [1].

In Figure 8 we see that each row reports $win, tie, loss$ statistics of ABE0 methods (*k=3,5,7,9,best*) as well as N-ABE methods (*k=[3,5,7,9,best]+kern where kern stands for kernel weighting*) subject to a particular performance measure. Similarly, each column shows the $win, tie, loss$ statistics associated with a particular bandwidth value. As can be seen in Figure 8, ABE0 methods always have higher win values and always have a $loss$ value of 0, meaning that they never lose against N-ABE methods. By analyzing each row/column intersection we can see that the performance of ABE0 is never improved by N-ABE.

The analysis of Figure 8 is only valid for Desharnais dataset and Gaussian kernel. So as to provide an executive summary of all 19 datasets and all kernel/bandwidth combinations, we will use Figure **??**. Each row of Figure **??** shows the comparison of ABE0 performance to that of N-ABE subject to 3 different performance measures. Every kernel/bandwidth intersection in Figure **??** has 3 symbols corresponding to MdMRE, MAR and Pred(25) comparisons from left to right. Each of these 3 symbols can have 3 values: $-, +, o$. A *"−"* symbol means that N-ABE decreased the accuracy of ABE0, whereas a *"o"* symbol means ABE0 and N-ABE are statistically same in all conditions and a *"+"* symbol shows that ABE0 accuracy was improved through kernel weighting (i.e. N-ABE has a better accuracy than ABE0).

So as to decide if ABE0 performance is improved ($+$), decreased ($-$) or not changed (o) by N-ABE we look at all of the 95 detail-tables such Figure 8 and see the behavior of 5 k values under ABE0 and N-ABE. If the majority of k values (at least 3 out of 5) are improved by N-ABE in terms of $win - loss$ values, then we assign a "+" symbol for that setting; if the performance associated with the majority of k values is decreased by N-ABE, then we assign a "−" symbol; otherwise, we assign a "o" symbol to that setting.

See in Figure **??** that out of 19 datasets, there is only one dataset (SDR) where N-ABE provides a performance improvement in certain cases. Even for that dataset there are 15 "+" symbols and 21 "−" symbols, meaning that most of the time N-ABE is still destructive. In 18 datasets, which is $\frac{18}{19} = 95\%$ of all the datasets, there is not a single case where N-ABE improves the performance of ABE0. Also note that Figure **??** provides an extensive summary of different error measures (MdMRE, MAR, Pred(25)) as well as kernels and bandwidths. Therefore, our conclusion from Figure **??** that *"non-uniform weighting through standard kernel methods does not improve the performance of ABE"* is quite stable accross different datasets and error measures.

Another summary table is given in Figure 11. Figure 11 is very similar to Figure **??** in the sense that it summarizes the performance of N-ABE over 19 datasets w.r.t. three different performance measures. The difference is that Figure **??** summarizes the results of standard kernel methods, whereas in Figure 11 we see the N-ABE performance under an *expert-based* kernel, i.e. IRWM. Although there are important differences between standard and expert-based kernels (IRWM has no bandwidth parameter), the results

		h=1/sqrt(size)			h=2			h=4			h=8			h=16		
		WIN	TIE	LOSS	WIN	TIE	LOSS	WIN	TIE	LOSS	WIN	TIE	LOSS	WIN	TIE	LOSS
MdMRE	k=3	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=5	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=7	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=9	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=best	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=3+kern	80	0	100	80	0	100	80	0	100	80	0	100	80	0	100
	k=5+kern	0	60	120	60	0	120	60	0	120	60	0	120	60	0	120
	k=7+kern	0	60	120	20	20	140	20	20	140	20	20	140	20	20	140
	k=9+kern	0	60	120	0	0	180	0	0	180	0	0	180	0	0	180
	k=best+kern	0	60	120	20	20	140	20	20	140	20	20	140	20	20	140
MAR	k=3	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=5	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=7	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=9	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=best	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=3+kern	0	80	100	60	20	100	60	20	100	60	20	100	60	20	100
	k=5+kern	0	80	100	0	80	100	0	80	100	0	80	100	0	80	100
	k=7+kern	0	80	100	0	60	120	0	60	120	0	60	120	0	60	120
	k=9+kern	0	80	100	0	60	120	0	60	120	0	60	120	0	60	120
	k=best+kern	0	80	100	0	60	120	0	60	120	0	60	120	0	60	120
Pred(25)	k=3	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=5	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=7	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=9	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=best	100	80	0	100	80	0	100	80	0	100	80	0	100	80	0
	k=3+kern	60	0	120	80	0	100	80	0	100	80	0	100	80	0	100
	k=5+kern	20	60	100	60	0	120	60	0	120	60	0	120	60	0	120
	k=7+kern	0	60	120	0	20	160	0	20	160	0	20	160	0	20	160
	k=9+kern	0	60	120	20	0	160	20	0	160	20	0	160	20	0	160
	k=best+kern	0	60	120	20	20	140	20	20	140	20	20	140	20	20	140

Fig. 8 Desharnais dataset $win, tie, loss$ statistics for ABE0 and N-ABE through Gaussian kernel. For each dataset we have 4 of these tables (one for each kernel). In total it amounts to *19 Datasets* $\times$ *4 tables = 76 tables*. It is infeasible to include all the tables in this paper, therefore an executive summary of 76 tables is provided in Figure 9. Furthermore, we provide all 76 tables in excel format at http://goo.gl/qpQiD.

Dataset	Kernel	h=1/sqrt(size)	h = 2	h = 4	h = 8	h = 16
Coc81	**Uniform**	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Triangular	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Epanechnikov	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Gaussian	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
Coc81e	**Uniform**	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Triangular	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Epanechnikov	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Gaussian	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
Coc81o	**Uniform**	*−o−*	*−oo*	*−oo*	*−oo*	*−oo*
	Triangular	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Epanechnikov	*− − −*	*ooo*	*ooo*	*ooo*	*ooo*
	Gaussian	*−o−*	*ooo*	*ooo*	*ooo*	*ooo*
Coc81s	**Uniform**	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Triangular	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Epanechnikov	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
	Gaussian	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
Ns93	**Uniform**	*−o−*	*−o−*	*−o−*	*−o−*	*−o−*
	Triangular	*−o−*	*−o−*	*−o−*	*−o−*	*−o−*
	Epanechnikov	*−o−*	*ooo*	*ooo*	*ooo*	*ooo*
	Gaussian	*−o−*	*ooo*	*ooo*	*ooo*	*ooo*
Ns93c1	**Uniform**	*− − −*	*− − −*	*− − −*	*− − −*	*− − −*
	Triangular	*− − −*	*−o−*	*−o−*	*−o−*	*−o−*
	Epanechnikov	*− − −*	*−o−*	*−o−*	*−o−*	*−o−*
	Gaussian	*−o−*	*−o−*	*−o−*	*−o−*	*−o−*
Ns93c2	**Uniform**	*ooo*	*−o−*	*−o−*	*−o−*	*−o−*
	Triangular	*−o−*	*ooo*	*ooo*	*ooo*	*ooo*
	Epanechnikov	*−o−*	*ooo*	*ooo*	*ooo*	*ooo*
	Gaussian	*ooo*	*ooo*	*ooo*	*ooo*	*ooo*
Ns93c5	**Uniform**	*− − −*	*−o−*	*−o−*	*−o−*	*−o−*
	Triangular	*− − −*	*ooo*	*ooo*	*ooo*	*ooo*
	Epanechnikov	*− − −*	*ooo*	*ooo*	*ooo*	*ooo*
	Gaussian	*− − −*	*ooo*	*ooo*	*ooo*	*ooo*

Fig. 9 Nine data sets comparing ABE0 to N-ABE. For every row in each cell, there are three symbols indicating the effect of N-ABE w.r.t. 3 different error measures. From left to right, the first symbol stands for N-ABE effect w.r.t. MdMRE, the second symbol w.r.t. MAR and the third one w.r.t. Pred(25). A "+" indicates that for majority of k values (at least 3 out of 5 k values), N-ABE improved ABE0 in terms of $win - loss$ values. "−" indicates that N-ABE decreased the performance of ABE0 in the majority case. If the former conditions do not satisfy, then a "*o*" symbol is assigned. Note that dataset order here is the same as Figure 6, yet the dataset names are abbreviated to 3 to 5 letters due to space constraints.

seen in Figure 11 is quite similar to those of Figure **??**. As can be seen in Figure 11, there is not a single case where N-ABE (under IRWM kernel) improves the performance of ABE0. Furthermore, the amount of "−" symbols is much more than "*o*", meaning that N-ABE decreases the performance of ABE0 most of the time.

5 Discussion

Rigorous experimentation presented in this research shows that our results are mostly negative, i.e. different variants of kernel estimation do not increase estimation accuracy. In theory, it seems that the selection of the right kernel could significantly improve effort estimation: For example, an intelligent selection of the kernel might compensate for data scarcity. Other effort estimation researchers have shared this theoretical point of view [29, 30]. However, to the best of our knowledge, the effect of selecting different kernel methods has not been rigorously explored in the SE literature. Our pre-experimental intuition in this research was that non-uniform weighting could end up performing no better and perhaps even worse than simple uniform weighting for

Dataset	Kernel	h=1/sqrt(size)	h = 2	h = 4	h = 8	h = 16
Des	**Uniform**	− − −	− − −	− − −	− − −	− − −
	Triangular	*ooo*	− − −	− − −	− − −	− − −
	Epanechnikov	− − −	− − −	− − −	− − −	− − −
	Gaussian	− − −	− − −	− − −	− − −	− − −
DesL1	**Uniform**	− − −	− − −	− − −	− − −	− − −
	Triangular	*ooo*	−*o*−	−*o*−	−*o*−	−*o*−
	Epanechnikov	− − −	−*o*−	−*o*−	−*o*−	−*o*−
	Gaussian	− − −	−*o*−	−*o*−	−*o*−	−*o*−
DesL2	**Uniform**	− − −	− − −	− − −	− − −	− − −
	Triangular	*ooo*	− − −	− − −	− − −	− − −
	Epanechnikov	− − −	−*o*−	−*o*−	−*o*−	−*o*−
	Gaussian	− − −	−*o*−	−*o*−	−*o*−	−*o*−
DesL3	**Uniform**	−*o*−	−*o*−	−*o*−	−*o*−	−*o*−
	Triangular	−*o*−	*ooo*	*ooo*	*ooo*	*ooo*
	Epanechnikov	−*o*−	*ooo*	*ooo*	*ooo*	*ooo*
	Gaussian	−*o*−	*ooo*	*ooo*	*ooo*	*ooo*
SDR	**Uniform**	−*o*−	−*o*−	−*o*−	−*o*−	−*o*−
	Triangular	+ + −	+*o*−	+*o*−	+*o*−	+*o*−
	Epanechnikov	− + −	+ + −	+ + −	+ + −	+ + −
	Gaussian					
Albr	**Uniform**	− − −	− − −	− − −	− − −	− − −
	Triangular	− − −	−*o*−	−*o*−	−*o*−	−*o*−
	Epanechnikov	− − −	−*o*−	−*o*−	−*o*−	−*o*−
	Gaussian	− − −	−*o*−	−*o*−	−*o*−	−*o*−
Finn	**Uniform**	− − −	− − −	− − −	− − −	− − −
	Triangular	*ooo*	− − −	− − −	− − −	− − −
	Epanechnikov	− − −	−*o*−	−*o*−	−*o*−	−*o*−
	Gaussian	− − −	−*o*−	−*o*−	−*o*−	−*o*−
Kem	**Uniform**	−*o*−	−*o*−	−*o*−	−*o*−	−*o*−
	Triangular	− − −	*ooo*	*ooo*	*ooo*	*ooo*
	Epanechnikov	− − −	*ooo*	*ooo*	*ooo*	*ooo*
	Gaussian	− − −	*ooo*	*ooo*	*ooo*	*ooo*
Maxw	**Uniform**	− − −	− − −	− − −	− − −	− − −
	Triangular	− − −	*ooo*	*ooo*	*ooo*	*ooo*
	Epanechnikov	− − −	*ooo*	*ooo*	*ooo*	*ooo*
	Gaussian	− − −	*ooo*	*ooo*	*ooo*	*ooo*
Miy94	**Uniform**	− − −	− − −	− − −	− − −	− − −
	Triangular	− − −	− − −	− − −	− − −	− − −
	Epanechnikov	− − −	−*o*−	−*o*−	−*o*−	−*o*−
	Gaussian	− − −	−*o*−	−*o*−	−*o*−	−*o*−
Tel	**Uniform**	− − −	− − −	− − −	− − −	− − −
	Triangular	−*o*−	−*o*−	−*o*−	−*o*−	−*o*−
	Epanechnikov	− − −	−*o*−	−*o*−	−*o*−	−*o*−
	Gaussian	− − −	−*o*−	−*o*−	−*o*−	−*o*−

Fig. 10 Ten more data sets comparing ABE0 to N-ABE. Same format as Figure 9.

effort estimation datasets. To demonstrate why we had such a negative opinion, we provide an intuitive example in Figure 12. Assume that for a particular test project, 3 analogies are chosen (P_1, P_2, P_3) with effort values of 10.0, 20.0 and 60.0. Also assume that kernel density estimation assigned the the probabilities of 0.1, 0.3 and 0.5 to these analogies respectively. When we normalize these probability values to a 0-1 interval, the weights assigned to P_1, P_2, P_3 become 0.0, 0.5 and 1.0. As we can see in Figure 12, the estimation (i.e. median of analogies) for uniform weighting case would be 20.0, whereas the estimation for non-uniform weighting case would be 35.0. A shift from an estimate of 20.0 to 35.0 is a dramatic change of 75% rather than a slight correction. Therefore, our intuition was that non-uniform weighting in software effort estimation could be disruptive rather than constructive.

Then the most likely question to be raised is "Why do other fields [13,15,36] benefit from weighting, whereas effort estimation does not?". Our belief is that the answer is partially hidden behind the low sample sizes of effort datasets. Scarcity of the samples

Dataset	Improvement		
	MdMRE	MAR	Pred(25)
Cocomo81	–	*o*	–
Cocomo8e	*o*	*o*	*o*
Cocomo8o	–	–	–
Cocomo8s	*o*	*o*	*o*
Nasa93	–	–	–
Nasa93_center_1	–	–	–
Nasa93_center_2	–	*o*	–
Nasa93_center_5	–	–	–
Desharnais	–	–	–
DesharnaisL1	–	–	–
DesharnaisL2	–	–	–
DesharnaisL3	–	*o*	–
SDR	–	*o*	–
Albrecht	–	–	–
Finnish	–	–	–
Kemerer	–	–	–
Maxwell	–	–	–
Miyazaki94	–	–	–
Telecom	–	–	–

Fig. 11 The comparison of ABE0 to N-ABE under IRWM kernel. Similar to Figure 9 three symbols indicate the effect of N-ABE w.r.t. 3 different error measures and "+" indicates that for majority of k values N-ABE improved ABE0 in terms of $win - loss$ values. A "–" symbol indicates a decrease and a "*o*" symbol indicates neither decrease nor increase. Notice that subject to IRWM kernel, N-ABE fails to improve ABE0 w.r.t. 3 different performance measures.

Analogies	P_1	P_2	P_3
Effort Values	10.0	20.0	60.0
Probabilities	0.1	0.3	0.5
Weights	0.0	0.5	1.0
Uniform Weighting Estimate		20.0	
Non-Uniform Weighting Estimate		35.0	

Fig. 12 An intuitive example. In a 3 analogy case, there is a 75% change for a hypothetical test project between uniform and non-uniform weighting effort estimates.

means that the weighting observes a signal being broadcast from very small number of points in the neighborhood. Although we can tune the size of the neighborhood with the bandwidth value, in low sample sizes kernel estimation still performs poorly.

We can observe the effect of instance size and bandwidth value on the estimation accuracy in Figure 13. In Figure 13 we simulate 50, 100 and 1000 samples coming from two Gaussian probability distribution functions (PDFs): $N(20, 5)$ and $N(35, 5)$. Then we use kernel density estimation technique with a Gaussian kernel to estimate the density at discrete values of x in [0-55] interval with a step-size of 1. In Figure 13 the x-axis shows the instance IDs for discrete x values and instance IDs are equal to actual x values. An x with the ID of say 10 corresponds to the discrete x value of 10. The y-axis in Figure 13 shows the probability values for each x instance. To better observe the probability values, we plot data as logarithmic scale for y-axis while still reporting the actual probability values rather than the logged values. The probability values of x for the actual Normal distributions are shown with a solid line. Ideally we want to get estimates as close as possible to the envelope indicated by the solid line. The density estimates with different bandwidths (different h values) are shown with different lines in Figure 13. Three bandwidth values mean different neighborhood sizes and they enable different number of sample points to fall into neighborhoods. $h = 0.001$ is an example of a too small bandwidht value. As we see, for all sample sizes $h = 0.001$

does hardly let any points to fall into the neighborhood of x, therefore the probability is zero most of the time and whenever point(s) fall into neighborhood of x, we observe sparks of probability. Contrary to 0.001, bandwidth value of 10 is a too large value and it cannot model the underlying Gaussian distributions either. When we look at Figure 13, we see that estimations made with $h = 10$ are over-smoothed and $h = 10$ can only estimate a single Gaussian distribution instead of two. In this simple example $h = 1$ yields the closest estimates to the actual probability values. However, we see that even for $h = 1$, the sample size plays a critical role in estimation. When we look at the fit between $h = 1$ estimates and the actual PDFs in Figure 13, we see that as we increase the size of simulated points the large deviations that we observe in Figure 13(a) diminishes through Figure 13(c).

In case of signal processing the sample sizes are closer to Figure 13(c) and as we see from the simulation example, kernel estimates can successfully model such densely populated datasets. In this simulation example we chose the instance sizes of 50 and 100 on purpose, because, software effort datasets used in our research fall into the range of these numbers. Hence Figure 13(a) and Figure 13(b) can give us hints about the kernel density estimation applied on software effort estimates. When we observe behaviour of kernel estimates for low sample sizes in Figure 13(a) and Figure 13(b), it is somewhat expected to observe lower performance values in sparsely populated datasets like software effort datasets.

Although simulation study also confirms our intuitive feeling, data coming from simulation can hardly model the real-life data in software effort estimation domain and we wanted to observe whether results of intuitive examples would also hold for real-life datasets.

6 Threats to Validity

We will address the threats to validity of this research under 3 categories: Internal validity, external validity and construct validity. Before addressing our research in terms of these categories of threats to validity, we would like to give their conscise definitions.

- *Internal validity* asks to what extent the cause-effect relationship between dependent and independent variables holds [2].
- *External validity* questions the ability to generalize the results [32].
- *Construct validity* (i.e. face validity) makes sure that we in fact measure what we intend to measure [39].

The perfect case for the satisfaction of internal validity would be the application of a theory that was learned from past experiences to new situations. However, data in software effort estimation domain is a relatively sparse resource and most of the studies make use of commonly-explored datasets like the ones we use in this research. Therefore, the issue of internal validity thereatens all effort studies that use past data. However, we can mitigate this threat by simulating the behavior of a learned theory in new settings. In our study, we utilize leave-one-out method for all treatments to address such internal validity issues. Leave-one-out selection enables us to separate the training and test sets completely in each experiment, thereby making the test sets completely new situations for the training sets.

To observe the generalizability of our results, we perform extensive experiments on 3 datasets. The datasets are widely used in software effort estimation community and have very diffferent characteristics in terms of various criteria such as size, number of features, types of features and measurement method. Furthermore datasets are subject to rigorous experimentation where we investigate the effects of non-uniform weighting on performance under 330 settings. Our observations for all the settings are extremely similar. Therefore, for the datasets used in our research, our opinion is that the results have external validity. However, to have full confidence in our claims when saying that N-ABE methods fail to outperform ABE0, our study needs to be replicated on other dataset and possibly with different non-uniform weighting strategies.

The choice of performance measures is an open issue in software effort estimation domain. For example MdMRE is recognized as de facto evaluation criterion for cost estimation models [47] and it appears as a practical performance evaluation option to a number of researchers [24, 26, 38]. On the other hand, use MdMRE is still criticized for being unreliable [12, 35]. Foss et. al. for instance shows that MRE can be misleading, if used as the only performance criterion [12]. Therefore, a study willing to have construct validity should not merely rely on MRE-based measures. To measure what we really intend to measure, we make use of win-tie-loss measures apart from MdMRE and Pred(25). Of course all these criteria have their inherent weaknesses and strenghts. Our aim in combining these measures is to use them in a complementary manner. For example strenght of MdMRE as well as Pred(25) is that they give a general picture of per instance-based evaluation. However, median error measure and Pred(25) become too general when averaged over 20 runs and we do not have much knowledge regarding individual runs. Win-tie-loss measures on the other hand compare each method with one-another for each run and allow us to have a better opinion concerning individual runs. Furthermore, MdMRE and Pred(25) show us the difference between two methods based on MRE. But this difference may not always have statistical significance. To ensure the statistical validity of our results we make use of Mann-Whitney U test at a significance level of 95% in win-tie-loss calculations. Therefore, our use of different performance measures such as MdMRE, Pred(25) as well as win-tie-loss, provides different perspectives of the results and lets us know if the results have statistical significance.

7 Conclusions

In this research we tried kernel density estimation as a non-uniform weighting strategy for ABE. We conducted extensive experiments with various kernels and observed the performance variations between U-ABE and N-ABE methods. For the datasets used in our research (Cocomo81, Nasa93 and Desharnais) there was not a single case where N-ABE methods outperformed ABE0. Unlike previous studies in different domains that use kernel density estimation and report improved accuracy values [15,36], we did not observe such an effect on software effort datasets. The reason for different results between previous research and our research may lie in the different characteristics of the datasets used in different studies. For instance none of the previous studies we searched through uses software effort datasets and the used datasets are much more densely populated than software effort datasets.

The literature is doubtful about the ability to make model rankings as offered in this paper between ABE0 and N-ABE [12, 18, 35]. For example Foss et. al. comments that "...is futile to search for the Holy Grail: A single, simple-to-use, universal goodness-of-

fit kind of metric, which can be applied with ease to compare (different methods)" [12]. The evidence offered in literature that such rankings may change is based on:

- the random selection of data,
- the dataset used,
- the evaluation metric used for comparison.

However, a convincing counter example to this scenario would be to show that method A is better than method B over multiple datasets, over multiple random selections of train/test data and over multiple evaluation criteria. In this research:

- we used 3 of the most commonly utilized public datasets,
- we adopted leave-one-out method in which every instance is considered as a test instance once,
- covered 330 different settings,
- evaluated our results w.r.t. to 3 different evaluation criteria that have different inherent assumptions.

Therefore, we believe our results point to the conclusion that in terms of estimation accuracy it is possible to rank ABE0 better than kernel weighted N-ABE methods in software effort estimation.

7.1 Answers To Research Questions

In this section we map the evaluation of our results to particular research questions that guided us in this research.

RQ1 Is there any evidence that non-uniform weighting improves the performance of ABE?
In our experiments we do not see a single case in which non-uniform weighting improved ABE. On the contrary, for all settings ABE0 yields much better results than N-ABE methods. Therefore, the evidence suggests that non-uniform weighting decreases estimation accuracy in ABE systems.

RQ2 What is the effect of different kernels for non-uniform weighting ABE?
We observe inconsistent and extremely limited effect due to change of kernels. There are only slight variations in performance when different kernels are used. However, these variations do not follow a definite pattern and they are far from being considerable.

RQ3 What is the effect of different bandwidths when used for non-uniform weighting ABE?
Change of bandwidths shows a very limited and random effect on the accuracy values too. Therefore, we cannot say that applying different bandwidths has a certain effect on N-ABE performance.

RQ4 How do the characteristics of software effort datasets influence the performance of kernel weighting in non-uniform weighting ABE?
In this paper we reported different results than the previous studies that used kernel estimation on different data types. This fact can be attributed to the particular characteristics of software effort datasets. Effort datasets are much smaller than most of the datasets in different domains. The dependent variable (effort value of a completed project) is highly variable. Furthermore, the attribute values are very open to personal judgment and error. All these factors suggest that non-parametric methods may be failing due to inherent characteristics of software effort data.

RQ5 Which parameters among kernel, bandwidth and selected analogy number matter the most in terms of estimation accuracy?
Our results showed that in a N-ABE model with kernel density estimation neither kernel nor the bandwidth matters. However, the selected number of analogies (k) matters greatly. Unlike change of kernel and bandwidth that did not provide any noticable change in terms of accuracy values, k value influenced performance measures considerably and we always observed higher accuracies for lower k values.

8 Future Work

We can identify 3 main domains in which this study may be extended:

1. *Dataset:* In this research we used 3 common software effort datasets. However, this study may be replicated on other software effort datasets as well. Furthermore, ABE is not restricted to software effort estimation domain. Kernel density estimation may be experimented on other ABE-applicable datasets as well.
2. *Weighting Strategy:* Strategy for non-uniform weighting in a N-ABE method may be completely different than kernel density estimation. Another future direction can be experimentation on different non-uniform weighting strategies that are preferably based on different assumptions.
3. *Kernel Type:* We used 5 kernels (including IRWM) as non-uniform weighting strategies in our research. But the ones used in this research are obviously not the only kernel types. It may be the case that other particular kernels will perform differently than the ones used here. Therefore, one future direction to this research would be the investigation of different kernels for better performance.

The experiments shown in this research took three months to research, design, execute, then write up. It turns out that we could have spent the time more productively on other issues. Our pre-experimental intuition that "non-uniform weighting in the data sparse domain of software effort estimation could be disruptive rather than constructive" turned out to be correct. We suggest researchers willing to follow future directions to bear these facts in mind as well.

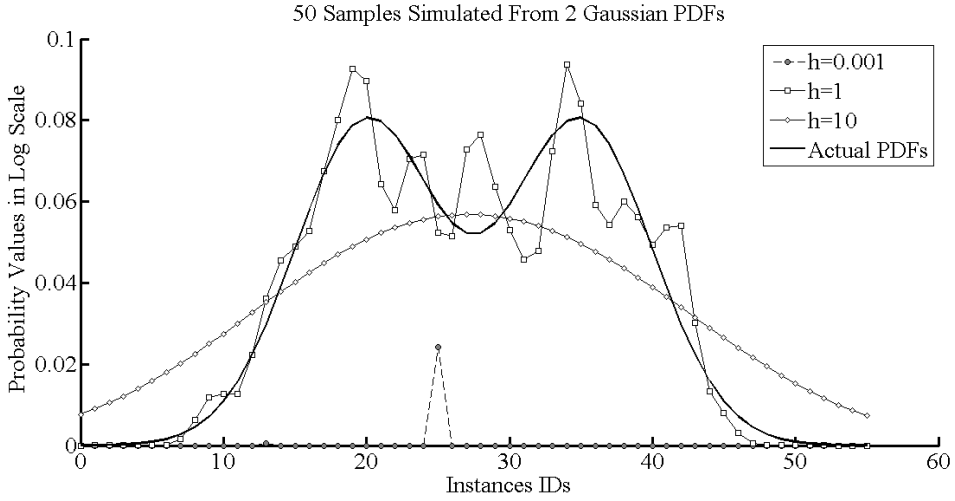
9 Acknowledgements

References

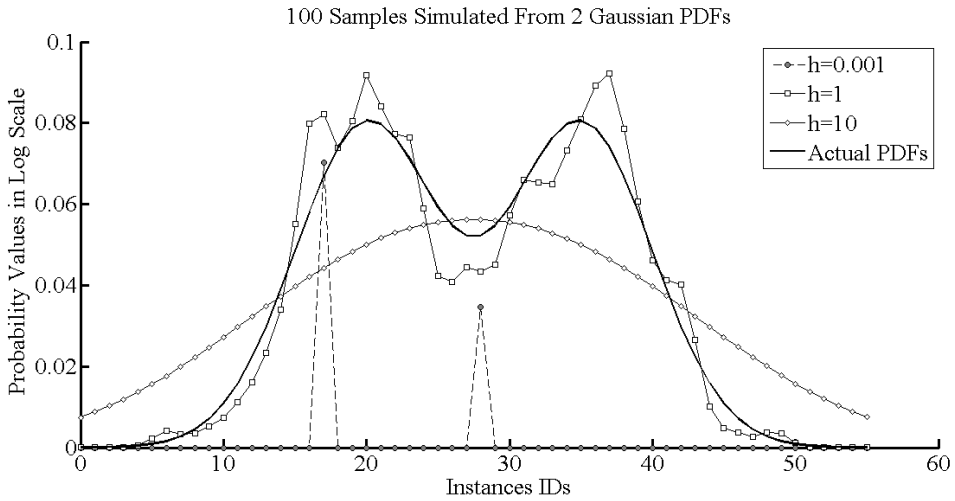
1. Supplementary tables: http://unbox.org/wisp/var/ekrem/kernel/supplementaryresults/.
2. E. Alpaydin. *Introduction to Machine Learning.* MIT Press, 2004.
3. L. Angelis and I. Stamelos. A simulation tool for efficient analogy based cost estimation. *Empirical Softw. Engg.*, 5(1):35–68, 2000.
4. M. Auer, A. Trendowicz, B. Graser, E. Haunschmid, and S. Biffl. Optimal project feature weights in analogy-based cost estimation: Improvement and limitations. *IEEE Trans. Softw. Eng.*, 32:83–92, 2006.
5. D. Baker. A hybrid approach to expert and model-based effort estimation. Master's thesis, Lane Department of Computer Science and Electrical Engineering, West Virginia University, 2007.
6. B. Boehm, C. Abts, and S. Chulani. Software development cost estimation approaches: A survey. *Annals of Software Engineering*, 10:177–205, 2000.
7. B. W. Boehm. *Software Engineering Economics.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.

8. L. C. Briand, K. El Emam, D. Surmann, I. Wieczorek, and K. D. Maxwell. An assessment and comparison of common software cost estimation modeling techniques. pages 313–322, 1999.
9. S. D. Conte, H. E. Dunsmore, and V. Y. Shen. *Software engineering metrics and models*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1986.
10. N. A. C. Cressie. *Statistics for Spatial Data (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 1993.
11. J. Desharnais. Analyse statistique de la productivitie des projets informatique a partie de la technique des point des fonction. Master's thesis, Univ. of Montreal, 1989.
12. T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion mmre. *IEEE Trans. Softw. Eng.*, vol:29no11pp985–995, 2003.
13. E. Frank, M. Hall, and B. Pfahringer. Locally weighted naive bayes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 249–256. Morgan Kaufmann, 2003.
14. R. Jeffery, M. Ruhe, and I. Wieczorek. Using public domain metrics to estimate software development effort. In *METRICS '01: Proceedings of the 7th International Symposium on Software Metrics*, page 16, Washington, DC, USA, 2001. IEEE Computer Society.
15. G. John and P. Langley. Estimating continuous distributions in bayesian classifiers. In *In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 338–345. Morgan Kaufmann, 1995.
16. M. Jorgensen. A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70:37–60, February 2004.
17. M. Jorgensen and T. Gruschke. The impact of lessons-learned sessions on effort estimation and uncertainty assessments. *IEEE Trans. Softw. Eng.*, 35(3):368–383, May-June 2009.
18. M. Jorgensen and M. Shepperd. A systematic review of software development cost estimation studies. *IEEE Trans. Softw. Eng.*, 33(1):33–53, 2007.
19. G. Kadoda, M. Cartwright, and M. Shepperd. On configuring a case-based reasoning software project prediction system. *UK CBR Workshop, Cambridge, UK*, pages 1–10, 2000.
20. C. Kemerer. An empirical validation of software cost estimation models. *Communications of the ACM*, 30(5):416–429, May 1987.
21. B. Kitchenham and E. Mendes. Why comparative effort prediction studies may be invalid. In *PROMISE '09: Proceedings of the 5th International Conference on Predictor Models in Software Engineering*, pages 1–5, New York, NY, USA, 2009. ACM.
22. B. Kitchenham, E. Mendes, and G. H. Travassos. Cross versus within-company cost estimation studies: A systematic review. *IEEE Trans. Softw. Eng.*, 33(5):316–329, 2007. Member-Kitchenham, Barbara A.
23. B. A. Kitchenham, E. Mendes, and G. H. Travassos. Cross versus within-company cost estimation studies: A systematic review. *IEEE Trans. Softw. Eng.*, 33(5):316–329, 2007.
24. J. Li and G. Ruhe. A comparative study of attribute weighting heuristics for effort estimation by analogy. *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*, page 74, 2006.
25. Y. Li, M. Xie, and T. Goh. A study of project selection and feature weighting for analogy based software cost estimation. *Journal of Systems and Software*, 82:241–252, 2009.
26. K. Lum, T. Menzies, and D. Baker. 2CEE, A TWENTY FIRST CENTURY EFFORT ESTIMATION METHODOLOGY. *ISPA / SCEA*, pages 12 – 14, 2008.
27. E. Mendes and N. Mosley. Further investigation into the use of cbr and stepwise regression to predict development effort for web hypermedia applications. *Empirical Software Engineering, International Symposium on*, 0:79, 2002.
28. E. Mendes and N. Mosley. Bayesian network models for web effort prediction: A comparative study. *IEEE Trans. Softw. Eng.*, 34:723–737, 2008.
29. E. Mendes, N. Mosley, and I. Watson. A comparison of case-based reasoning approaches. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 272–280, New York, NY, USA, 2002. ACM.
30. E. Mendes, I. D. Watson, C. Triggs, N. Mosley, and S. Counsell. A comparative study of cost estimation models for web hypermedia applications. *Empirical Software Engineering*, 8(2):163–196, 2003.
31. T. Menzies, Z. Chen, J. Hihn, and K. Lum. Selecting best practices for effort estimation. *IEEE Trans. Softw. Eng.*, 32:883–895, 2006.
32. D. Milic and C. Wohlin. Distribution Patterns of Effort Estimations. In *Euromicro*, 2004.

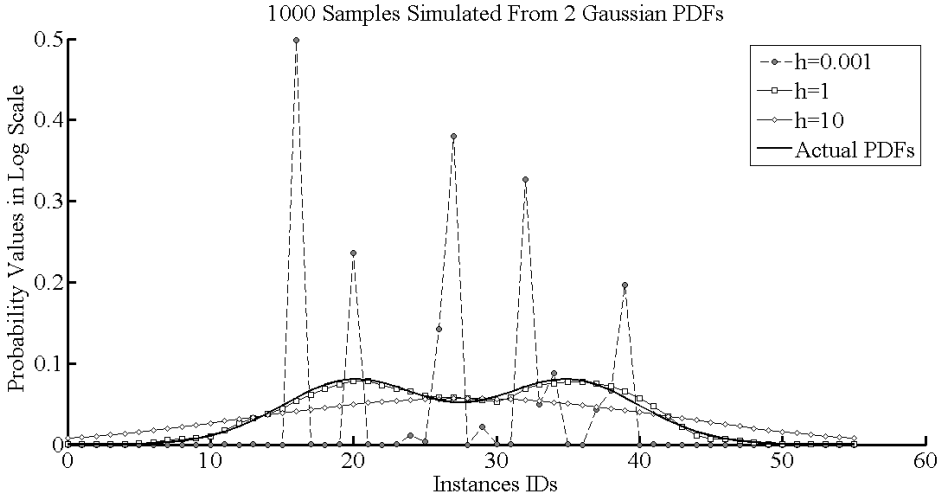
33. K. Moløkken-Østvold, M. Jørgensen, S. S. Tanilkan, H. Gallis, A. C. Lien, and S. E. Hove. A survey on software estimation in the norwegian industry. *IEEE International Symposium on Software Metrics*, pages 208–219, 2004.
34. I. Myrtveit and E. Stensrud. A controlled experiment to assess the benefits of estimating with analogy and regression models. *IEEE Trans. Softw. Eng.*, 25:510–525, 1999.
35. I. Myrtveit, E. Stensrud, and M. Shepperd. Reliability and validity in comparative studies of software prediction models. *IEEE Trans. Softw. Eng.*, vol:31no5pp380–391, May 2005.
36. T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Distributed deviation detection in sensor networks. *SIGMOD Rec*, 32:2003, 2003.
37. P. C. Pendharkar, G. H. Subramanian, and J. A. Rodger. A probabilistic model for predicting software development effort. *IEEE Trans. Softw. Eng.*, 31:615–624, 2005.
38. R. Premraj and T. Zimmermann. Building Software Cost Estimation Models using Homogenous Data. *ESEM*, 2007.
39. C. Robson. Real world research: a resource for social scientists and practitioner-researchers. *Blackwell Publisher Ltd*, 2002.
40. S. Scheid. Introduction to kernel smoothing. Talk, January 2004.
41. D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization (Wiley Series in Probability and Statistics)*. Wiley-Interscience, September 1992.
42. M. Shepperd. Software project economics: a roadmap. In *FOSE '07: 2007 Future of Software Engineering*, pages 304–315, 2007.
43. M. Shepperd and M. Cartwright. Predicting with sparse data. *IEEE Trans. Softw. Eng.*, 27:987–998, 2001.
44. M. Shepperd and G. Kadoda. Comparing software prediction models using simulation. *IEEE Trans. Softw. Eng.*, pages 1014–1022, 2001.
45. M. Shepperd and C. Schofield. Estimating software project effort using analogies. *IEEE Trans. Softw. Eng.*, 23(11):736–743, 1997.
46. M. Shepperd, C. Schofield, and B. Kitchenham. Effort estimation using analogy. In *International Conference on Software Engineering*, pages 170–178, 1996.
47. E. Stensrud, T. Foss, B. Kitchenham, and I. Myrtveit. An empirical validation of the relationship between the magnitude of relative error and project size. In *METRICS '02: Proceedings of the 8th International Symposium on Software Metrics*, page 3, Washington, DC, USA, 2002. IEEE Computer Society.
48. E. Stensrud, T. Foss, B. Kitchenham, and I. Myrtveit. A further empirical investigation of the relationship between mre and project size. *Empirical Software Engineering*, 8(2):139–161, 2003.
49. M. P. Wand and M. C. Jones. *Kernel Smoothing (Monographs on Statistics and Applied Probability)*. Chapman & Hall/CRC, December 1994.



(a) 50 Sample Points: Note the bad fit due to low sample size.



(b) 100 Sample Points: Note the better fit due to increased sample size.



(c) 1000 Sample Points: Note the optimum fit due to high sample size.

Fig. 13 The effect of sample size and bandwidth on kernel density estimation. The choice of optimum bandwidth (h value) is important. However, even with the optimum bandwidth, one still needs enough number of points for successful kernel density estimation. In this figure the best h value is 1. In terms of sample size, the value of 50 appears to be too small. As we increase the sample size to 100, we get better fit between estimated and actual probability values. But for a very close fit, we need to go up to 1000 sample points.