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 The importance of software cost models and effort estimation is obvious: They help practitioners to predict the expected cost of a project, better allocate the available resources and efficiently schedule the processes. Thereby, enabling them to finish software projects within time and budget with allocated resources. However, estimates are usually wrong by higher than acceptable factors. This shows that despite the considerable amount of research invested in software effort estimation, we still need to further investigate new methods and discover *dos and don'ts* of the domain.

Analogy based estimation (ABE) is a promising field in software effort estimation. In this research we investigate weighting analogies in the context of ABE and compare the performances of weighted analogy based estimation (WABE) and ABE. We use a novel weighting approach called kernel density estimation. Our research investigates a total of 330 settings to see the effect of instance weighting by means of kernel density estimation. Our results indicate that standart ABE methods are more successful than WABE in all our experimental settings.

Keywords Effort estimation, data mining, kernel function, bandwidth

1 Introduction

Software effort estimates are reported to be often wrong by a factor of four [6] or even more [18]. 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 budget and worst of all 3) over-running projects may be cancelled

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
School of Computer Science and Engineering
University of New South Wales
Sydney, Australia
E-mail: jacky.keung@nicta.com.au

thereby wasting the entire effort. Therefore, effort estimation is an active research area [5, 16, 20, 40] that constantly explores more variations with each model being developed or improved. For example, Auer et al. [3] proposed an extensive search to learn the best weights for different project features in 2006. Menzies et al.'s COSEEKMO tool explored thousands of combinations of discretizers, data pre-processors, feature subset selectors, and inductive learners in the same year [29]. In 2007, Baker proposed an exhaustive search of all possible project features, learners and other variables [4]. Pendharkar et. al. used Bayesian Network (BN) for effort estimation and incorporated BN into decision making procedure aginst risks [35]. Mendes and Mosley employed data-driven and hybrid BN models for web effort estimation [26]. Li et. al. investigated the feature weighting as well as instance selection in analogy based estimation domain to address the memory and computation costs in their 2009 study [23].

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.

In this research, we investigate a the concept of kernel density estimation [39]. Kernel-based methods are reported to be one of the most popular non-parametric estimators that can uncover structural features in the data [47]. Furthermore, in various different contexts different researchers have benefited from kernel density estimation and have reported successful results [11, 13, 34].

ABE is based on the premise that effort of a future project can be estimated by adapting the effort values of past k similar projects (adapted k projects are called *analogies*) [17, 23, 28]. Among proposed adaptation methods we can name choosing closest analogy [7, 12], taking mean or median of k analogies [28, 43]. 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. To overcome the equal impact problem, Mendes et. al. proposes a method called inverse rank weighted mean (IRWM) that allows higher ranked analogies to have greater influence than the lower ones [27, 28].

Experts like Mendes et. al. have an intuition about the weighting approach and use their domain knowledge to propose weighting strategies like IRWM. However, expert judgment may not be available for all practitioners willing to use ABE. In this research we use kernel density estimation as a weighting method in ABE. To the best of our knowledge, kernel methods have not been explored in this domain.

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

RQ1 Is there any evidence that weighting improves the performance of ABE?
RQ2 What is the effect of different kernels for weighting ABE?
RQ3 What is the effect of different bandwidths when used for weighting ABE?
RQ4 How do the characteristics of software effort datasets influence the performance of kernel weighting for ABE?

Our results are mostly negative (different variants of kernel estimation have little effect on estimation accuracy). However, 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 (the value of k appears to be a more important factor in ABE rather than the kernel weighting). Lastly, unlike studies in other domains concerning kernel weighting we cannot offer

supportive evidence for a statistical heuristic that the kernel does not matter but the bandwidth does.

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 4, in which we provide the details of the methodology we adopted in this research such as the weighting strategy and datasets we used as well as the experimental details and the performance criteria according to which we evaluated our results. In Section 5 we give the results of our research and continue with Section 6, where we summarize the possible threats the validity of our results. Finally we discuss the conclusions of our research in Section 7 and present our answers to the research questions we followed. 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 for ABE.

2.1 Software Effort Estimation

We can divide software effort estimation into at least two groups [40]: Expert judgment and model-based techniques.

Expert judgment methods are widely used in software effort estimation practices [14]. Expert judgment can be applied either explicitly (following a method like Delphi [5]) or implicitly (informal discussions 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 are and the improvement of human capability in making estimations is very limited. This fact is also indicated by Jorgensen et. al. and they evaluate capability of humans to improve their own expert judgment as poor [15].

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 [6]. 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 [29, 42]. 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 assumes 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 [6,19]. 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 [33]. 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) [41], 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) methods belong to this class [33].

According to the taxonomy presented by Myrtveit et. al. CBR may belong to both sparse-data or many data category [33]. 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 [33].

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. [44], Mendes et. al. [28] and Li et. al. [23], 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 is a training set).
- The columns of this set 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.

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

In Equation 1 we can see how weighting is used in the baseline approach for project features. In Equation 1, w_i corresponds to feature weights applied to independent features. ABE0 framework does not favor any features over the others, therefore ABE0 uses a uniform weighting, 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 [25]. Using effort value of the nearest neighbor [7], taking mean [32] or median [2] of closest analogies, inverse distance and inverse rank weighted mean of closest analogies are among the commonly used adaptation methods proposed in CBR literature [25]. 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. Angelis et. al. suggests that as the number of the closest projects increase, median is a robust solution [2]. 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 k values as well as low 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, which may or may not be noise. Therefore, ABE0 uses median instead of mean.

In this research we will compare the results of ABE0 framework with another version of it: Weighted Analogy Based Estimation (WABE). The word *weighted* in WABE may at first be considered to refer to both *weighting attributes* as well as *weighting anologies*. However, ABE0 framework already includes a mechanism for weighting independent attributes (see Equation 1). Therefore, when we talk about WABE, weighting will refer to weighting of instances rather than features.

WABE has been previously adressed in literature. For example inverse rank weighted mean (IRWM) was proposed by Mendes et. al. [28], which can be considered as a form of WABE. 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: $\sum_{i=1}^{n} i$

$$Effort = (3 * CA + 2 * SA + 1 * LA)/(3 + 2 + 1) \quad (2)$$

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. Being inspired by WABE methods like IRWM, in this research we question whether it is possible to develop an automated WABE approach.

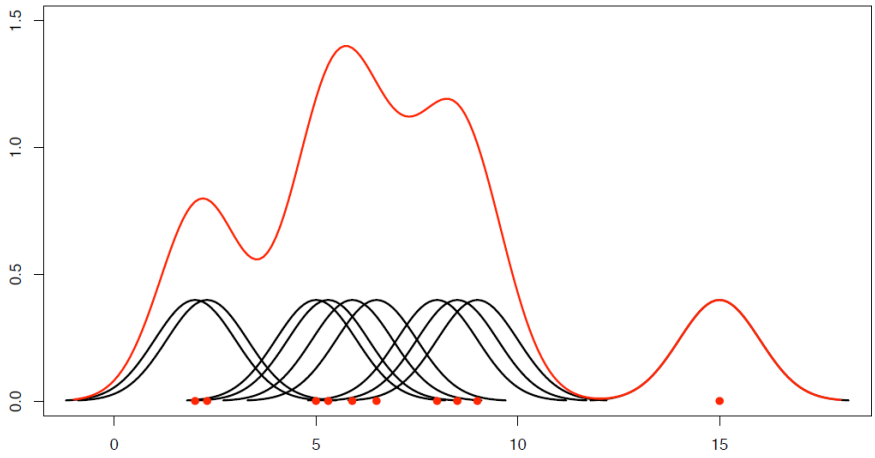


Fig. 1 We see a Gaussian kernel density estimate built on individual data points. Each point is in the center of a kernel and its effect is distributed to its neighborhood. The sum of all kernels make up the final Gaussian kernel density estimate.

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 [47]. 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 [39, 47].

Assuming that we are given a sample $X_1, ..., X_n$ with a continuous, univariate density f, the kernel density estimator is defined as in Equation 3.

$$\hat{f}(x,h) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right) \tag{3}$$

In Equation 3, K is defined as the *kernel* and h is defined to be the bandwidth. Kernel is usually chosen to be unimodal and symmetric about zero [47]. 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. We can observe how each data point and kernel contributes to the kernel estimate in Figure 1 [38].

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 [34]. Frank et. al. use kernel estimation for locally weighting the attributes of Naive Bayes, thereby relaxing the independence assumption [11]. Furthermore John et. al. use kernel estimation to tackle the normality assumption regarding continuous datasets [13]. They replace single

Gaussian distribution that is used to model continuous data with non-parametric kernel density estimation and they report considerable improvements in 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 for assigning weights to selected analogies (k values) in a WABE model.

For WABE, apart from the k value we have different parameters that can be tuned: Kernel type and bandwidth. Previously it is reported that the choice of kernel does not have a significant effect on the performance [8]. However, this statement is valid for spatial data and the effect of different kernels have not been investigated for software effort data. Therefore, in our research we inlcluded different types of kernels to observe the effect of kernel selection on effort data.

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 4, where $\mathbf{1}_{(|x|<1)}$ is the indicator function. Furthermore, Equation 5 and Equation 6 explain for the calculation of other functions in Equation 4. Depending on the value of p in Equation 4, we can derive different kernels. For example for $p = 0$ we elicit the uniform kernel, for $p = 1$ we elicit Epanechnikov kernel etc.

$$K(x, p) = \frac{\left(1 - x^2\right)^p}{2^{2p+1} B\left(p + 1, p + 1\right)} \mathbf{1}_{(|x|<1)} \tag{4}$$

$$B\left(p + 1, p + 1\right) = \frac{\Gamma\left(a\right) \Gamma\left(b\right)}{\Gamma\left(a + b\right)} \tag{5}$$

$$\Gamma\left(n\right) = (n - 1)! \tag{6}$$

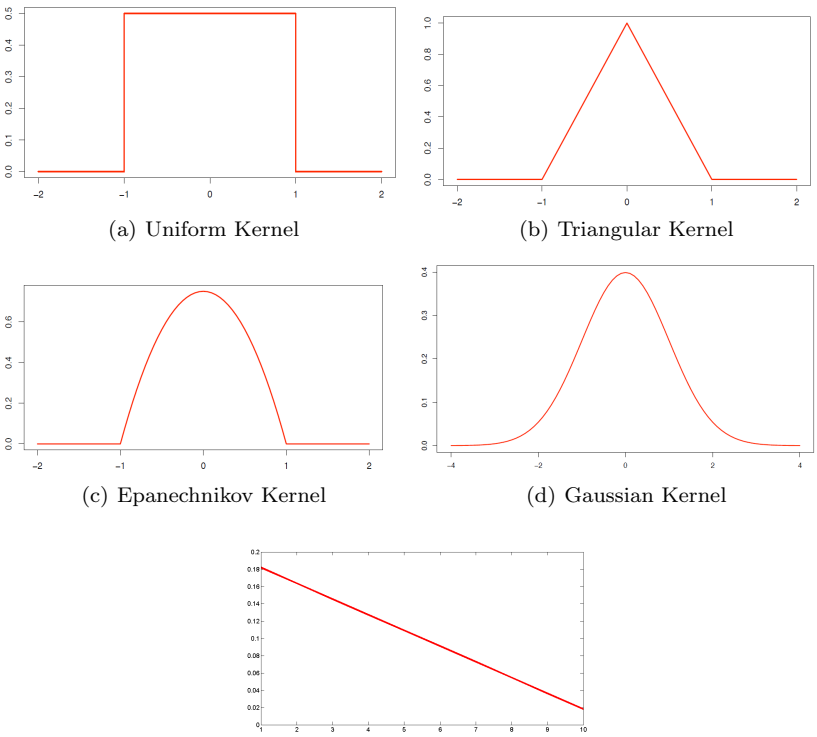
Since it is not the aim of this research, we will not go into more details regarding the derivation of kernel equations and we will suffice to provide the final formulas. The formula used for the calculation of each kernel is provided in Figure 2.

Kernel Type	**Formula**
Uniform Kernel	$K(\rho) = \frac{1}{2} \mathbf{1}_{(\|\rho\|<1)}$
Triangular Kernel	$K(\rho) = (1 - \|\rho\|) \mathbf{1}_{(\|\rho\|<1)}$
Epanechnikov Kernel	$K(\rho) = \frac{3}{4} \left(1 - \rho^2\right) \mathbf{1}_{(\|\rho\|<1)}$
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.

Furthermore, in addition to these kernels we used IRWM [27, 28] for weighting. The general shapes of these kernels are given in Figure 3 [38]. IRWM is not actually

proposed as a kernel method and it does not fully conform to the kernel definition (not being symmetrical etc.). 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).



(a) Uniform Kernel

(b) Triangular Kernel

(c) Epanechnikov Kernel

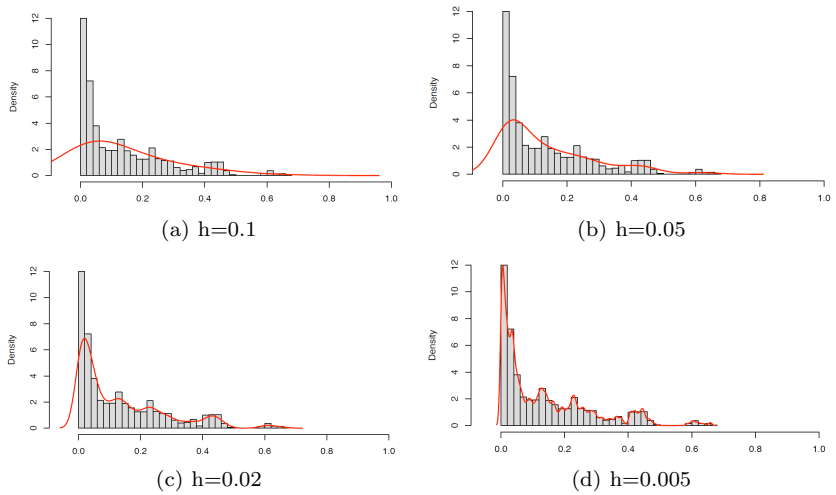
(d) Gaussian Kernel

(e) IRWM Kernel for k=10

Fig. 3 General shapes of the kernel types used in this research.

Although kernels seem to have very different shapes in Figure 3, their effect in accuracy is limited. The selection of bandwidth for kernels has been reported to have more influence on the performance than the kernel types [8, 39]. Bandwidth basically controls how wide a probability mass is spread around a data point [38]. We can use various bandwidth values for our kernel. However, using wrong bandwidth values pose the danger of both under-smoothing and over-smoothing. We can see how choosing different bandwidth values affect kernel density estimation in Figure 4 [38].

To avoid both under and over-smoothing conditions we used various bandwidth values 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 [13]. The other bandwidth values we used are: 2, 4, 8 and 16.



(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 Motivation

Software effort estimation is a very big research area and we have still discovered only a small portion of it. With every new method proposed, we discover *dos and don'ts* of it. We adopt the beneficial methods and retire the others, thereby consistently discovering the estimation domain. However, the tendency is usually to report the attempts that yield improvements on current methods and let go of the promising trials that did not result in considerable improvements. In other words mostly we hear only the *"dos"* of the domain. *"Don'ts"* part usually go unreported. However, it is fundamentally important to report the results of all promising approaches, regardless of whether it outperforms previous methods or not. Because they may save time in another similar study or may inspire new directions.

In this research we elaborate on weighting in ABE and propose a novel method for weighting analogies. For weighting we use kernel density estimation. Kernel density estimation has the advantage of being a non-parametric estimation method that can uncover particular properties of a dataset. It is reported to yield significant improvements in various settings [11,11,34], but its effects are not addressed in effort estimation domain. Software effort datasets also have particular challenges due to their inherent characteristics and we need to figure out different ways to discover these characteristics. We conduct extensive experiments with various kernels and try multiple bandwidths for each kernel in our research. At the end we cover a considerable number of settings, which were never adressed or reported before in effort estimation domain. Although kernel methods have yielded relatively successful results in different domains, we did not observe a significant improvement for ABE. Basing on the fact that other researchers may or will be conducting similar studies, we think that our results can give hints regarding kernel methods for weighting in ABE.

4 Methodology

In this section we provide the methodology that we adopted in our research. We discuss how we use kernel density estimation as a weighting method for WABE as well as which kernels we use for weighting. 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 WABE to ABE0.

4.1 Weighting Method

Here we summarize how kernel density estimation is employed as a 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 7 shows the probability calculation with kernel density estimation. In Equation 7 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.

$$f(x_k, h) = \frac{1}{nh} \sum_{t_i \in R} K\left(\frac{x_k - t_i}{h}\right) \tag{7}$$

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 $x_k \in A$. We use these probability values as weights for analogies. Note that before using a probability value as a weight, we we scale it to 0-1 interval according to Equation 8 where x_k represents all the analogies in A except x_i.

$$weight_{x_i} = \frac{f(x_i, h) - max(f(x_k, h))}{max(f(x_k, h)) - min(f(x_k, h))} \tag{8}$$

After calculating $weight_{x_i}$ for each one of the selected k analogies, we update their recorded actual effort values according to their weights. Updating the actual effort values simply means to multiply the actual effort value of an analogy with its related weight. Equation 9 shows the calculation of the updated effort value for analogy x_i. In our research WABE approaches use the updated effort values for adaptation to estimate the effort of a test instance.

$$updatedEffort_{x_i} = actualEffort_{x_i} * weight_{x_i} \tag{9}$$

4.2 Data

In our research, we have used three commonly used datasets in software effort estimation research: Nasa93, the original Cocomo81 [6], and Desharnais [9]. Cocomo81 and Nasa93 datasets contain projects developed in NASA, whereas Desharnais dataset contains projects developed by Canadian software houses.

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) [21]. 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 5.

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. 5 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.

4.3 Experiments

Our experimental settings aim at comparing the performance of standart ABE (ABE0) to that of weighted ABE (WABE). We first run ABE0 on each of the 3 datasets employed in this research. To separate train and test sets we used 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 WABE for the same test instance and store the estimated effort coming from WABE. Both for ABE0 and WABE we tried different k values since number of analogies plays a critical role in estimation accuracy. 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 ABE methods (ABE0 and WABE) induced on 3 datasets (Cocomo81, Nasa93 and Desharnais) with 5 different k values ($k \in \{1, 3, 5, 7, 9, dynamicK\}$). Furthermore, we use 4 different kernels (Uniform, triangular, Epanechnikov and Gaussian) with 5 bandwidth values as well as IRWM in WABE experiments. Therefore, to further explore field of software effort estimation, we investigate a total of 330 different settings in this research:

- ABE0 Experiments: 15 settings
  - *3 datasets * 5 k values = 15*
- WABE Experiments: 315 settings
  - Kernel Weighting: *3 datasets * 5 k values * 4 kernels * 5 bandwidths = 300*
  - IRWM: *3 datasets * 5 k values = 15*

4.4 Performance Criteria

To observe the effect of weighting in ABE, we use the following performance measures: the magnitude of relative error (MRE), median magnitude of relative error (MdMRE), mean magnitude of relative error (MMRE) 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 [7,10,31]. Furthermore, as we can see from Formula 10, MRE value is a direct measure of the absolute difference between the prediction and actual value [46] and hence it gives a per-instance based performance evaluation.

$$MRE = \frac{|actual_i - predicted_i|}{actual_i} \tag{10}$$

MMRE and MdMRE have emerged as two of the de facto standard evaluation criteria for cost estimation models [45]. 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 [28]. 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 5.1, we provide both the MMRE and MdMRE values. The formulas of MdMRE and MMRE are given in Equations 11 and 12 respectively, where n is the test set size.

$$MdMRE = median(MRE_1, MRE_2, ..., MRE_n) \tag{11}$$

$$MMRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|actual_i - predicted_i|}{actual_i} \tag{12}$$

$win_i = 0$, $tie_i = 0$, $loss_i = 0$
$win_j = 0$, $tie_j = 0$, $loss_j = 0$
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. 6 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 [10]. 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 6. 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%.

5 Results

As we have mentioned before, we will evaluate the effect of weighting closest analogies via kernel density estimation in a WABE model according to three performance measures: Win-tie-loss values, MdMRE and MMRE. In this section we present the results. We first evaluate the MMRE and MdMRE results for each dataset and then present the win-tie-loss values.

5.1 Evaluation of MRE-Based Measures

The MdMRE and MMRE values of kernel weighted WABE for Cocomo81, Nasa93 and Desharnais datasets are provided in Figure 7, Figure 8 and Figure 9 respectively. Similar to the notation of the previously introduced figures, kernel weighted settings are shown with a $+W$ sign and the *dynamic* k is represented with a d symbol.

5.1.1 Results for Cocomo81

In Figure 7 we see the MdMRE and MMRE values for Cocomo81. The results are exteremely similar to win-tie-loss results, i.e. the general trend we have observed from win-tie-loss values are present for MdMRE and MMRE: For the same k value WABE fails to improve ABE0 and smaller k values yield lower MdMRE and MMRE values. Lower k values have also yielded higher *win* and lower *loss* values, hence better performance. Furthermore application of different kernels for weighting in WABE method does not make a significant change in terms of MdMRE and MMRE results. Changing bandwidths for kernels does not create a recognizable pattern in the results either. Therefore, MdMRE and MMRE results for Cocomo81 dataset do not tell us anything further than confirmation of our previous observations from win-tie-loss results.

5.1.2 Results for Nasa93

Figure 8 lists the MdMRE as well as MMRE results for Nasa93 dataset. As we can see from Figure 8, different kernel types generate very similar results of WABE for various number of analogies (k values). In other words, change of kernel does not have a considerable effect on the performance of WABE. Furthermore, small changes due to change of kernels do not follow a particular pattern.

Like the change of kernels, changing bandwidth for a particular kernel has almost non-existent effect. We see in Figure 8 that different bandwidths generate very close MdMRE and MMRE results of WABE. More importantly there is no observable pattern in the changes due to kernel or bandwidth alterations. Another common property of Figure 8 to previous win-tie-loss figures as well as MdMRE-MMRE figure of Cocomo81 is that ABE0 methods gain higher estimation accuracies (higher *win* values, lower MdMRE-MMRE).

		h=1/sqrt(size)		h=2		h=4		h=8		h=16	
	k	MdMRE	MMRE	MdMRE	MMRE	MdMRE	MMRE	MdMRE	MMRE	MdMRE	MMRE
Uniform	3	0.33	0.35	0.33	0.35	0.35	0.40	0.31	0.35	0.40	0.40
	5	0.35	0.37	0.36	0.37	0.39	0.43	0.35	0.37	0.43	0.41
	7	0.37	0.40	0.39	0.40	0.44	0.48	0.40	0.41	0.45	0.45
	9	0.43	0.44	0.44	0.44	0.47	0.53	0.46	0.47	0.49	0.50
	d	0.79	1.32	0.82	1.32	0.78	1.19	0.62	0.72	0.61	0.66
	3+W	0.82	0.80	0.77	0.80	0.79	0.78	0.78	0.77	0.79	0.79
	5+W	0.90	0.88	0.87	0.88	0.88	0.87	0.87	0.86	0.88	0.87
	7+W	0.92	0.89	0.91	0.89	0.92	0.90	0.91	0.90	0.92	0.91
	9+W	0.93	0.90	0.93	0.90	0.94	0.92	0.93	0.92	0.94	0.92
	d+W	0.94	0.86	0.97	0.86	0.98	0.95	0.97	0.94	0.96	0.94
Triangular	3	0.33	0.36	0.28	0.35	0.33	0.36	0.36	0.39	0.33	0.39
	5	0.36	0.39	0.35	0.38	0.36	0.38	0.39	0.41	0.39	0.42
	7	0.40	0.42	0.39	0.41	0.40	0.41	0.43	0.44	0.44	0.45
	9	0.43	0.46	0.45	0.47	0.45	0.46	0.47	0.50	0.49	0.50
	d	0.82	1.65	0.69	1.06	0.63	0.72	0.54	0.61	0.68	0.90
	3+W	0.82	0.81	0.72	0.72	0.73	0.73	0.76	0.75	0.76	0.74
	5+W	0.90	0.88	0.73	0.72	0.75	0.72	0.78	0.74	0.78	0.73
	7+W	0.92	0.91	0.74	0.69	0.75	0.70	0.78	0.72	0.77	0.71
	9+W	0.94	0.91	0.74	0.69	0.75	0.68	0.77	0.71	0.77	0.70
	d+W	0.92	0.84	0.78	0.80	0.76	0.71	0.77	0.70	0.77	0.77
Epanechnikov	3	0.38	0.40	0.25	0.34	0.30	0.35	0.33	0.35	0.28	0.34
	5	0.41	0.43	0.33	0.36	0.35	0.37	0.36	0.37	0.33	0.36
	7	0.44	0.47	0.36	0.39	0.37	0.39	0.40	0.41	0.38	0.39
	9	0.50	0.54	0.43	0.44	0.43	0.44	0.46	0.46	0.44	0.44
	d	0.56	0.64	0.44	0.47	0.60	0.60	0.50	0.52	0.51	0.54
	3+W	0.84	0.83	0.66	0.67	0.68	0.68	0.69	0.69	0.68	0.68
	5+W	0.92	0.89	0.65	0.65	0.67	0.65	0.68	0.66	0.67	0.65
	7+W	0.93	0.90	0.64	0.61	0.67	0.62	0.68	0.63	0.66	0.62
	9+W	0.94	0.89	0.65	0.61	0.66	0.61	0.67	0.63	0.65	0.61
	d+W	0.94	0.89	0.65	0.61	0.69	0.66	0.68	0.63	0.67	0.63
Gaussian	3	0.42	0.42	0.33	0.35	0.33	0.37	0.33	0.38	0.41	0.40
	5	0.43	0.43	0.36	0.36	0.37	0.40	0.36	0.40	0.43	0.42
	7	0.46	0.46	0.39	0.40	0.40	0.42	0.40	0.45	0.47	0.46
	9	0.49	0.52	0.44	0.44	0.44	0.47	0.45	0.49	0.50	0.51
	d	0.59	0.65	0.66	0.99	0.63	0.74	0.60	0.67	0.61	0.69
	3+W	0.84	0.83	0.69	0.69	0.71	0.70	0.70	0.70	0.73	0.73
	5+W	0.92	0.89	0.68	0.66	0.70	0.68	0.68	0.67	0.71	0.69
	7+W	0.94	0.90	0.67	0.63	0.68	0.64	0.66	0.63	0.70	0.66
	9+W	0.95	0.91	0.67	0.62	0.67	0.63	0.66	0.63	0.70	0.65
	d+W	0.95	0.90	0.71	0.79	0.70	0.71	0.69	0.68	0.71	0.70

Fig. 7 MdMRE and MMRE results for Cocomo81 dataset. The column ***k*** lists the k values. $+W$ stands for weighting, i.e. WABE. Cocomo81 results confirm the previous conclusions: 1) Neither the bandwidth nor the kernel type have a significant effect on the performance and 2) WABE via kernel methods do not outperform ABE0.

5.1.3 Results for Desharnais

We provide the MdMRE and MMRE values for Desharnais dataset in Figure 9. Among all the kernels-bandwidth combinations we do not see a case where WABE improves the performance of ABE0. Therefore, particular characteristic of being indifferent to kernel methods that we observed in previous experiments is valid for Desharnais dataset as well. Furthermore, what we see from Figure 9 is that instead of improving ABE0 methods, kernel weighted WABE methods generate considerably worse MdMRE and MMRE results. Only in one case (Epanechnikov kernel) do the MdMRE and MMRE values for WABE goes down to values around 0.6. However, that is still far worse than the standart ABE0 values. These results suggest that non-parametric weighting for

		h=1/sqrt(size)		h=2		h=4		h=8		h=16	
	k	MdMRE	MMRE	MdMRE	MMRE	MdMRE	MMRE	MdMRE	MMRE	MdMRE	MMRE
Uniform	3	0.40	0.37	0.23	0.34	0.20	0.29	0.43	0.37	0.43	0.39
	5	0.43	0.39	0.26	0.35	0.21	0.31	0.43	0.39	0.44	0.42
	7	0.43	0.43	0.33	0.39	0.25	0.34	0.43	0.43	0.44	0.45
	9	0.43	0.46	0.35	0.43	0.29	0.39	0.43	0.46	0.44	0.48
	d	0.81	1.75	0.30	0.34	0.42	0.57	0.43	0.46	0.49	0.60
	3+W	0.83	0.80	0.77	0.77	0.75	0.75	0.78	0.78	0.79	0.78
	5+W	0.90	0.87	0.86	0.86	0.85	0.85	0.87	0.86	0.88	0.86
	7+W	0.93	0.90	0.90	0.89	0.89	0.89	0.91	0.90	0.91	0.90
	9+W	0.94	0.92	0.92	0.92	0.92	0.91	0.93	0.92	0.93	0.92
	d+W	0.90	0.84	0.83	0.82	0.97	0.96	0.93	0.92	0.97	0.96
Triangular	3	0.32	0.34	0.30	0.35	0.45	0.41	0.29	0.35	0.30	0.36
	5	0.40	0.37	0.40	0.38	0.46	0.42	0.31	0.36	0.40	0.39
	7	0.43	0.40	0.40	0.42	0.47	0.46	0.37	0.40	0.41	0.42
	9	0.42	0.43	0.40	0.45	0.47	0.48	0.37	0.43	0.40	0.46
	d	0.32	0.34	0.40	0.49	0.50	0.60	0.31	0.36	0.44	0.54
	3+W	0.83	0.79	0.76	0.72	0.79	0.74	0.78	0.73	0.80	0.74
	5+W	0.90	0.86	0.77	0.71	0.78	0.73	0.77	0.72	0.78	0.73
	7+W	0.92	0.89	0.76	0.71	0.77	0.72	0.76	0.71	0.78	0.73
	9+W	0.94	0.92	0.75	0.71	0.76	0.72	0.75	0.71	0.77	0.72
	d+W	0.83	0.79	0.74	0.70	0.73	0.71	0.77	0.72	0.74	0.71
Epanechnikov	3	0.29	0.33	0.34	0.34	0.45	0.39	0.45	0.40	0.48	0.43
	5	0.39	0.36	0.40	0.36	0.45	0.41	0.46	0.42	0.48	0.45
	7	0.39	0.40	0.41	0.40	0.46	0.45	0.47	0.45	0.49	0.49
	9	0.39	0.44	0.40	0.43	0.46	0.48	0.47	0.48	0.49	0.52
	d	0.40	0.50	0.57	0.71	0.45	0.50	0.47	0.46	0.62	0.72
	3+W	0.83	0.79	0.74	0.71	0.76	0.73	0.75	0.72	0.80	0.75
	5+W	0.90	0.86	0.71	0.68	0.74	0.70	0.72	0.68	0.75	0.71
	7+W	0.93	0.90	0.68	0.66	0.72	0.69	0.70	0.67	0.73	0.70
	9+W	0.95	0.92	0.67	0.65	0.70	0.68	0.68	0.66	0.71	0.69
	d+W	0.96	0.94	0.66	0.71	0.68	0.67	0.69	0.66	0.68	0.72
Gaussian	3	0.29	0.34	0.41	0.37	0.43	0.35	0.25	0.32	0.46	0.43
	5	0.37	0.37	0.41	0.39	0.43	0.37	0.28	0.34	0.47	0.44
	7	0.38	0.41	0.44	0.42	0.44	0.40	0.32	0.38	0.49	0.48
	9	0.38	0.44	0.42	0.45	0.43	0.43	0.33	0.41	0.49	0.51
	d	0.36	0.42	0.67	0.82	0.48	0.59	0.30	0.33	0.51	0.63
	3+W	0.83	0.79	0.78	0.72	0.69	0.68	0.72	0.69	0.78	0.74
	5+W	0.90	0.86	0.76	0.70	0.67	0.64	0.70	0.66	0.74	0.70
	7+W	0.93	0.90	0.73	0.69	0.66	0.63	0.69	0.65	0.72	0.69
	9+W	0.94	0.92	0.71	0.68	0.65	0.63	0.67	0.64	0.70	0.68
	d+W	0.93	0.91	0.70	0.83	0.63	0.65	0.71	0.67	0.66	0.68

Fig. 8 MdMRE and MMRE results for Nasa93 dataset. Neither change of kernel nor the change of bandwidth generates a considerable difference in results. Furthermore, small changes in MdMRE and MMRE values due to different kernel-bandwidth combinations do not follow a regular pattern. Another cocnlusions from this figure is that WABE fails to improve ABE0 and lower k values generate lower MdMRE-MMRE values.

WABE method may not be a good idea. Therefore, we will finally take a look at the MdMRE and MMRE values of an expert-weighted WABE method: IRWM.

5.1.4 IRWM Results for All Datasets

Figure 10 presents our last table for MdMRE and MMRE results. The difference between the previous MdMRE-MMRE results and the ones in Figure 10 is that previous results belong to a WABE method in which weighting was done via non-parametric methods (minimum human interaction), whereas results in Figure 10 belong to a WABE method whose weights are assigned by human experts (complete human dependence).

		h=1/sqrt(size)		h=2		h=4		h=8		h=16	
	k	MdMRE	MMRE	MdMRE	MMRE	MdMRE	MMRE	MdMRE	MMRE	MdMRE	MMRE
Uniform	3	0.22	0.25	0.29	0.30	0.19	0.25	0.25	0.27	0.23	0.26
	5	0.21	0.25	0.30	0.31	0.19	0.25	0.25	0.27	0.23	0.27
	7	0.23	0.26	0.32	0.32	0.20	0.26	0.26	0.28	0.25	0.28
	9	0.24	0.27	0.33	0.32	0.24	0.27	0.28	0.29	0.27	0.29
	d	0.21	0.25	0.36	0.40	0.28	0.33	0.27	0.28	0.36	0.51
	3+W	0.77	0.76	0.76	0.76	0.75	0.75	0.75	0.76	0.75	0.75
	5+W	0.86	0.84	0.86	0.86	0.85	0.85	0.85	0.85	0.85	0.85
	7+W	0.90	0.88	0.90	0.90	0.89	0.89	0.90	0.89	0.89	0.89
	9+W	0.92	0.89	0.92	0.92	0.92	0.91	0.92	0.91	0.92	0.92
	d+W	0.86	0.84	0.96	0.95	0.95	0.94	0.88	0.88	0.98	0.97
Triangular	3	0.25	0.28	0.20	0.25	0.28	0.29	0.27	0.29	0.27	0.29
	5	0.25	0.29	0.20	0.25	0.27	0.30	0.27	0.29	0.28	0.29
	7	0.27	0.30	0.22	0.26	0.28	0.30	0.28	0.30	0.30	0.31
	9	0.30	0.31	0.24	0.27	0.30	0.30	0.29	0.31	0.32	0.31
	d	0.33	0.37	0.21	0.26	0.29	0.30	0.35	0.45	0.40	0.54
	3+W	0.48	0.49	0.67	0.65	0.67	0.67	0.67	0.66	0.66	0.65
	5+W	0.39	0.41	0.65	0.64	0.66	0.65	0.66	0.64	0.64	0.63
	7+W	0.36	0.38	0.65	0.62	0.66	0.62	0.65	0.62	0.63	0.60
	9+W	0.35	0.37	0.64	0.61	0.65	0.61	0.64	0.61	0.62	0.59
	d+W	0.35	0.39	0.65	0.63	0.65	0.61	0.62	0.58	0.61	0.58
Epanechnikov	3	0.23	0.26	0.26	0.29	0.24	0.26	0.19	0.26	0.27	0.30
	5	0.23	0.26	0.27	0.29	0.23	0.26	0.20	0.26	0.26	0.30
	7	0.24	0.27	0.29	0.30	0.24	0.27	0.21	0.27	0.28	0.30
	9	0.25	0.28	0.30	0.32	0.26	0.28	0.23	0.28	0.30	0.31
	d	0.30	0.37	0.36	0.51	0.30	0.34	0.30	0.39	0.44	0.63
	3+W	0.77	0.77	0.67	0.66	0.64	0.63	0.65	0.63	0.65	0.64
	5+W	0.87	0.85	0.64	0.62	0.62	0.60	0.62	0.60	0.62	0.61
	7+W	0.90	0.89	0.62	0.59	0.60	0.57	0.59	0.56	0.60	0.58
	9+W	0.92	0.90	0.61	0.58	0.58	0.55	0.58	0.54	0.59	0.56
	d+W	0.95	0.94	0.56	0.54	0.56	0.53	0.54	0.51	0.53	0.59
Gaussian	3	0.28	0.30	0.29	0.30	0.20	0.25	0.24	0.26	0.21	0.26
	5	0.29	0.30	0.29	0.31	0.22	0.25	0.23	0.26	0.23	0.26
	7	0.31	0.31	0.30	0.31	0.23	0.26	0.24	0.26	0.24	0.27
	9	0.33	0.32	0.32	0.32	0.24	0.27	0.26	0.28	0.26	0.28
	d	0.36	0.37	0.36	0.38	0.23	0.26	0.33	0.42	0.26	0.27
	3+W	0.79	0.77	0.66	0.65	0.64	0.63	0.63	0.63	0.64	0.63
	5+W	0.87	0.85	0.64	0.62	0.61	0.60	0.60	0.59	0.61	0.59
	7+W	0.90	0.88	0.61	0.58	0.60	0.57	0.57	0.55	0.58	0.56
	9+W	0.92	0.90	0.59	0.56	0.58	0.55	0.57	0.54	0.56	0.54
	d+W	0.93	0.92	0.57	0.54	0.60	0.59	0.54	0.52	0.67	0.67

Fig. 9 MdMRE and MMRE results for Desharnais dataset. None of the different kernel-bandwidth combinations can improve the performance of WABE to a point better than ABE0 method.

The weighting strategies between previous figures and Figure 10 are different. However, the trend in the results are very alike, i.e. in none of the 3 datasets can WABE methods outperform ABE0 methods. Therefore, after 330 settings which involve both non-parametric methods and expert methods for weighting WABE, we still do not observe a case where WABE methods could bring an improvement on simple ABE0 method.

5.2 Evaluation of WIN-TIE-LOSS Results

Since we have 10 settings for each kernel subject to 20 runs, the sum of *win*, *tie* and *loss* values can be at most 180 (*(10 settings - 1 setting itself) * 20 = 180*).

		Cocomo81		Nasa93		Desharnais	
	k	MdMRE	MMRE	MdMRE	MMRE	MdMRE	MMRE
IRWM	3	0.42	0.43	0.40	0.36	0.24	0.27
	5	0.44	0.45	0.42	0.38	0.23	0.27
	7	0.48	0.49	0.43	0.42	0.24	0.28
	9	0.51	0.54	0.43	0.45	0.27	0.29
	d	0.86	2.04	0.80	1.79	0.29	0.31
	3+W	0.59	0.58	0.57	0.56	0.50	0.51
	5+W	0.64	0.62	0.61	0.60	0.55	0.55
	7+W	0.66	0.64	0.63	0.62	0.57	0.57
	9+W	0.68	0.65	0.64	0.63	0.59	0.58
	d+W	0.79	1.41	0.74	0.95	0.60	0.58

Fig. 10 MdMRE and MMRE results of Cocomo81, Nasa93 and Desharnais for IRWM weighted WABE. ***k*** stands for the number of analogies used for estimation and $+W$ sign means that IRWM weighted WABE is used for estimation. Similar to kernel weighted WABE, expert weighted WABE can not perform an improvement to ABE0 method.

5.2.1 Results for Cocomo81

In Figure 11 the win-tie-loss values for Cocomo81 are given. The first observationg we can make from Figure 11 is that smaller number of analogies have always attained higher *win* values and lower *loss* values. In other words, in all treatments $k = 3$ attains the highest *win* and the lowest *loss* values.

Remember that the total sum of win-tie-loss values for a single treatment can be at most 180. For all settings, the *tie* values are most of the time less than 45 (less than 25% of all the comparisons), which means that in 75% or more of the comparisons there is a statistical difference between two methods. Furthermore, when we mutually compare the results of ABE0 with WABE for a single k value, we see that for none of the k values weighting via kernel density estimation improves the *win* values.

From Figure 11 we can also see the effect of applying different kernels and different bandwidths on the performance of WABE. In terms of kernels, we can say that there is not a considerable performance difference between different types. Note that our results are consistent with prior research that reported different kernels yield similar results [8]. For Cocomo81 dataset we observe that the same fact is also valid for software effort estimation data.

The bandwidth was reported to be influential in different contexts [8, 39, 47]. However, we are unable to observe the considerable effect of various bandwidths on software effort estimation data. In Figure 11 the win-tie-loss values kernels when used with 5 different bandwidths are very similar. In fact, for the uniform kernel the performance is completely identical between different bandwidths. Therefore, from Cocomo81 dataset we see that software effort data behaves differently than other data types, i.e. unlike spatial data software effort data does not respond to changes in bandwidths.

5.2.2 Results for Nasa93

Figure 12 shows the win-tie-loss results for Nasa93 dataset. The results for Nasa93 are extremely similar to Cocomo81, that is in all cases the highest *win* values belong to $k = 3$ and *tie* values are usually around 25% of 180 comparisons. Furthermore, application of different kernels for WABE does not yield a considerable difference.

		h=1/sqrt(size)			h=2			h=4			h=8			h=16		
	k	**WIN**	**TIE**	**LOSS**	**WIN**	**TIE**	**LOSS**	**WIN**	**TIE**	**LOSS**	**WIN**	**TIE**	**LOSS**	**WIN**	**TIE**	**LOSS**
Uniform	3	144	32	4	144	32	4	144	32	4	144	32	4	144	32	4
	5	127	48	5	127	48	5	127	48	5	127	48	5	127	48	5
	7	116	48	16	116	48	16	116	48	16	116	48	16	116	48	16
	9	113	28	39	113	28	39	113	28	39	113	28	39	113	28	39
	d	76	24	80	76	24	80	76	24	80	76	24	80	76	24	80
	3+W	74	51	55	74	51	55	74	51	55	74	51	55	74	51	55
	5+W	28	34	118	28	34	118	28	34	118	28	34	118	28	34	118
	7+W	12	41	127	12	41	127	12	41	127	12	41	127	12	41	127
	9+W	6	34	140	6	34	140	6	34	140	6	34	140	6	34	140
	d+W	16	32	132	16	32	132	16	32	132	16	32	132	16	32	132
Triangular	3	147	31	2	147	31	2	147	31	2	147	31	2	147	31	2
	5	132	46	2	132	46	2	132	46	2	132	46	2	132	46	2
	7	123	43	13	123	43	14	123	43	14	123	43	14	123	43	14
	9	116	23	40	116	23	41	116	23	41	116	23	41	116	23	41
	d	97	2	80	97	2	81	97	2	81	97	2	81	97	2	81
	3+W	78	17	85	78	17	85	78	17	85	78	17	85	78	17	85
	5+W	54	8	117	54	8	118	54	8	118	54	8	118	54	8	118
	7+W	22	23	134	22	23	135	22	23	135	22	23	135	22	23	135
	9+W	9	25	145	9	25	146	9	25	146	9	25	146	9	25	146
	d+W	0	22	157	0	22	158	0	22	158	0	22	158	0	22	158
Epanechnikov	3	145	33	1	145	33	2	145	33	2	145	33	2	145	33	2
	5	139	38	2	139	38	3	139	38	3	139	38	3	139	38	3
	7	124	40	15	124	40	16	124	40	16	124	40	16	124	40	16
	9	116	18	45	116	18	46	116	18	46	116	18	46	116	18	46
	d	97	4	78	97	4	79	97	4	79	97	4	79	97	4	79
	3+W	79	16	85	79	16	85	79	16	85	79	16	85	79	16	85
	5+W	41	18	120	41	18	121	41	18	121	41	18	121	41	18	121
	7+W	10	42	127	10	42	128	10	42	128	10	42	128	10	42	128
	9+W	6	40	133	6	40	134	6	40	134	6	40	134	6	40	134
	d+W	2	29	148	2	29	149	2	29	149	2	29	149	2	29	149
Gaussian	3	136	42	2	137	32	11	138	38	4	139	36	5	142	32	6
	5	130	48	2	129	40	11	133	44	3	131	44	5	132	42	6
	7	116	57	7	117	41	22	122	47	11	119	46	15	122	41	17
	9	114	33	33	108	19	53	115	25	40	114	26	40	113	22	45
	d	95	7	78	78	27	75	88	16	76	70	32	78	95	4	81
	3+W	66	34	80	80	60	40	80	24	76	79	40	61	85	21	74
	5+W	27	39	114	59	13	108	61	3	116	61	14	105	60	5	115
	7+W	7	50	123	41	10	129	40	3	137	39	7	134	38	7	135
	9+W	4	53	123	20	10	150	19	4	157	20	6	154	20	7	153
	d+W	1	45	134	0	10	170	0	4	176	0	5	175	0	5	175

Fig. 11 Win-tie-loss results for Cocomo81. The WABE experiments are shown with a $+W$ sign, whereas the dynamic k is represented with a d under the column $\boldsymbol{k}$. We used 5 different bandwidths (represented with $\boldsymbol{h}$) for 4 different kernels. Similar to other data types, for Cocomo81 we do not see an improvement coming from different kernels. However, unlike other data types, we are unable to observe an improvement coming from change of bandwidth values.

For instance, for the treatment $k = 3$ and $h = 1/sqrt(size)$ the difference between the highest and the lowest *win* value (141 and 122 respectively) is only 19, which is around 10% of all 180 comparisons. Similar to the effect of changing kernels, changing bandwidth also falls short of providing any noticable increase or decrease in estimation performance. Furthermore, we need to point out in Figure 12 is hat in none of the k values has WABE provided any improvement in estimation accuracy. This shows us that like Cocomo81 dataset, Nasa93 dataset does not favor WABE over ABE0.

		h=1/sqrt(size)			h=2			h=4			h=8			h=16		
	k	**WIN**	**TIE**	**LOSS**	**WIN**	**TIE**	**LOSS**	**WIN**	**TIE**	**LOSS**	**WIN**	**TIE**	**LOSS**	**WIN**	**TIE**	**LOSS**
Uniform	3	141	39	0	138	42	0	146	34	0	138	42	0	146	34	0
	5	135	45	0	127	52	1	130	49	1	129	51	0	133	46	1
	7	120	52	8	111	56	13	119	46	15	120	53	7	118	50	12
	9	119	32	29	105	50	25	116	36	28	120	39	21	114	35	31
	d	100	2	78	100	38	42	100	13	67	100	1	79	100	13	67
	3+W	76	4	100	80	0	100	80	0	100	80	0	100	80	0	100
	5+W	51	10	119	60	0	120	60	0	120	60	0	120	60	0	120
	7+W	27	18	135	40	0	140	40	0	140	40	0	140	40	0	140
	9+W	16	15	149	20	0	160	20	0	160	20	0	160	20	0	160
	d+W	2	9	169	0	0	180	0	0	180	0	0	180	0	0	180
Triangular	3	122	47	11	119	46	15	125	43	12	128	40	12	110	53	17
	5	115	52	13	107	57	16	115	54	11	120	49	11	98	63	19
	7	103	60	17	97	58	25	104	57	19	110	49	21	88	61	31
	9	99	41	40	91	52	37	104	46	30	109	35	36	83	36	61
	d	90	32	58	85	39	56	90	14	76	89	5	86	98	63	19
	3+W	91	44	45	71	68	41	50	57	73	55	62	63	77	77	26
	5+W	59	12	109	11	59	110	11	50	119	3	60	117	7	73	100
	7+W	32	20	128	6	67	107	15	53	112	9	61	110	2	77	101
	9+W	16	22	142	10	68	102	19	54	107	12	59	109	9	66	105
	d+W	0	16	164	17	58	105	37	32	111	33	44	103	7	73	100
Epanechnikov	3	139	41	0	135	43	2	144	35	1	133	47	0	137	43	0
	5	126	54	0	118	61	1	133	47	0	124	55	1	121	58	1
	7	122	48	10	108	56	16	122	48	10	112	62	6	111	59	10
	9	121	41	18	103	56	21	119	31	30	112	49	19	112	53	15
	d	100	0	80	102	52	26	99	4	77	100	25	55	100	25	55
	3+W	77	3	100	0	22	158	0	22	158	0	15	165	0	7	173
	5+W	48	13	119	16	34	130	16	34	130	15	28	137	16	28	136
	7+W	21	24	135	24	44	112	26	42	112	27	32	121	24	36	120
	9+W	14	24	142	27	49	104	34	40	106	38	30	112	39	34	107
	d+W	2	12	166	42	33	105	39	33	108	61	13	106	57	23	100
Gaussian	3	124	44	12	122	45	13	102	60	18	127	41	12	117	52	11
	5	113	54	13	113	54	13	92	70	18	119	50	11	114	56	10
	7	97	63	20	103	57	20	86	71	23	108	55	17	105	61	14
	9	90	53	37	105	46	29	83	66	31	108	38	34	107	52	21
	d	88	42	50	88	13	79	83	61	36	85	8	87	90	13	77
	3+W	92	48	40	60	52	68	75	60	45	50	47	83	50	46	84
	5+W	55	16	109	7	38	135	20	37	123	13	37	130	8	32	140
	7+W	23	33	124	16	50	114	17	56	107	16	48	116	17	49	114
	9+W	4	40	136	25	44	111	19	61	100	23	48	109	26	46	108
	d+W	0	35	145	44	35	101	24	56	100	48	34	98	47	31	102

Fig. 12 Win-tie-loss results for Nasa93. Results we have for Nasa93 are very similar to Cocomo81 dataset: Neither changing kernels nor the bandwidths provides a noticable change in win-tie-loss values. Also ABE0 results are better than the WABE values.

5.2.3 Results for Desharnais

The win-tie-loss values for our last dataset Desharnais are given in Figure 13. The interpretation of Figure 13 shows us a similar scenario to previous two datasets: Highest *win* values were attained by $k = 3$ and the treatments are statistically different from one another for most of the cases. Furthermore, just like the Cocomo81 and Nasa93 datasets, the effect of different kernels as well as the effect of various bandwidths are negligible and do not follow a certain pattern. Another similarity is that in none of the kernel-bandwidth combinations has WABE yielded higher estimation performance than ABE0.

		h=1/sqrt(size)			h=2			h=4			h=8			h=16		
	k	**WIN**	**TIE**	**LOSS**	**WIN**	**TIE**	**LOSS**	**WIN**	**TIE**	**LOSS**	**WIN**	**TIE**	**LOSS**	**WIN**	**TIE**	**LOSS**
Uniform	3	123	55	2	123	57	0	120	60	0	120	59	1	126	54	0
	5	124	56	0	121	59	0	118	62	0	119	61	0	121	59	0
	7	116	61	3	116	62	2	115	64	1	115	64	1	114	64	2
	9	116	53	11	115	56	9	115	59	6	115	59	6	115	50	15
	d	101	15	64	100	16	64	100	19	61	101	17	62	101	19	60
	3+W	79	1	100	80	0	100	80	0	100	80	0	100	80	0	100
	5+W	52	9	119	60	0	120	60	0	120	60	0	120	60	0	120
	7+W	26	21	133	40	0	140	40	0	140	40	0	140	40	0	140
	9+W	18	16	146	20	0	160	20	0	160	20	0	160	20	0	160
	d+W	0	3	177	0	0	180	0	0	180	0	0	180	0	0	180
Triangular	3	120	60	0	122	58	0	122	57	1	114	65	1	112	68	0
	5	116	64	0	122	58	0	120	60	0	108	72	0	103	77	0
	7	102	76	2	115	63	2	114	65	1	100	79	1	100	76	4
	9	101	64	15	111	56	13	104	69	7	100	70	10	100	65	15
	d	96	48	36	100	25	55	101	27	52	100	70	10	104	76	0
	3+W	0	0	180	2	34	144	0	44	136	0	46	134	0	47	133
	5+W	20	15	145	3	53	124	5	59	116	3	65	112	8	66	106
	7+W	33	50	97	14	53	113	12	62	106	11	63	106	16	64	100
	9+W	36	49	95	23	54	103	17	61	102	19	61	100	26	53	101
	d+W	39	48	93	42	38	100	27	52	101	19	61	100	3	64	113
Epanechnikov	3	132	48	0	130	50	0	126	54	0	123	57	0	123	57	0
	5	118	61	1	126	53	1	118	61	1	120	60	0	123	57	0
	7	116	56	8	119	59	2	105	73	2	114	65	1	117	60	3
	9	114	57	9	118	49	13	100	68	12	114	53	13	110	61	9
	d	103	12	65	93	10	77	100	46	34	100	23	57	100	19	61
	3+W	80	0	100	1	26	153	0	21	159	0	12	168	0	18	162
	5+W	57	2	121	9	41	130	10	43	127	12	36	132	11	36	133
	7+W	34	8	138	23	42	115	18	59	103	23	42	115	21	49	110
	9+W	20	8	152	34	35	111	29	51	100	35	36	109	30	46	104
	d+W	0	0	180	49	31	100	32	48	100	54	26	100	47	33	100
Gaussian	3	121	59	0	122	58	0	126	54	0	126	54	0	123	57	0
	5	121	59	0	115	65	0	120	60	0	119	61	0	115	64	1
	7	117	62	1	113	65	2	118	59	3	113	65	2	113	65	2
	9	115	55	10	109	63	8	118	53	9	113	55	12	108	65	7
	d	100	17	63	101	29	50	98	12	70	100	23	57	102	27	51
	3+W	80	0	100	0	17	163	0	21	159	0	19	161	0	14	166
	5+W	54	4	122	10	30	140	10	44	126	10	38	132	10	36	134
	7+W	35	11	134	25	36	119	20	48	112	17	45	118	23	43	114
	9+W	20	7	153	36	36	108	32	36	112	31	37	112	33	40	107
	d+W	0	0	180	59	21	100	50	29	101	65	15	100	55	25	100

Fig. 13 Win-tie-loss results for Desharnais. The implications we have observed in Cocomo81 and Nasa93 repeats for Desharnais dataset: Change of kernels does not provide a significant change in win-tie-loss values and neither does changing bandwidth. There are some small changes in different kernel-bandwidth combinations but we can not observe a pattern. Furthermore, ABE0 has a better estimation performance than WABE.

5.2.4 IRWM Results for All Datasets

Up to this point we have observed 315 different settings and saw that neither kernel nor the bandwidth change does have a considerable impact on the performance of WABE. Furthermore, we found out that simple ABE0 approach yields higher performance measures in terms of win-tie-loss values. However, kernel estimation is not the only alternative of weighting in a WABE model. Another WABE weighting approach we use in this research is so called IRWM [27, 28]. The win-tie-loss values of all 3 datasets for IRWM weighted WABE are given in Figure 14. Since IRWM is a different weighting approach than kernel density estimation, we do not have kernels or bandwidths to compare in that scenario. On the other hand with IRWM results we can mutually

compare the estimation performances of WABE and ABE0 approaches. Our reading from Figure 14 is that for none of the three dataset does WABE outperform ABE0. In other words, just like the kernel weighted WABE, IRWM weighted WABE also fails to improve the ABE0 performance. Therefore, in a total of 330 settings (315 for kernel weighted WABE and 15 for IRWM weighted WABE) we see that WABE is unable to improve the performance of simple ABE0 approach.

		Cocomo81			**Nasa93**			**Desharnais**		
	k	**WIN**	**TIE**	**LOSS**	**WIN**	**TIE**	**LOSS**	**WIN**	**TIE**	**LOSS**
IRWM	3	143	37	0	141	39	0	126	54	0
	5	128	50	2	126	54	0	120	60	0
	7	115	55	10	115	53	12	115	62	3
	9	101	46	33	117	43	20	116	55	9
	d	0	56	124	97	16	67	101	13	66
	3+W	88	25	67	78	4	98	80	0	100
	5+W	49	48	83	49	14	117	50	10	120
	7+W	28	59	93	22	29	129	24	25	131
	9+W	23	52	105	19	19	142	16	19	145
	d+W	0	22	158	0	1	179	0	6	174

Fig. 14 Win-tie-loss results of Cocomo81, Nasa93 and Desharnais for IRWM weighted WABE. The notation in this figure is similar to previous figures: Weighting is represented by a $+W$ sign and dynamic kernel is represented by a d sign. IRWM is a different weighting strategy than kernel weighting, hence we do not see kernel or bandwidth information in this figure. Results are similar to previous scenarios: Lower k values attain higher *win* values and lower *loss* values. Furhermore, most importantly WABE is unable to outperform ABE0.

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 [1].
- *External validity* questions the ability to generalize the results [30].
- *Construct validity* (i.e. face validity) makes sure that we in fact measure what we intend to measure [37].

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 WABE on performance under 330 settings. Our observations for all the settings are extremely similar. Therefore, for the datasets used in our research, our humble opinion is that the results have external validity. However, to have full confidence in our claims when saying that WABE methods fail to improve ABE0, our study needs to be replicated on other dataset and possibly with different weighting strategies.

The choice of performance measures is an open issue in software effort estimation domain. For example MMRE and MdMRE are recognized as de facto evaluation criteria for cost estimation models [45] and they appear as a practical performance evaluation option to a number of researchers [22, 24, 36]. On the other hand, use of MMRE as well as MdMRE is still criticized for being unreliable [10, 33]. Foss et. al. for instance shows that MRE can be misleding, if used as the only performance criterion [10]. 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 MMRE and MdMRE. 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 MMRE as well as MdMRE is that they give a general picture of per instance-based evaluation. However, mean and median error measures 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, MMRE and MdMRE show us the difference between two methods in terms of 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 MMRE, MdMRE 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 used kernel density estimation as a weighting strategy for WABE. We conducted extensive experiments with various kernels and observed the performance variations between ABE0 and WABE. Unlike previous studies that report improved accuracy values through the use of kernel density estimation [13, 34], we did not observe such an effect on software effort datasets. For the datasets used in our research (Cocomo81, Nasa93 and Desharnais) there was not a single case where WABE outperformed ABE0.

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

RQ2 What is the effect of different kernels for 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, performance 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 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 WABE performance.

RQ4 How do the characteristics of software effort datasets influence the performance of kernel weighting for ABE?
Kernel density estimation for WABE falls short of improving ABE0 in our experimental settings. 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.

8 Future Work

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

1. *Dataset:* In this research we used 3 commonly used 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. *Kernel Type:* We used 5 kernels (including IRWM) as 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.
3. *Weighting Strategy:* Weighting strategy in a WABE method may be completely different than kernel density estimation. Another future direction can be experimentation on different weighting strategies that are preferably based on different assumptions.

9 Acknowledgements

References

1. E. Alpaydin. *Introduction to Machine Learning.* MIT Press, 2004.

2. L. Angelis and I. Stamelos. A simulation tool for efficient analogy based cost estimation. *Empirical Softw. Engg.*, 5(1):35–68, 2000.
3. 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.
4. 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.
5. B. Boehm, C. Abts, and S. Chulani. Software development cost estimation approaches: A survey. *Annals of Software Engineering*, 10:177–205, 2000.
6. B. W. Boehm. *Software Engineering Economics*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.
7. 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.
8. N. A. C. Cressie. *Statistics for Spatial Data (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 1993.
9. 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.
10. 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.
11. 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.
12. 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.
13. G. John and P. Langley. Estimating continuous distributions in bayesian classifiers. pages 338–345, 1995.
14. M. Jorgensen. A review of studies on expert estimation of software development effort. *Journal of Systems and Software*, 70:37–60, February 2004.
15. 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.
16. M. Jorgensen and M. Shepperd. A systematic review of software development cost estimation studies. *IEEE Trans. Softw. Eng.*, 33(1):33–53, 2007.
17. 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.
18. C. Kemerer. An empirical validation of software cost estimation models. *Communications of the ACM*, 30(5):416–429, May 1987.
19. 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.
20. 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.
21. 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.
22. 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.
23. 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.
24. K. Lum, T. Menzies, and D. Baker. 2CEE, A TWENTY FIRST CENTURY EFFORT ESTIMATION METHODOLOGY. *ISPA / SCEA*, pages 12 – 14, 2008.
25. 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.
26. E. Mendes and N. Mosley. Bayesian network models for web effort prediction: A comparative study. *IEEE Transactions on Software Engineering*, 34:723–737, 2008.

27. 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.
28. 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.
29. T. Menzies, Z. Chen, J. Hihn, and K. Lum. Selecting Best Practices for Effort Estimation. *IEEE Trans. Softw. Eng.*, 32:883–895, 2006.
30. D. Milic and C. Wohlin. Distribution Patterns of Effort Estimations. In *Euromicro*, 2004.
31. 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.
32. I. Myrtveit and E. Stensrud. A controlled experiment to assess the benefits of estimating with analogy and regression models. *IEEE Transactions on Software Engineering*, 25:510–525, 1999.
33. 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.
34. T. Palpanas, D. Papadopoulos, V. Kalogeraki, and D. Gunopulos. Distributed deviation detection in sensor networks. *SIGMOD Rec*, 32:2003, 2003.
35. P. C. Pendharkar, G. H. Subramanian, and J. A. Rodger. A probabilistic model for predicting software development effort. *IEEE Transactions on Software Engineering*, 31:615–624, 2005.
36. R. Premraj and T. Zimmermann. Building Software Cost Estimation Models using Homogenous Data. *ESEM*, 2007.
37. C. Robson. Real world research: a resource for social scientists and practitioner-researchers. *Blackwell Publisher Ltd*, 2002.
38. S. Scheid. Introduction to kernel smoothing. Talk, January 2004.
39. D. W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization (Wiley Series in Probability and Statistics)*. Wiley-Interscience, September 1992.
40. M. Shepperd. Software project economics: a roadmap. In *FOSE '07: 2007 Future of Software Engineering*, pages 304–315, 2007.
41. M. Shepperd and M. Cartwright. Predicting with sparse data. *IEEE Transactions on Software Engineering*, 27:987–998, 2001.
42. M. Shepperd and G. Kadoda. Comparing software prediction models using simulation. *IEEE Trans. Softw. Eng.*, pages 1014–1022, 2001.
43. M. Shepperd and C. Schofield. Estimating software project effort using analogies. *IEEE Trans. Softw. Eng.*, 23(11):736–743, 1997.
44. M. Shepperd, C. Schofield, and B. Kitchenham. Effort estimation using analogy. In *International Conference on Software Engineering*, pages 170–178, 1996.
45. 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.
46. 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.
47. M. P. Wand and M. C. Jones. *Kernel Smoothing (Monographs on Statistics and Applied Probability)*. Chapman & Hall/CRC, December 1994.